# STAT2020 Assignment2

Paul Askie, c2104049

19/10/2020

## Activity 1 — Clustering Cancerous Tissue Samples [13 Marks]

```
library(foreign)
```

```
## Warning: package 'foreign' was built under R version 4.0.3
```

```
Golub_Data <- read.arff("golub-1999-v1_database.arff")
```

1. [0.5 Marks]: Set aside the rightmost column (containing the class labels) from the data, storing it separately from the remaining data frame (with the 1868 predictors).

```
Golab_Classe <- Golub_Data[, 1869]
Golub_Data_Predictors <- Golub_Data[, 1:1868]

summary(Golab_Classe)
```

```
##  1  2
## 47 25
```

```
dim(Golub_Data_Predictors)
```

```
## [1]   72 1868
```

2. [0.5 Marks]: Use the 72×1868 data frame to compute a matrix containing all the pairwise Euclidean distances between observations, i.e., a 72×72 matrix with distances between tissue samples according to their 1868 expression levels. This matrix must be of type dist, which can be achieved either by using function dist() from base R package stats or by type coercion using function as.dist().

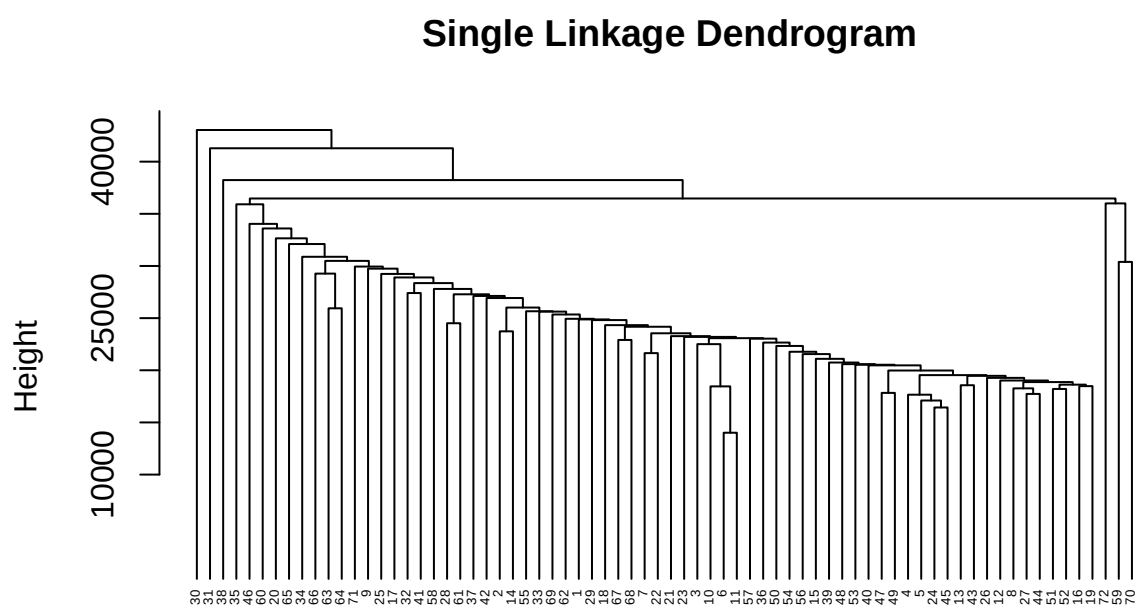
```
Golub_Data_Predictors_Dist <- dist(Golub_Data_Predictors, method = "euclidean")

str(Golub_Data_Predictors_Dist)
```

```
##  'dist' num [1:2556] 43921 41847 31997 33678 39833 ...
##  - attr(*, "Size")= int 72
##  - attr(*, "Diag")= logi FALSE
##  - attr(*, "Upper")= logi FALSE
##  - attr(*, "method")= chr "euclidean"
##  - attr(*, "call")= language dist(x = Golub_Data_Predictors, method = "euclidean")
```

```
# to view distance matrix in RStudio
Golub_Data_Predictors_Dist_Matrix <- as.matrix(Golub_Data_Predictors_Dist)
```

