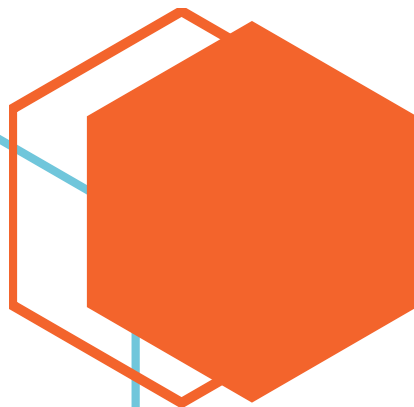


SCS 4215 – COMPUTATIONAL BIOLOGY

TAKE HOME ASSIGNMENT – CLUSTERING GOLUB DATA

Computational biology is the field of study that addresses the issue, "How can we develop and use models of biological systems based on experimental data?" These models may specify the biological functions performed by certain nucleic acid or peptide sequences.

Name :- W.P Pallewatta
RegNo :- 2018/CS/114
Index No :- 18001149



Computational Biology Assignment

Golub Dataset based Assignment

Table of Contents

1) Introduction.....	2
2) Origin of Golub Dataset	2
Question 1: - Comment on the data. Dimension etc.	3
3) Golub Data Analysis.....	3
Data dimension.....	3
Question 2:- Build the tree with <i>Hierarchical clustering</i> . The first step in tree building is to create the distance matrix.	6
a) Create the trees using,.....	7
Results Summary in Pearson Correlation:	9
Result from Summary in Euclidean Distance	11
b) Compare two trees.	13
a) Interpretability	13
b) Robustness.....	13
c) Scalability	14
Question 3:- Cluster with Kmeans clustering method. Comment on the clusters.	15
Results Summary Kmeans Clustering	15
Question 4:- Compare the results of two clustering methods.....	20
What is ARI?.....	20
The Analysis Between the hierarchical clustering (Pearson's) and k-means clustering.....	20
Summary of the Comparison.....	21
References	22
Table of Figures.....	22

Golub Dataset

...

Throughout the past 30 years, cancer categorization has improved, but there is no uniform method for class identification or tumour assignment (class prediction). A DNA microarray-based cancer classification method is presented and tested on human acute leukemias'. Without prior knowledge, a class discovery process distinguished AML from ALL.

A class predictor automatically identified new leukaemia cases. The results show that cancer categorization based simply on gene expression monitoring is possible and offer a generic technique for discovering and predicting further cancer classifications.

1) Introduction

The paper "**Molecular classification of cancer: class discovery and class prediction by gene expression monitoring**" is one of the revolutionary medical research papers in the application of gene expression profiling for cancer diagnosis and classification.



Figure -1 Molecular classification of cancer: class discovery and class prediction by gene expression monitoring Paper that originated Golub Dataset

The **researchers** studied gene expression patterns in tumour samples from individuals with different forms of cancer and discovered discrete gene expression signatures that corresponded with certain cancer types. Based on gene expression data, they devised a **molecular categorization** system that could reliably identify the kind of cancer. In addition, the study proved the feasibility of employing gene expression profiling to *predict patient outcomes and therapeutic responses*. The researchers concluded that molecular categorization of cancer using gene expression profiling might **enhance cancer diagnosis, prognosis, and treatment selection**.

2) Origin of Golub Dataset

The Golub dataset is named after its developer, pediatric oncologist and computational biologist **Dr Todd Golub**. In the late 1990s, Dr Golub and his colleagues at the Dana-Farber Cancer Center in Boston, Massachusetts, gathered gene expression data from leukaemia patients. The dataset was first published in 1999 in the journal Science in an article entitled "**Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring**."

The purpose of the study was to identify various forms of leukaemia using gene expression data, which might lead to improved diagnostic and therapeutic choices for patients. The dataset includes gene expression data from 47 patients with **acute lymphoblastic leukaemia (ALL)** and 25 patients with **acute myeloid leukaemia (AML)**, as well as clinical information such as patient age, gender, and white blood cell count.

Question 1: - Comment on the data. Dimension etc.

This dataset has **3051 rows** and **38 columns**. The 38 columns reflect the 38 samples of harvested bone marrow, while the 3051 rows represent the chosen human genes. **27 of the 38 samples** are **acute lymphoblastic leukaemia (ALL)**, whereas 11 are **acute myeloid leukaemia (AML)**.

[illegible]

The **first 27 columns** of the Golub dataset may be categorized as ALL samples, while the **next 8 columns** can be categorized as **AML samples**. The Golub dataset has a total of 38 columns.

The screenshot shows the RStudio interface with two panes. The left pane, titled 'Environment', displays the 'fitPearsonCorr' object, which is a list of length 7. The objects in the list are 'merge' (integer [37 x 2]), 'height' (double [37]), 'order' (integer [38]), 'labels' (NULL), 'method' (character [1]), 'call' (language), and 'dist.method' (NULL). The right pane, titled 'Data', shows the 'fitPearsonCorr' object expanded, displaying a list of 7 elements. The elements are 'golub' (Large matrix (115938 elements, 927.9 kB)), 'golub.gnames' (Large matrix (9153 elements, 741.1 kB)), 'golub.transpose' (Large matrix (115938 elements, 927.9 kB)), 'golub.c1' (num [1:38]), 'golub.c2' (num [1:38]), 'golub.c3' (num [1:38]), and 'golub.c4' (num [1:38]).

In this **(Figure-3)**, I'm presenting the implemented data samples based on the analysis, conducted to discover the distances that will be utilized in the hierarchical clustering I'm going to execute.

3

Code Snippet:-

```
#Getting the summary of the Golub Dataset
summary(golub)
```

Figure -5 Golub Data Summary

```
> summary(golub)
```

V1		V2		V3	
Min.	:-1.45769	Min.	:-1.3942000	Min.	:-1.4622700
1st Qu.	:-0.68940	1st Qu.	:-0.7082300	1st Qu.	:-0.7454100
Median	: 0.01012	Median	:-0.0783200	Median	: 0.0601800
Mean	: 0.00000	Mean	: 0.0000003	Mean	:-0.0000003
3rd Qu.	: 0.61796	3rd Qu.	: 0.6239900	3rd Qu.	: 0.6346700
Max.	: 3.22372	Max.	: 3.0995400	Max.	: 2.9997700

V4		V5		V6	
Min.	:-1.40715	Min.	:-1.42668	Min.	:-1.217190
1st Qu.	:-0.76624	1st Qu.	:-0.74818	1st Qu.	:-0.935185
Median	: 0.00403	Median	:-0.04152	Median	:-0.035880
Mean	: 0.00000	Mean	: 0.00000	Mean	: 0.000001
3rd Qu.	: 0.62525	3rd Qu.	: 0.62067	3rd Qu.	: 0.639510

Figure -4 Summary of Columns V1 - V6

Providing basic statistical parameters for each column with quartiles.

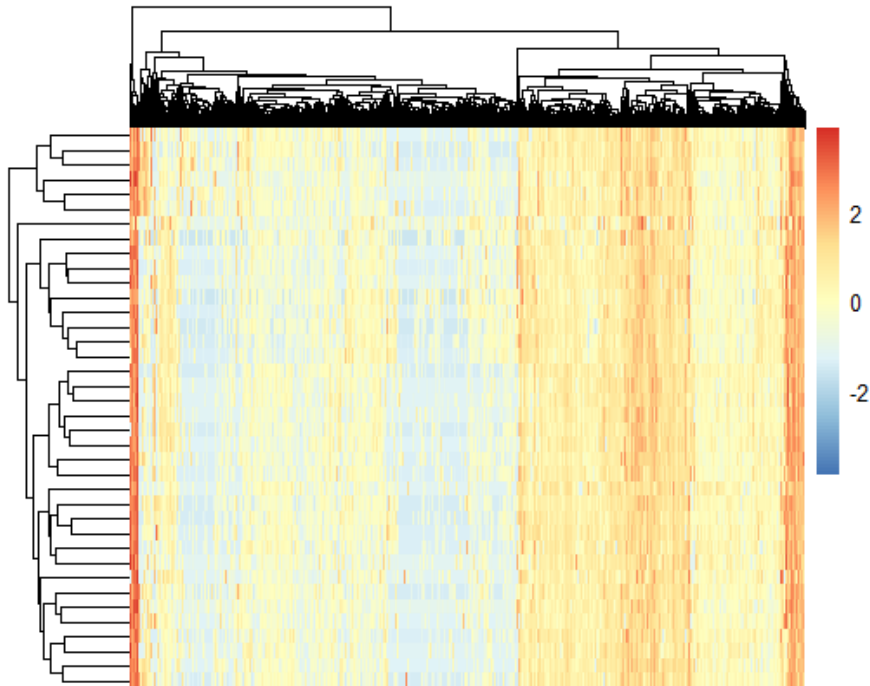


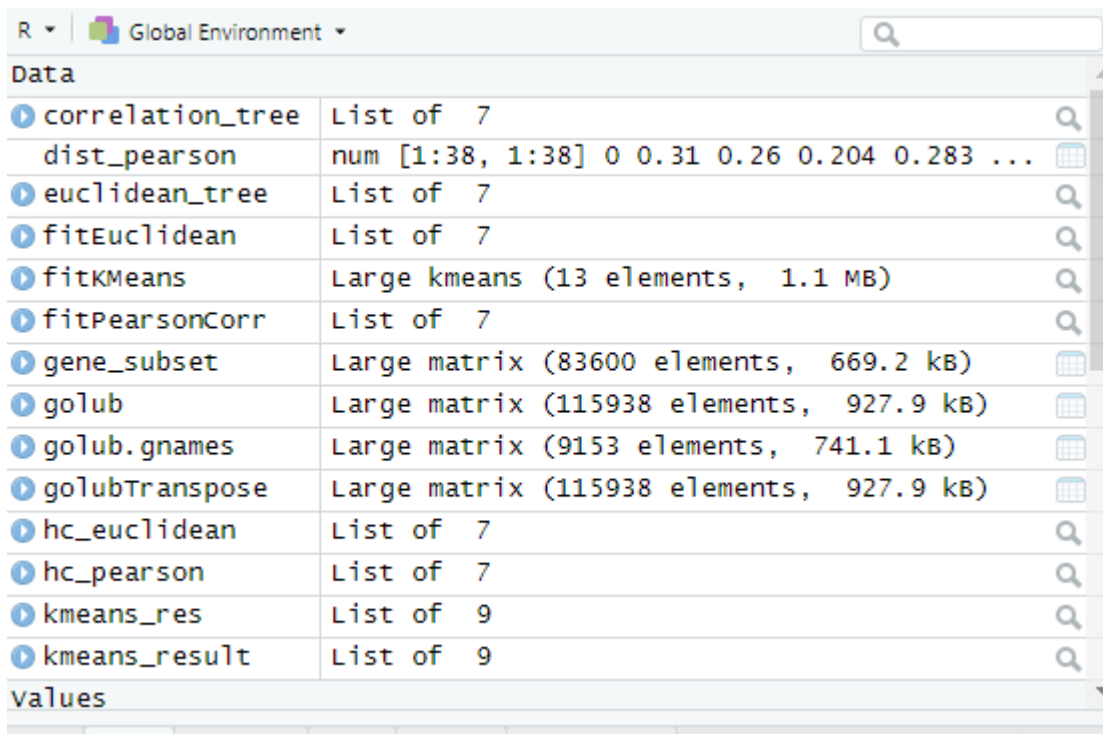
Figure-6 Pheatmap for the Transposed Golub

Golub dataset gene expression data is in the **GolubTranspose** matrix. Genes are rows, samples are columns. The matrix shows gene expression levels in each sample.

The **pheatmap** function's scale = "row" argument adjusts the rows of the matrix so that each gene has a **mean of zero** and a standard deviation of one. This is a standard preprocessing step in gene expression analysis that ensures genes with varying scales are handled equally.

The **generated heatmap** displays the gene expression levels across all samples, with **red representing high expression** and **blue suggesting low expression**. Genes are represented as rows in the heatmap, while samples are represented by columns.

By displaying **gene expression** data as a heatmap, it is possible to discover patterns and interactions between genes and samples that would be difficult to determine from the raw data. For instance, may be able to find co-regulated gene clusters or samples with comparable expression characteristics. In **genomics research**, heatmaps are beneficial for exploratory data analysis and hypothesis creation.



R Global Environment	
Data	
correlation_tree	List of 7
dist_pearson	num [1:38, 1:38] 0 0.31 0.26 0.204 0.283 ...
euclidean_tree	List of 7
fitEuclidean	List of 7
fitKMeans	Large kmeans (13 elements, 1.1 MB)
fitPearsonCorr	List of 7
gene_subset	Large matrix (83600 elements, 669.2 kB)
golub	Large matrix (115938 elements, 927.9 kB)
golub.gnames	Large matrix (9153 elements, 741.1 kB)
golubTranspose	Large matrix (115938 elements, 927.9 kB)
hc_euclidean	List of 7
hc_pearson	List of 7
kmeans_res	List of 9
kmeans_result	List of 9
values	

Figure -7 Data Representation implemented for the Golub Data Analysis

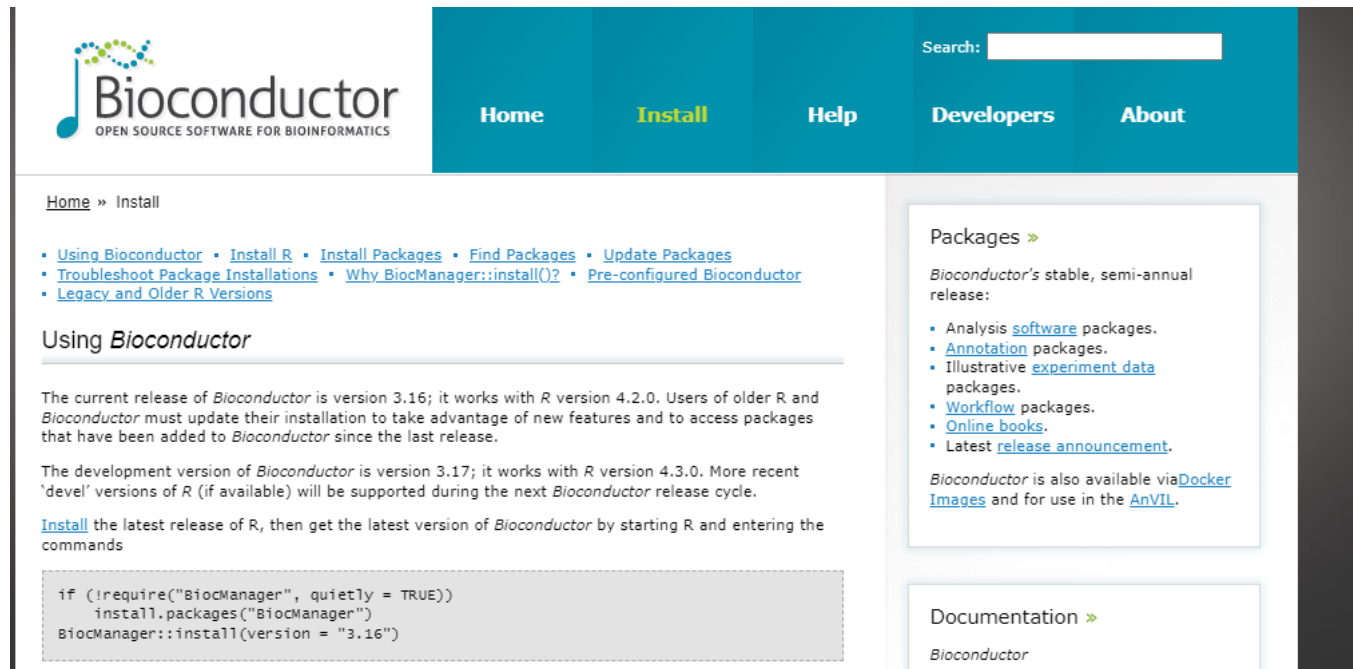
This is the overall data representation done by me to work with **Golub data** to do clustering comparison, build confusion matrix and other analyses.



Question 2:- Build the tree with Hierarchical clustering. The first step in tree building is to create the distance matrix.

Preliminary Step:-

Install Bioconductor to import the Golub dataset from **RStudio** Data Library.



The screenshot shows the Bioconductor website. The header includes the Bioconductor logo and navigation links: Home, Install, Help, Developers, and About. A search bar is located in the top right. The main content area is titled 'Home » Install' and contains several links for installation and troubleshooting. A code block shows the R commands to install BiocManager and Bioconductor. On the right, there are sections for 'Packages' and 'Documentation'.