3. [1 Mark]: Use the distance matrix as input to call the Single-Linkage clustering algorithm available from base R package stats and plot the resulting dendrogram. Do Not use any class labels to perform this step (not even to display the dendrogram). Briefly comment on the dendrogram.

```
hc_single_Golab <- hclust(Golub_Data_Predictors_Dist, method = "single")
plot(hc_single_Golab,
     main=" Single Linkage Dendrogram",
     xlab="",
     sub ="",
     cex = 0.4,
     hang = -1)
```



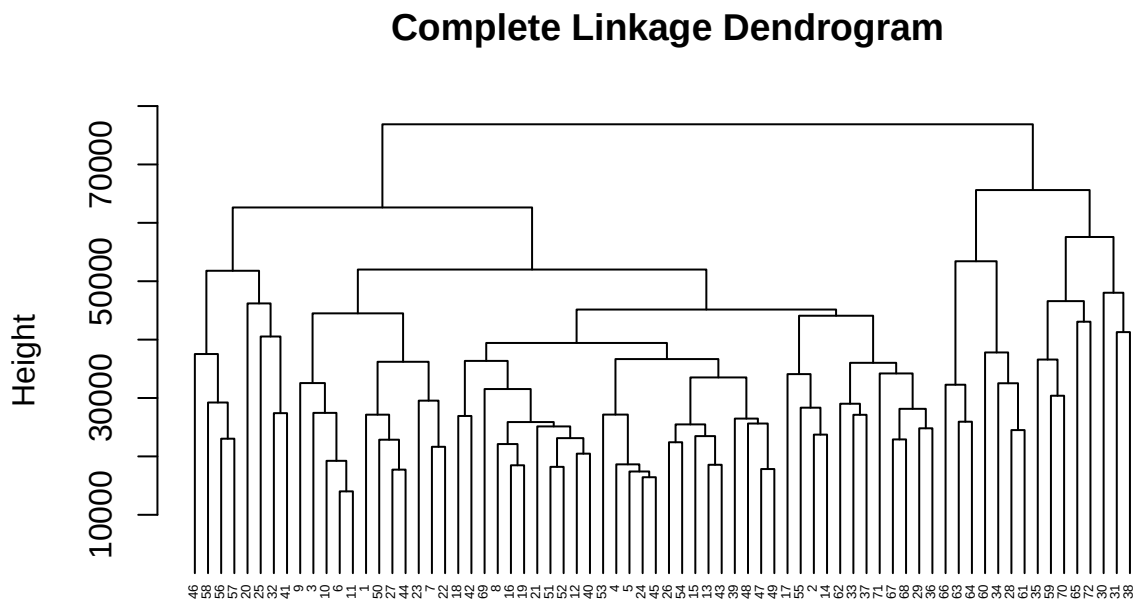

```
### removed, as need to reveal clusters in an unsupervised way

# # argument border = 1:2 colours are black and red so to better distinguish I have used
# # border = 2:3 which is red and green borders
#
# rect.hclust(hc_single_Golab,
#             k = 2,
#             border = 2:3)
#
```

```
# # Counting the number of group members where k=2
# hc_single_Golab_cut_2 <- cutree(hc_single_Golab, k=2)
# hc_single_Golab_cut_2_table <- table(hc_single_Golab_cut_2, dnn = "Cluster")
# kable(hc_single_Golab_cut_2_table, caption = "Single Linkage Cluster Frequency")
```

The observation 30 is the has the largest minimum Euclidean distance from the rest of the observations and as such is the most different from the rest of the observations according to the Single-Linkage algorithm. The clusters are hard to distinguish and cuts for values of k produce clusters with of similar values. An unlikely scenario.

4. `[1 Mark]`: Use the distance matrix as input to call the Complete-Linkage clustering algorithm available from base R package stats and plot the resulting dendrogram. Do Not use any class labels to perform this step (not even to display the dendrogram). Briefly comment on the dendrogram.

```
hc_complete_Golab <- hclust(Golub_Data_Predictors_Dist, method = "complete")

plot(hc_complete_Golab,
     main=" Complete Linkage Dendrogram",
     xlab="",
     sub ="",
     cex = 0.4,
     hang = -1)
```