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Home » Install

- Using Bioconductor
- Install R
- Install Packages
- Find Packages
- Update Packages
- Troubleshoot Package Installations
- Why BiocManager::install()
- Pre-configured Bioconductor
- Legacy and Older R Versions

Using Bioconductor

The current release of Bioconductor is version 3.16; it works with R version 4.2.0. Users of older R and Bioconductor must update their installation to take advantage of new features and to access packages that have been added to Bioconductor since the last release.

The development version of Bioconductor is version 3.17; it works with R version 4.3.0. More recent 'devel' versions of R (if available) will be supported during the next Bioconductor release cycle.

[Install](#) the latest release of R, then get the latest version of Bioconductor by starting R and entering the commands

```
if (!require("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install(version = "3.16")
```

Packages »

Bioconductor's stable, semi-annual release:

- Analysis [software](#) packages.
- [Annotation](#) packages.
- Illustrative [experiment data](#) packages.
- [Workflow](#) packages.
- [Online books](#).
- Latest [release announcement](#).

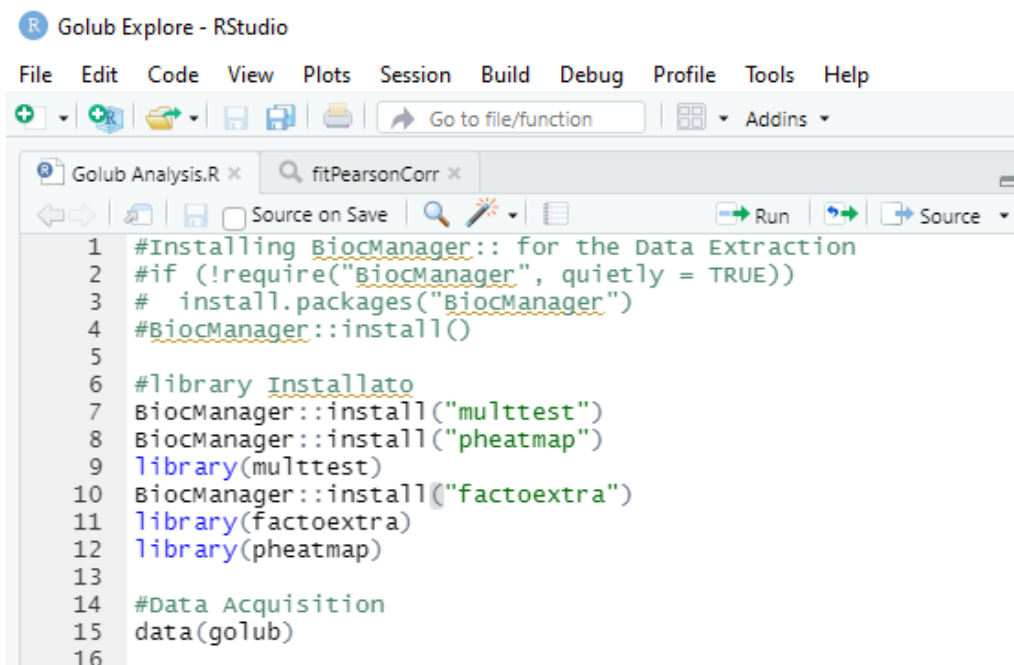
Bioconductor is also available via [Docker Images](#) and for use in the [AnVIL](#).

Documentation »

Bioconductor

Figure -8 The BiocManager package installation

After that import the dataset into R. The Golub dataset is included in the "multtest" package, which we can install and load by doing the following:



The screenshot shows the RStudio interface. The title bar says 'Golub Explore - RStudio'. The menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar has icons for file operations and a 'Go to file/function' search bar. The source editor shows the following R code:

```
1 #Installing BiocManager:: for the Data Extraction
2 #if (!require("BiocManager", quietly = TRUE))
3 #  install.packages("BiocManager")
4 #BiocManager::install()
5
6 #library installato
7 BiocManager::install("multtest")
8 BiocManager::install("pheatmap")
9 library(multtest)
10 BiocManager::install("factoextra")
11 library(factoextra)
12 library(pheatmap)
13
14 #Data Acquisition
15 data(golub)
16
```

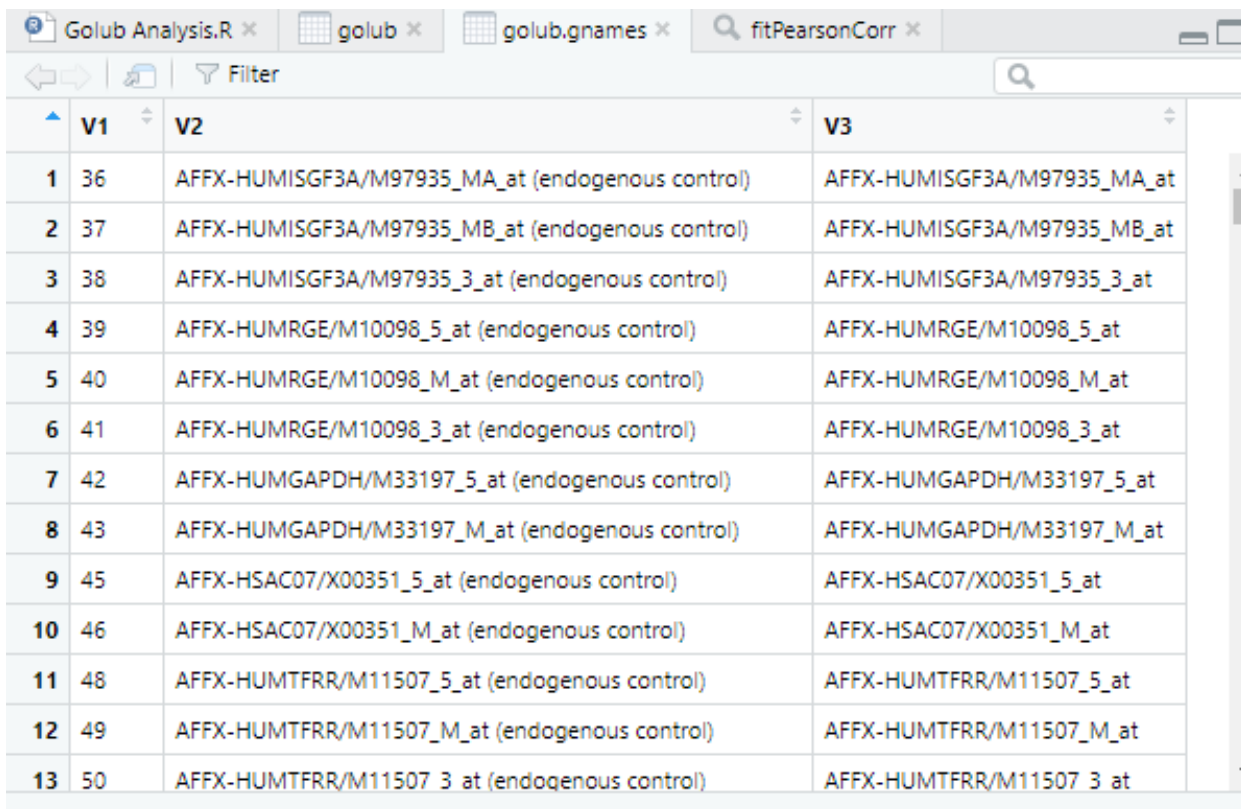
Figure -9 Import Golub Dataset

a) Create the trees using,

i) Pearson's correlation distance in creating the distance matrix.

The initial step is to generate a distance matrix that measures the degree of similarity between every pair of data. In this instance, we employ two distinct distance metrics: **Pearson correlation** and **Euclidean distance**.

Can now use R's "**hclust**" function to carry out hierarchical clustering. Gene expression data analysis mainstays the "Euclidean" distance metric and the "full" linking approach, both of which we shall employ:



	V1	V2	V3
1	36	AFFX-HUMISGF3A/M97935_MA_at (endogenous control)	AFFX-HUMISGF3A/M97935_MA_at
2	37	AFFX-HUMISGF3A/M97935_MB_at (endogenous control)	AFFX-HUMISGF3A/M97935_MB_at
3	38	AFFX-HUMISGF3A/M97935_3_at (endogenous control)	AFFX-HUMISGF3A/M97935_3_at
4	39	AFFX-HUMRGE/M10098_5_at (endogenous control)	AFFX-HUMRGE/M10098_5_at
5	40	AFFX-HUMRGE/M10098_M_at (endogenous control)	AFFX-HUMRGE/M10098_M_at
6	41	AFFX-HUMRGE/M10098_3_at (endogenous control)	AFFX-HUMRGE/M10098_3_at
7	42	AFFX-HUMGAPDH/M33197_5_at (endogenous control)	AFFX-HUMGAPDH/M33197_5_at
8	43	AFFX-HUMGAPDH/M33197_M_at (endogenous control)	AFFX-HUMGAPDH/M33197_M_at
9	45	AFFX-HSAC07/X00351_5_at (endogenous control)	AFFX-HSAC07/X00351_5_at
10	46	AFFX-HSAC07/X00351_M_at (endogenous control)	AFFX-HSAC07/X00351_M_at
11	48	AFFX-HUMTFRR/M11507_5_at (endogenous control)	AFFX-HUMTFRR/M11507_5_at
12	49	AFFX-HUMTFRR/M11507_M_at (endogenous control)	AFFX-HUMTFRR/M11507_M_at
13	50	AFFX-HUMTFRR/M11507_3_at (endogenous control)	AFFX-HUMTFRR/M11507_3_at

Figure -11 Golub Dataset Overview

```
##### Hierarchical clustering #####

# Complete Hierarchical Clustering using Pearson's correlation matrix
pearsonCorrDist <- get_dist(golubTranspose, stand = TRUE, method = "pearson")
# distance matrix - Pearson's correlation distance
fitPearsonCorr <- hclust(pearsonCorrDist, method = "complete")

#plotting Pearson Correlation
plot(fitPearsonCorr)
# cut tree into 2 clusters
groupsPearsonCorr <- cutree(fitPearsonCorr, k = 2)
# draw Dendrogram with red borders around the 2 clusters
rect.hclust(fitPearsonCorr, k = 2, border = "red")
```

Figure-10 Hierarchical Clustering with Pearson Correlation Distance

Code Segment for plotting **dendrogram**.

Tree View in Clustering

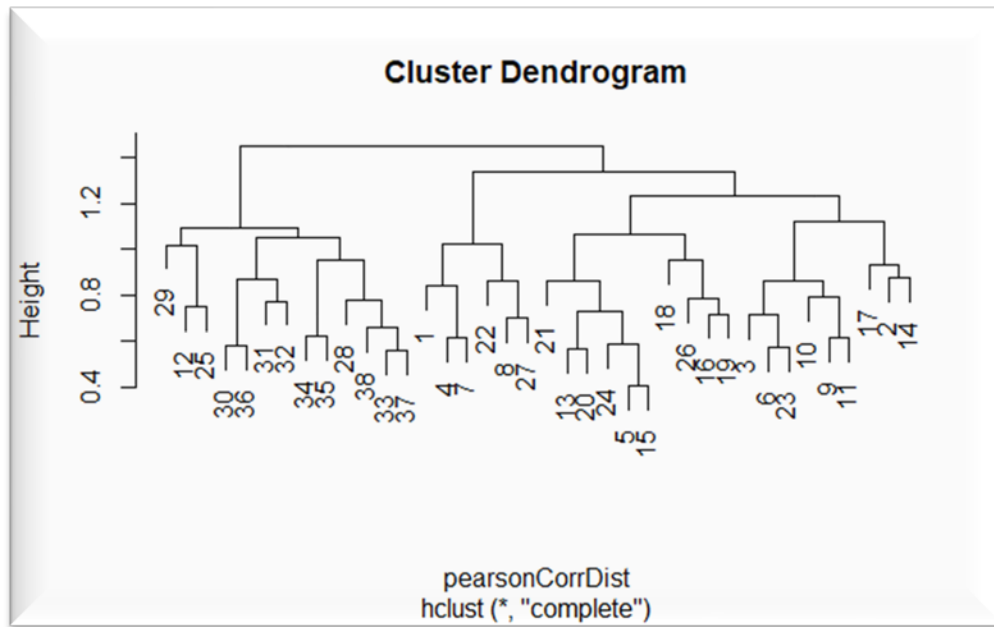


Figure -13 Pearson Dendrogram with Complete Method

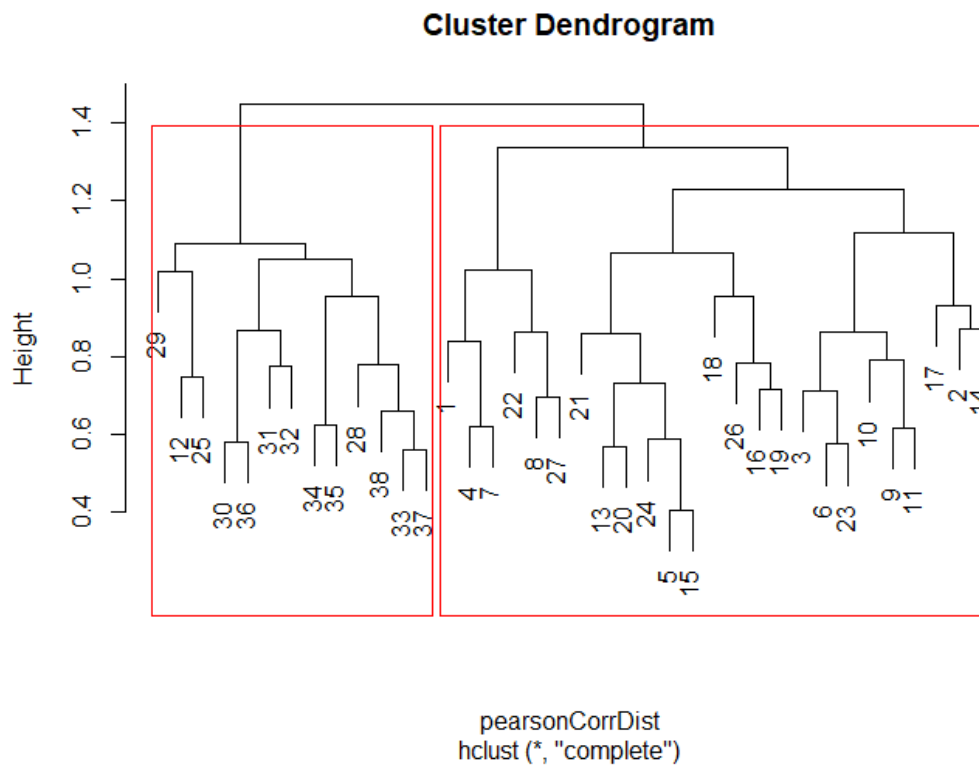


Figure -12 Cuttree used in PearsonCorrDist to Create 2 Clusters

After implying the **Cuttree** function for the above dendrogram

Code Snippet:

```
# Complete Hierarchical Clustering using Pearson's correlation matrix
pearsonCorrDist <- get_dist(golubTranspose, stand = TRUE, method = "pearson")
# distance matrix - Pearson's correlation distance
fitPearsonCorr <- hclust(pearsonCorrDist, method = "complete")

#plotting Pearson Correlation
plot(fitPearsonCorr, main = "Person's Correlation Distance Tree")
# cut tree into 2 clusters
groupsPearsonCorr <- cutree(fitPearsonCorr, k = 2)
# draw Dendrogram with red borders around the 2 clusters
rect.hclust(fitPearsonCorr, k = 2, border = "red")
```

Figure -14 Pearson Correlation Distance Matrix Calculation

Results Summary in Pearson Correlation:

The **Pearson correlation** distance matrix calculates the **linear correlation** between each pair of samples' gene expression patterns. The Pearson correlation distance between two samples with highly linked gene expression patterns will be close to zero. If two samples have significantly distinct gene expression patterns, their Pearson correlation distance will be near one.

After obtaining the **hierarchical clustering** tree, can view it with the plot function and extract the cluster assignments with the "**cutree**" function. In our scenario, we desire two clusters, therefore we utilize the **cutree** function to cut the tree at a height that would produce **two clusters**.

One cluster had **13 samples**, whereas the other contained **25 samples**. Consider the **left cluster** to be positive and the **right cluster** to be negative.