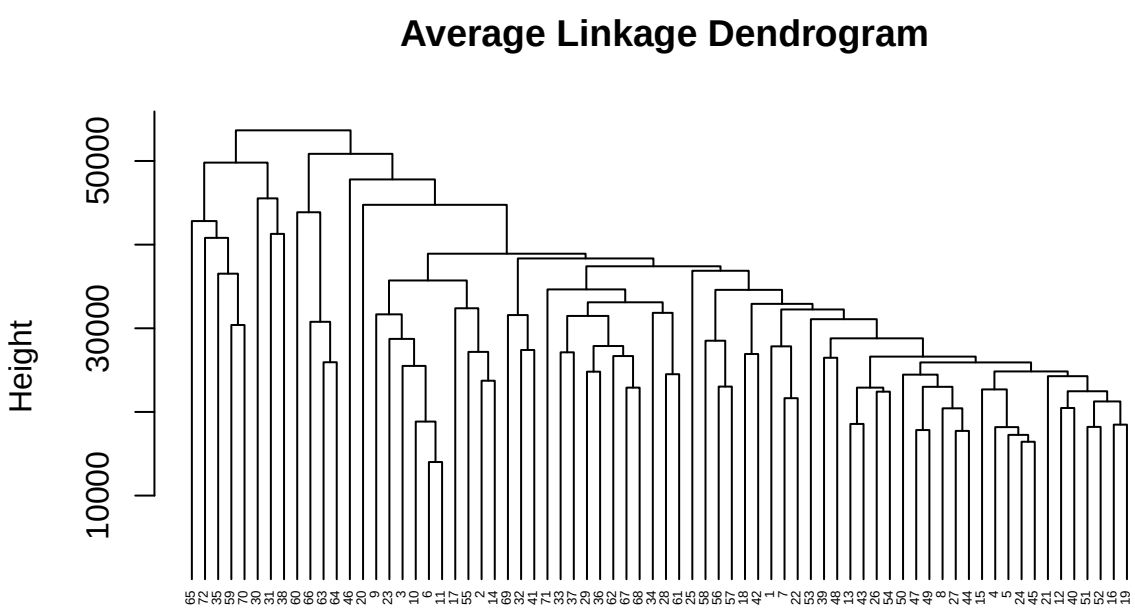


```
### removed, as need to reveal clusters in an unsupervised way

# rect.hclust(hc_complete_Golab,
#             k = 2,
#             border = 2:3)
#
# # Counting the number of group members where k=2
# hc_complete_Golab_cut_2 <- cutree(hc_complete_Golab, k=2)
#
# hc_complete_Golab_cut_2_table <- table(hc_complete_Golab_cut_2, dnn = "Cluster")
# kable(hc_complete_Golab_cut_2_table, caption = "Complete Linkage Cluster Frequency")
```

The Complete-Linkage alogrithm produces a dendrogram which produces a varying amount of cluster members at varying value of k. At a cut height of about 50000 the algorithm would produce 8 clusters but at a height of 70000 it would produce 2 clusters. The heights of the leaf node of clusters are more similar and hence a better representation of the clustering than the single linkage in this instance.

5. `[1 Mark]`: Use the distance matrix as input to call the Average-Linkage clustering algorithm available from base R package stats and plot the resulting dendrogram. Do Not use any class labels to perform this step (not even to display the dendrogram). Briefly comment on the dendrogram.

```
hc_average_Golab <- hclust(Golub_Data_Predictors_Dist, method = "average")
plot(hc_average_Golab,
     main=" Average Linkage Dendrogram",
     xlab="",
     sub ="",
     cex = 0.4,
     hang = -1)
```