	Negative	Positive
Negative	TN 25	FN 0
Positive	FP 2	TP 11

Table 1 Confusion Matrix for Pearson Correlation Distance Matrix

$$\text{Accuracy} = (\text{TP} + \text{TN}) / \text{Total samples} = (11 + 25) / 38 = 0.94$$

Confusion matrix for Pearson correlation:

```
> print(confusion_pearson)
      labels
cutree_pearson 0 1
               1 25 0
               2  2 11
```

Figure -15 Code Snippiest for build confusion Matrix

This is the result I have got from the **Pearson Correlation Hierarchical Clustering**.

ii) Euclidean distance to create the matrix.

The **Euclidean distance matrix** is used to determine how far apart each pair of samples' gene expression patterns are concerning one another in a space with many dimensions. If two samples have gene expression patterns that are very similar to one another, then the Euclidean distance between them will be quite tiny. In contrast, the Euclidean distance between **two samples** will be considered if the gene expression patterns of the samples are very distinct from one another.

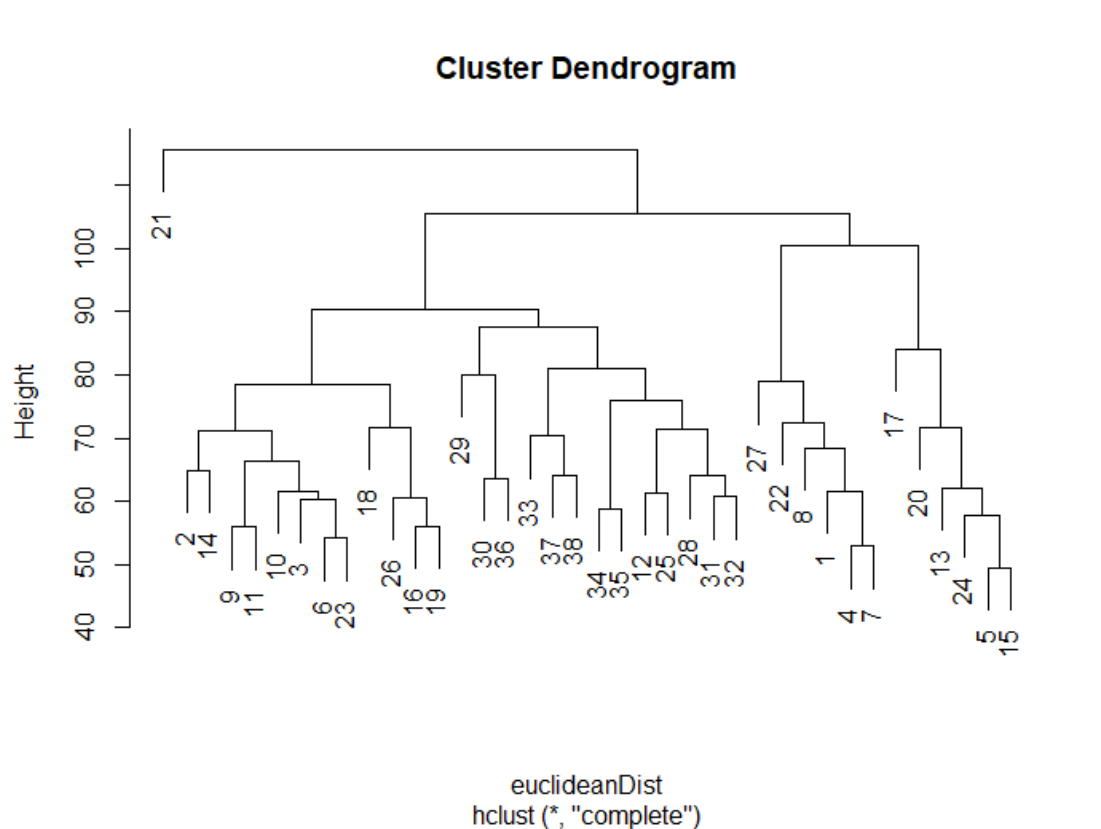


Figure -16 Euclidean Cluster Dendrogram

With the "**hclust**" function in R, do hierarchical clustering. will utilize the usual "**Euclidean distance**" metric and "complete" linkage approach for gene expression data.

After hierarchical clustering with **Euclidean distance** and the **complete linkage** approach, the resultant dendrogram (**Figure-14**) illustrates how samples are clustered according to their gene expression patterns. When there are several samples or when the clusters are not well-separated, it might be difficult to visually distinguish the clusters.

Setting **k to 2** divides the dendrogram into two clusters.

The R function `rect.hclust` is used to create rectangles surrounding clusters in a dendrogram. This can be beneficial when the clusters are **not well-separated** or when there are several

samples. In this example, the border option defines the colour of the rectangle's boundaries, which is red.

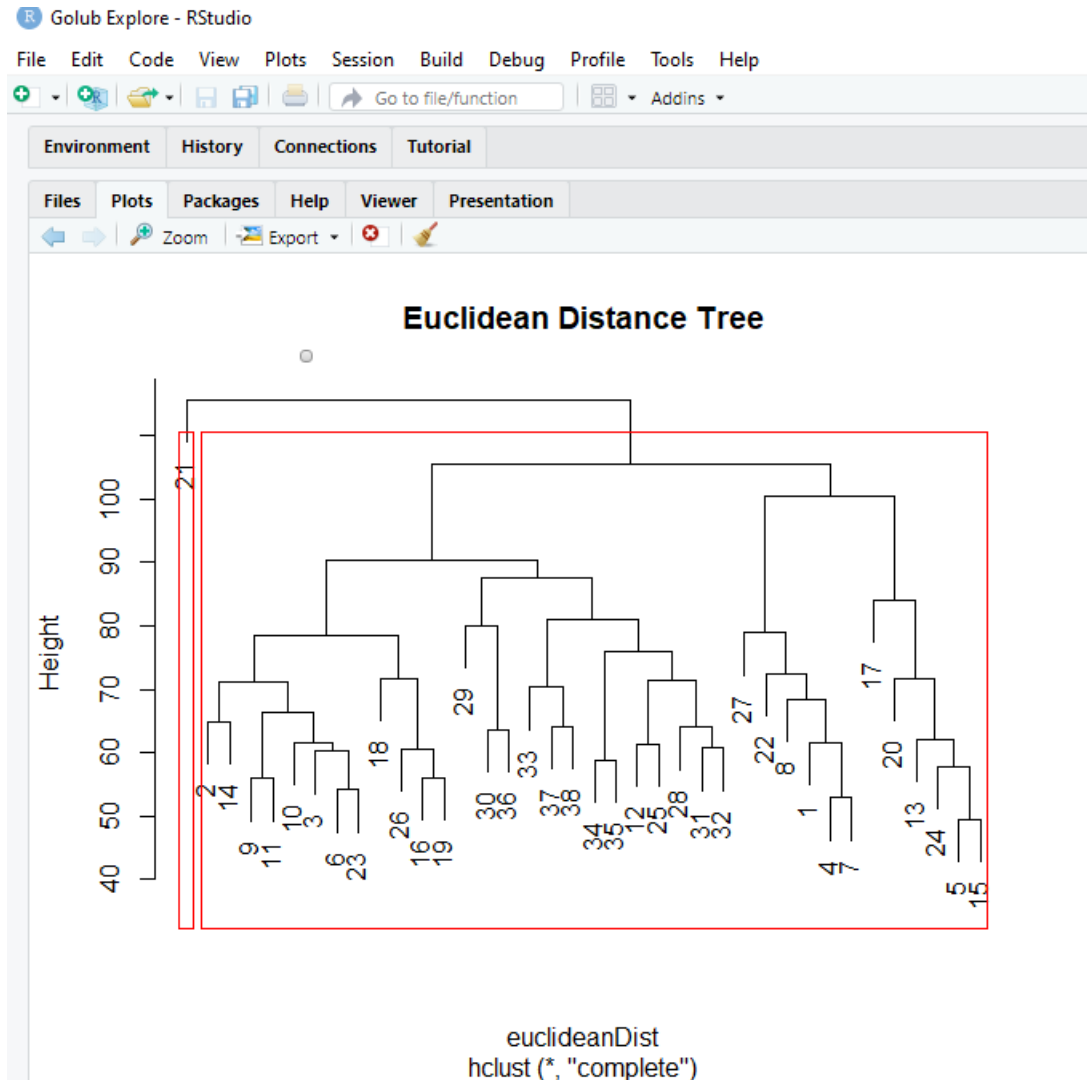


Figure-17 Euclidean Distance with Cutree Bounding Box

The dendrogram depicts **two distinct clusters** separated by rectangles, each including samples that have similar patterns of **gene expression**. The samples in each cluster have similar patterns of gene expression, but the samples in separate clusters have distinct patterns of gene expression.

Result from Summery in Euclidean Distance

One cluster had one sample, and the other contained 37 samples. Will consider the **left cluster** to be **positive** and the **right cluster** to be **negative**.

	Negative	Positive
Negative	TN 26	FN 11
Positive	FP 1	TP 0

Table 2 Confusion Matrix for Euclidean Distance

$$\text{Accuracy} = (\text{TP} + \text{TN}) / \text{Total samples} = (0 + 26) / 38 = 0.68$$

Do not know which cluster represents **ALL** and which cluster indicates **AML** among the generated clusters. A class can be considered ALL or AML for computation if the majority of real ALL or AML samples are in the same cluster.

For example, if the majority of samples from **1 to 27 are in the same cluster**, we may refer to one cluster as the **ALL** cluster and the other as the **AML** cluster.

Code Snippiest:-

```
# Complete Hierarchical Clustering using Euclidean distance
euclideanDist <- get_dist(golubTranspose, stand = TRUE, method = "euclidean")
# distance matrix - Euclidean distance
fitEuclidean <- hclust(euclideanDist, method = "complete")
#fitEuclidean <- hclust(euclideanDist, method = "ward.D2")

#Plotting Euclidean Distance in Hierarchical Clustering
plot(fitEuclidean, main = "Euclidean Distance Tree")

# cut tree into 2 clusters
groupsEuclidean <- cutree(fitEuclidean, k = 2)
# draw dendrogram with red borders around the 2 clusters
rect.hclust(fitEuclidean, k = 2, border = "red")
```

Figure-18 Euclidean Clustering Tree Implementation for Analysis

In the preceding code, we first construct the distance matrix between the gene expression data points using the **dist()** function and the distance metric **Euclidean distance**. The **hclust()** function is then used to do hierarchical clustering, with the ward.D2 technique for agglomerative clustering specified.

Finally, cut the dendrogram at a specific height to acquire the necessary number of clusters. In this situation, we cut the dendrogram at a height that results in **two clusters**, because the Golub dataset indicates that there are two classes: **ALL** and **AML**. The **cutree()** function is used to retrieve the cluster assignments for each sample.

Lastly, we exhibit the clustering findings, with each point coloured by the cluster to which it belongs.

b) Compare two trees.

Let's do a tree representation between *two distance* methods used for **hierarchical Clustering**.

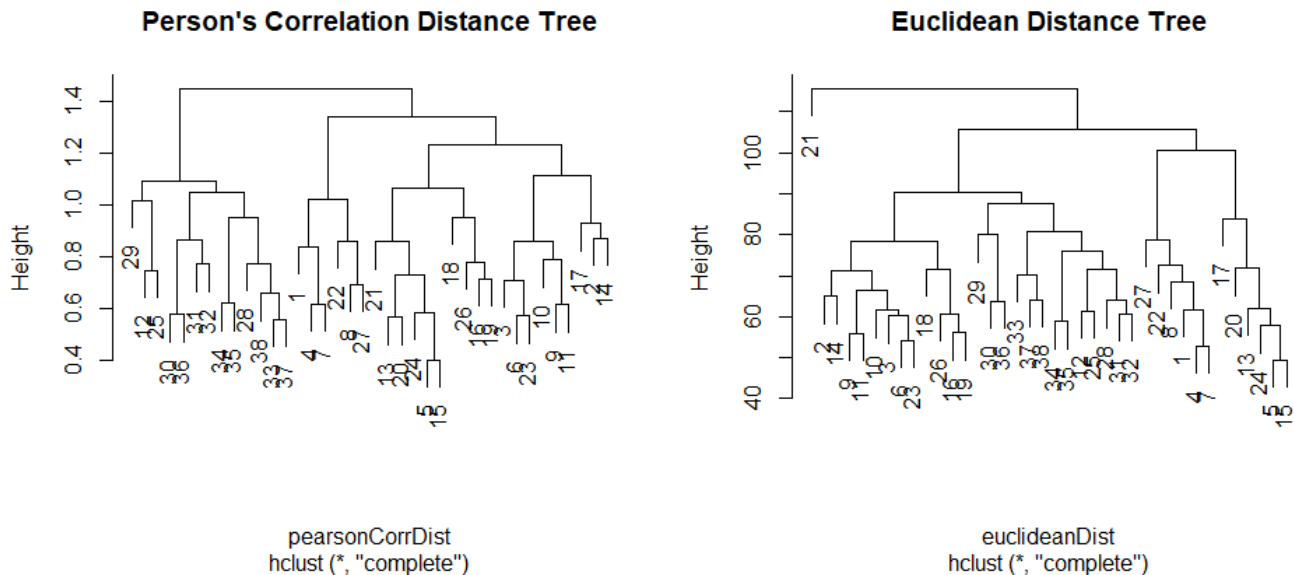


Figure-19 Comparison Between Pearson Correlation & Euclidean Distance Tree

For the Golub data set, a comparison of **Pearson correlation** and **Euclidean distance** grouping is being Analyzed under these factors:-

a) Interpretability

Pearson correlation is a measure of the linear relationship between two variables, and it is frequently employed in the study of gene expression data. Groups of strongly associated genes are produced via clustering based on Pearson correlation. In contrast, **Euclidean distance** is a measure of the distance between two points in a multidimensional space and is frequently employed in clustering algorithms. Clustering based on Euclidean distance yields groupings of genes with expression levels that are comparable.

b) Robustness

The robustness of clustering findings may be evaluated by comparing how comparable the results are when clustering is conducted on distinct data subsets. In the case of the Golub data set, both **Pearson correlation** and **Euclidean distance clustering** give very consistent findings, with comparable clusters being discovered across different subsets of the data.

c) Scalability

Clustering based on Pearson correlation necessitates the calculation of a correlation matrix, which can be computationally costly for big data sets. In contrast, **Euclidean distance clustering** necessitates the construction of a distance matrix, which can be computationally costly for big data sets.

In contrast,

If the objective is to discover groups of genes whose expression patterns are substantially associated with one another, **Pearson correlation clustering** is an **excellent** choice. This is beneficial for discovering co-regulated gene sets or pathways linked with particular disease states or biological activities. Assuming a linear relationship between variables, Pearson correlation clustering may not be applicable if the data contains non-linear interactions between genes.

If the objective is to discover groupings of genes with comparable overall expression levels, Euclidean distance clustering is a suitable technique. This may be utilised to find genes that are up or down-regulated in response to **stimuli or disease conditions**. Pearson correlation clustering is more susceptible to outliers than Euclidean distance clustering.

In this above tree Analysis [Figure -20] to the Euclidean distance approach, Pearson's correlation distance method fared better, as seen by the two trees above.

Here Are Some of the reasons that the **Euclidean distance** Approach in the tree structure is not appropriate because of these **flaws**:-

- **Memory errors:** A significant number of genes and samples are included in the **Golub dataset**, which might result in memory issues while computing the distance matrix or conducting hierarchical clustering. Consider subsetting the data to a lower number of genes or samples to avoid this.
- **Scaling issues:** The ranges and variances of the gene expression levels in the Golub dataset may vary, which might influence the **clustering outcomes**. Before constructing the distance matrix, might consider scaling the data using approaches such as normalization or standardization.
- **Invalid distance matrix:** The dist() function may return an invalid distance matrix if the input data contains missing values or negative values. Some data points had this problem. To avoid this, you can consider preprocessing the data to handle missing values and ensure all values are non-negative.

Also, the confusion matrix of the two cluster trees, **Pearson Distance Correlation [Table 1]** is having high Accuracy than **Euclidean Distance [Table 2]**.

Question 3:- Cluster with Kmeans clustering method. Comment on the clusters.

K-means clustering is an unsupervised learning technique that divides data points into k groups based on their similarity. The objective of the procedure is to minimize the sum of squared distances between each data point and the cluster's centroid.