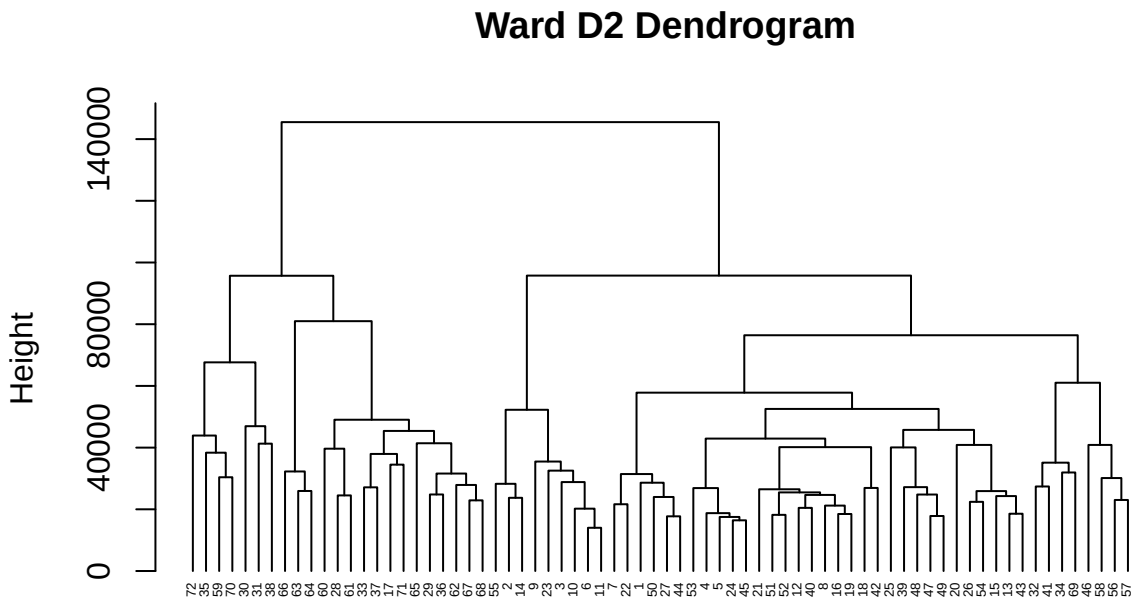

```
### removed, as need to reveal clusters in an unsupervised way

# rect.hclust(hc_average_Golab,
#             k = 2,
#             border = 2:3)
#
# # Counting the number of group members where k=2
# hc_average_Golab_cut_2 <- cutree(hc_average_Golab, k=2)
# hc_average_Golab_cut_2_table <- table(hc_average_Golab_cut_2, dnn = "Cluster")
# kable(hc_average_Golab_cut_2_table, caption = "Average Linkage Cluster Frequency")
```

The Average-Linkage algorithm dendrogram produces less defined clusters than the of the complete-linkage dendrogram, but the heights of the leaf node of clusters are more similar than the single linkage leaf node but less similar than that of the complete linkage algorithm. At heights below 40000 the number of clusters would increase substantially as the heights decreased.

6. `[1 Mark]`: Use the distance matrix as input to call Ward's clustering algorithm available from base R package stats and plot the resulting dendrogram. Do Not use any class labels to perform this step (not even to display the dendrogram). Briefly comment on the dendrogram. Note: Make sure you use variant ward.D2 of the Ward's algorithm as per our course notes and lab sheet.

```
hc_wardD2_Golab <- hclust(Golub_Data_Predictors_Dist, method = "ward.D2")
plot(hc_wardD2_Golab,
     main=" Ward D2 Dendrogram",
     xlab="",
```

```
     sub ="",
     cex = 0.4,
     hang = -1)
```




```
### removed, as need to reveal clusters in an unsupervised way

# rect.hclust(hc_wardD2_Golab,
#             k = 2,
#             border = 2:3)
#
# # Counting the number of group members where k=2
# hc_wardD2_Golab_cut_2 <- cutree(hc_wardD2_Golab, k=2)
#
# hc_WardD2_Golab_cut_2_table <- table(hc_wardD2_Golab_cut_2, dnn = "Cluster")
# kable(hc_WardD2_Golab_cut_2_table, caption = "Ward D2 Cluster Frequency")
```