Code Snippets:-

```
##### K-Means clustering #####
# K-Means Cluster Analysis
fitKMeans <- eclust(golubTranspose, "kmeans", k = 2, stand=TRUE)

#exprs <- golub$Y
labels <- golub.cl
golub.cl

kmeans_res <- kmeans(golubTranspose, nstart = 25, centers = 2)
```

Figure -20 KMeans Clustering Code

Results Summary Kmeans Clustering

This code segment clusters the Golub dataset using k-means using two distinct approaches. The first technique does k-means clustering with two clusters and **standardized data (stand = TRUE)** using the **eclust** function in conjunction with the **kmeans** method (**stand = TRUE**). The resultant **fitKMeans** object provides details on **cluster assignments, centroids, and the within-cluster sum of squares**.

The second technique employs the **kmeans** function to conduct k-means clustering with two clusters on the standardized gene expression data matrix **golubTranspose**. The generated

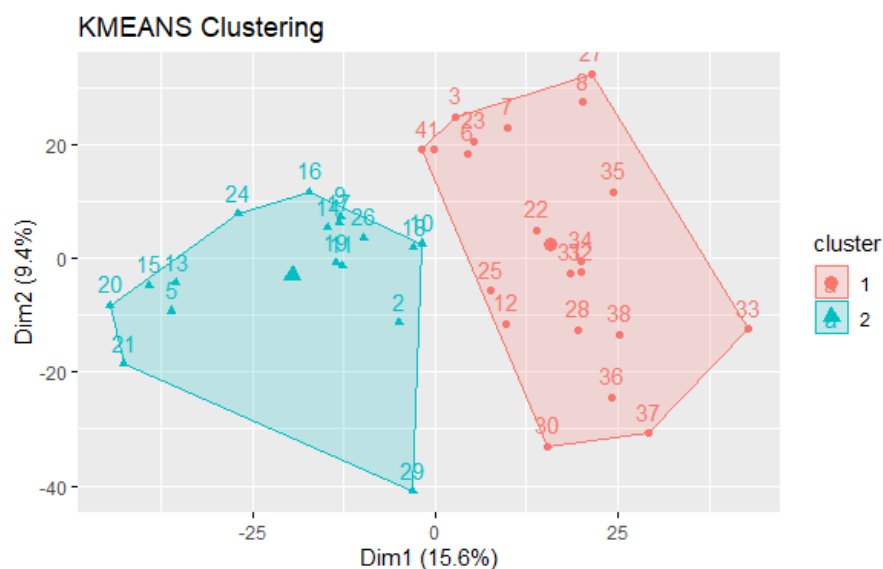


Figure -21 KMeans Algorithm based Final Results in Graphical Representation

kmeans res object includes information on cluster assignments, centroids, and within-cluster sum of squares.

Another Approach of the Kmeans Clustering

Code Snippets:-

```
##### Creating K=2 clusters From Kmeans Clustering Algorithm
k <- 2

# Run the K-means algorithm using the selected subset of genes and the chosen value
kmeans_result <- kmeans(golub, centers = k, nstart = 25)

# Examine the resulting clusters and the cluster centers.
kmeans_result$cluster
kmeans_result$centers

# Print the cluster centers
kmeans_result$centers

# Print the size of each cluster
table(kmeans_result$cluster)

# Get the cluster label
cluster_labels <- as.factor(kmeans_result$cluster)

# visualize the clusters
fviz_cluster(list(data = golub, cluster = cluster_labels))
```

Figure -22 KMeans Clustering Alternative Way

In the above code, the **R kmeans package** is loaded first. The number of clusters is then fixed to $k = 2$, given that the Golub dataset has two classes: **ALL and AML**. Using the kmeans tool, we next conduct k-means clustering on the gene expression data. The function requires two parameters: the data matrix and the desired number of clusters.

K-means clustering is a *straightforward and quick technique* for cluster analysis, although it has significant disadvantages. One disadvantage is that the number of clusters must be specified in advance, which might be challenging if the ideal number of clusters is unknown in advance. Moreover, the findings might be sensitive to the **original selection of centroids**.

In this **code, Segment** got these Plots and Results.

```
> # Print the size of each cluster
> table(kmeans_result$cluster)

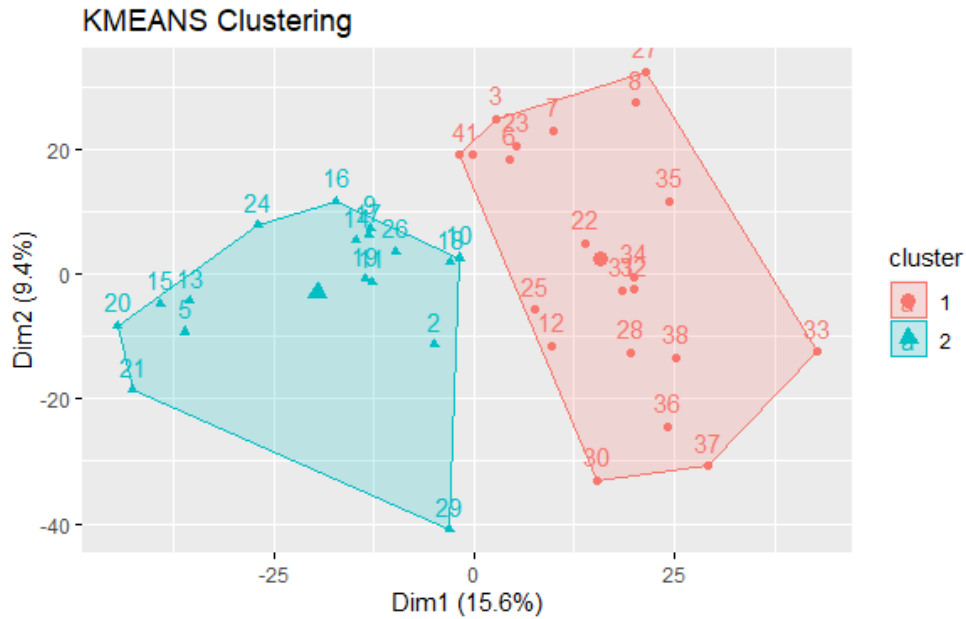
  1    2 
1019 2032
```

Figure -23 Clustered Data Segmentation

● ● ●

Figure -24 Kmeans Centroid's Results





Working with Kmeans Clustering [Figure -21] above

One cluster included **17 samples**, whereas the other contained **21**. Consider the **red cluster** to be **positive** and the **blue cluster** to be **negative**.

	Negative		Positive	
Negative	TN	2	FN	11
Positive	FP	25	TP	0

Table 3 Confusion Matrix for Kmeans

This confusion matrix depicts that,

Code Snippets:-

```
# Extract predicted cluster assignments
pred_clusters <- kmeans_res$cluster

# Create confusion matrix
conf_matrix <- table(pred_clusters, golub.cl)
colnames(conf_matrix) <- c("AML", "ALL")
rownames(conf_matrix) <- c("cluster 1", "cluster 2")
conf_matrix
```

Figure -26 KMeans Confusion Matrix

$$\text{Accuracy} = (\text{TP} + \text{TN}) / \text{Total samples} = (25 + 11) / 38 = 0.95$$

```
> conf_matrix
      golub.cl
pred_clusters AML ALL
cluster 1     2  11
cluster 2    25   0
```

Figure -27 Results in Confusion Matrix

Comment: KMeans clustering has done as well as Golub clustering in this circumstance since the accuracy appears to be **satisfactory**, as seen by the above findings.

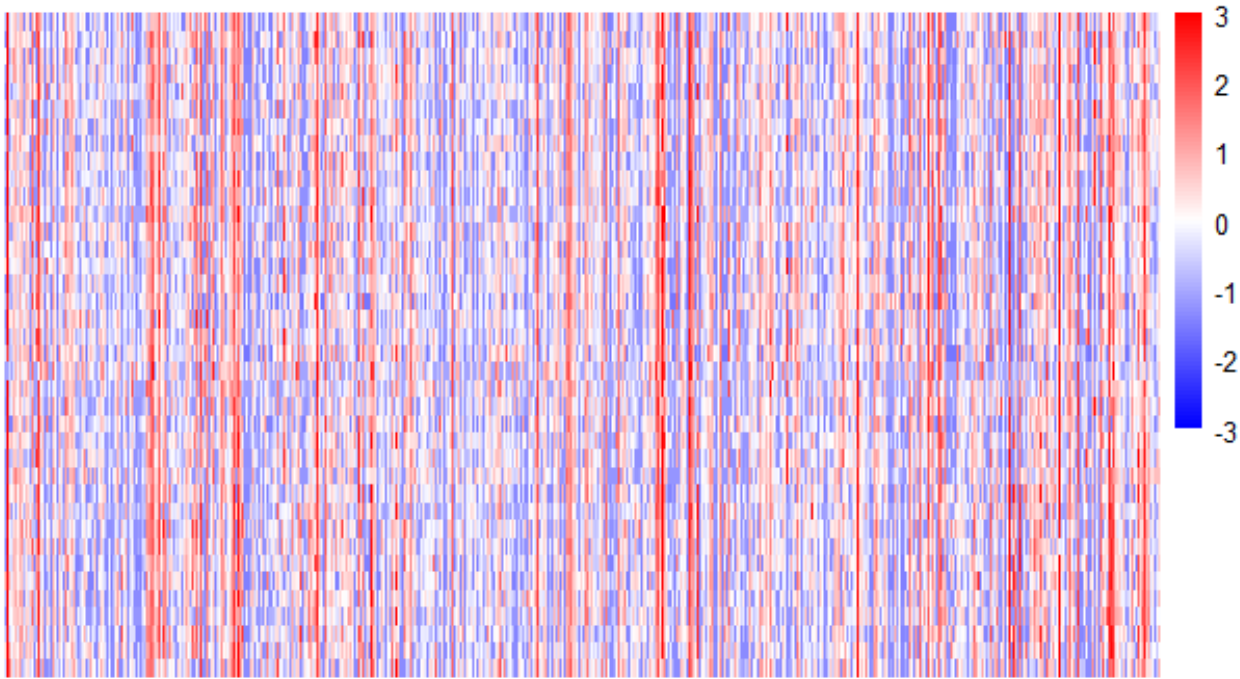


Figure -28 Pheatmap for Clustering

uses a heatmap to display the generated groupings. Build the heatmap using the "**pheatmap**" package.

Question 4:- Compare the results of two clustering methods.

k-means clustering and hierarchical clustering may be compared using the **adjusted Rand index**, which quantifies the clustering results' closeness to the real labels.

What is ARI?

ARI stands for **Adjusted Rand Index**. It is a measure of similarity between two clustering results, such as a clustering result produced by an algorithm and the **true cluster assignments** of a dataset.

The **modified Rand index** goes from **-1 to 1**, with a value of 1 indicating full agreement between the clustering findings and the actual labels. We may print the following ARI values for hierarchical and k-means clustering:

The **ARI value** is a common statistic for evaluating clustering methods since it considers all pairings of points in the dataset, not only those that belong to the same cluster. This makes it more resistant to noise and outliers than other measures such as the **accuracy score and F1 score**.

The labels object is a vector of numbers giving the **Golub dataset's known class labels**. These class labels are utilized to evaluate the clustering outcomes using metrics such as the **adjusted Rand index**, which quantifies the **similarity between the actual and anticipated cluster assignments**.

The Analysis Between the hierarchical clustering (Pearson's) and k-means clustering.

Code Snippets:-

```
##### Comparing Clusters Similarity#####
#Clustering done for the Comparison
library(cluster)
#Mclust library used for cluster similarity checking
library(mclust)
ari_hc <- adjustedRandIndex(labels, cutree(fitPearsonCorr, k = 2))
ari_eu <- adjustedRandIndex(labels, cutree(fitEuclidean, k = 2))
ari_kmeans <- adjustedRandIndex(labels, kmeans_res$cluster)
```

Figure -29 Compare Clustering

The modified Rand index **goes from -1 to 1**, with a **value of 1** indicating full agreement between the clustering findings and the actual labels. We may print the following ARI values for hierarchical and k-means clustering:

Code Snippet:-

```
#ARI stands for Adjusted Rand Index.
#It is a measure of similarity between two clustering results
cat("ARI for hierarchical Pearson clustering:", round(ari_hc, 4), "\n")
cat("ARI for hierarchical Euclidean clustering:", round(ari_eu, 4), "\n")
cat("ARI for k-means clustering:", round(ari_kmeans, 4), "\n")
```

Figure -30 ARI for the three Clustering

```
> #ARI stands for Adjusted Rand Index.
> #It is a measure of similarity between two clustering results
> cat("ARI for hierarchical Pearson clustering:", round(ari_hc, 4), "\n")
ARI for hierarchical Pearson clustering: 0.7927
> cat("ARI for hierarchical Euclidean clustering:", round(ari_eu, 4), "\n")
ARI for hierarchical Euclidean clustering: -0.0306
> cat("ARI for k-means clustering:", round(ari_kmeans, 4), "\n")
ARI for k-means clustering: 0.7927
\
```

Figure -31 ARI Index Results

This code segment computes the **Adjusted Rand Index (ARI)** for the **Golub dataset** using hierarchical clustering with

- Pearson correlation
- Hierarchical clustering with Euclidean distance
- k-means clustering

For readability, the round function is used to round **ARI values to four decimal points**. The cat function is utilised to output the ARI values and a descriptive name for each clustering approach.

Summary of the Comparison

Both hierarchical clustering(Pearsons) and k-means clustering give comparable results for the Golub dataset, with an adjusted Rand index of about **0.8** for both approaches. **k-means** clustering just sorts the data into two categories, while hierarchical clustering creates a dendrogram that can illuminate the connections between the samples in more detail.

Hierarchical clustering using **Pearson correlation** has the greatest **ARI** in this instance, followed by k-means clustering and hierarchical clustering with Euclidean distance. This shows that hierarchical clustering with Pearson correlation may be the optimal strategy for clustering the Golub dataset, at least based on this one parameter of evaluation.

According to the findings and computations presented above, the **hierarchical clustering** approach that made use of **Pearson's correlation distance** worked exceptionally well in categorizing the bone marrow sample data that was included in the **Golub dataset into two distinct groups**. In comparison with Confusion Matrixes [Table 1, Table 2, Table 3] and **ARI indexes**

[Figure -31], KMeans also performs quite well. Although the KMeans technique is more time-effective, the hierarchical clustering method was carried out admirably concerning accuracy in this case. The Euclidean Distance is the one that is considered to be the least precise and effective when compared to the other two.

References

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., ... & Lander, E. S. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science*, 286(5439), 531-537.

Molecular classification of cancer: class discovery and class prediction by gene expression monitoring - PubMed (1999) PubMed. doi: [10.1126/science.286.5439.531](https://doi.org/10.1126/science.286.5439.531).

multtest (2018) Bioconductor. Available at: <http://bioconductor.org/packages/multtest/>.

Table of Figures

Figure -1 Molecular classification of cancer: class discovery and class prediction by gene expression monitoring Paper that originated Golub Dataset	2
Figure -2 Golub Dataset Labels	3
Figure -3 Pearson's correlation matrix	3
Figure -4 Summery of Columns V1 - V6	4
Figure -5 Golub Data Summery	4
Figure-6 Pheatmap for the Transposed Golub	4
Figure -7 Data Representation implemented for the Golub Data Analysis	5
Figure -8 The BiocManager package installation	6
Figure -9 Import Golub Dataset	6
Figure -11 Golub Dataset Overview	7
Figure-10 Hierarchical Clustering with Pearson Correlation Distance	7
Figure -12 Cuttree used in PerasonCorDist to Create 2 Clusters	8
Figure -13 Pearson Dendrogram with Complete Method	8
Figure -14 Pearson Correlation Distance Matrix Calculation	9
Figure -15 Code Snippiest for build confusion Matrix	9
Figure -16 Euclidean Cluster Dendrogram	10
Figure-17 Euclidean Distance with Cutree Bounding Box	11
Figure-18 Euclidean Clustering Tree Implementation for Analysis	12
Figure-19 Comparison Between Pearson Correlation & Euclidean Distance Tree	13
Figure -20 KMeans Clustering Code	15
Figure -21 KMeans Algorithm based Final Results in Graphical Representation	15
Figure -22 KMeans Clustering Alternative Way	16
Figure -23 Clustered Data Segmentation	16
Figure -24 Kmeans Centroid's Results	17
Figure -25 Datapoints with Two Clusters	17
Figure -26 KMeans Confusion Matrix	18



Figure -27 Results in Confusion Matrix.....	19
Figure -28 Pheatmap for Clustering	19
Figure -29 Compare Clustering	20
Figure -30 ARI for the three Clustering	21
Figure -31 ARI Index Results.....	21