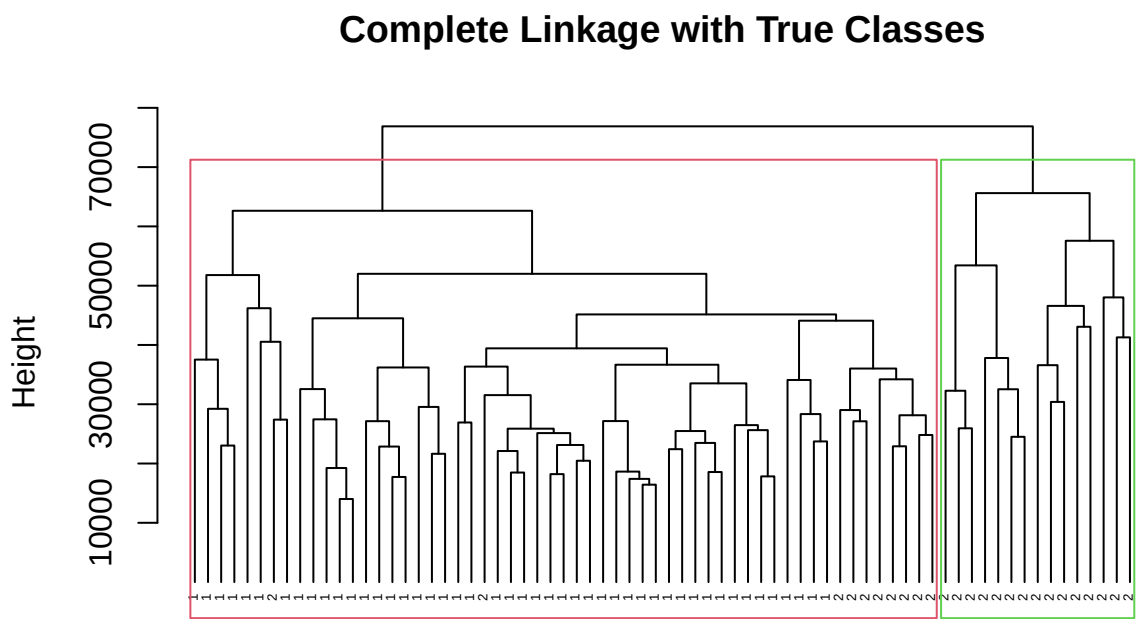
This dendrogram using the Ward D2 algorithm produces clearly defined areas for cuts to be made produce a range of clusters at different heights. It is visually the easiest to find cuts to produce a certain number of clusters at set heights.

7. `[2 Marks]`: Visually, the dendrograms suggest that some clustering algorithm(s) above generate more clear clusters than the others. In your opinion, what algorithm(s) may we be referring to and why? In particular, in which aspects the results produced by this/these algorithm(s) look more clear? Perform Item 8 below only for this/those algorithm(s).

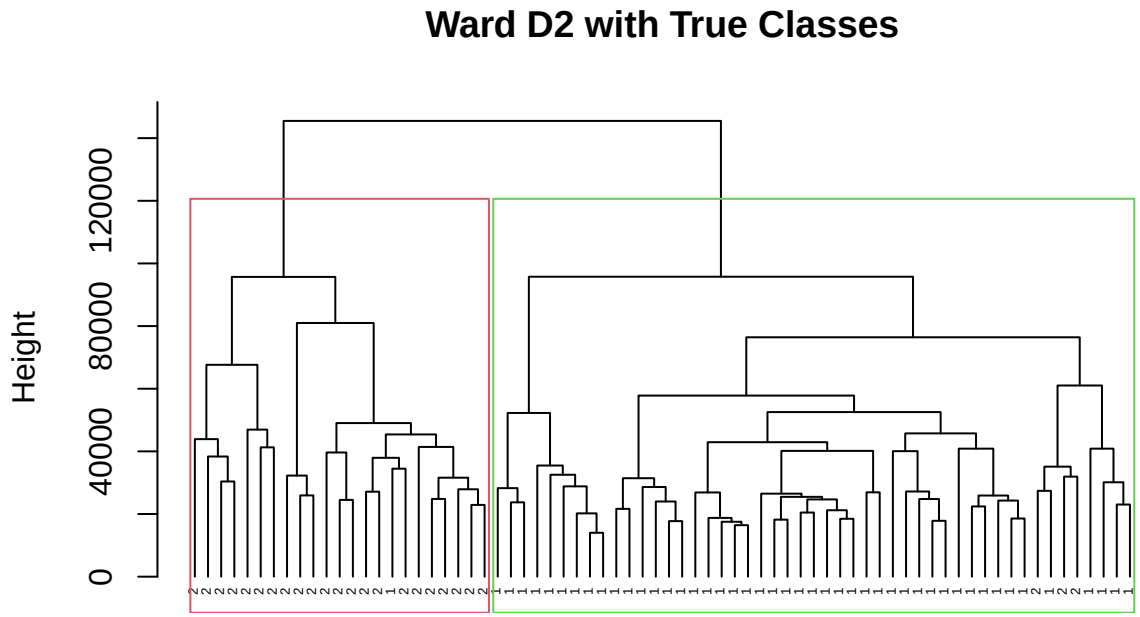
The heights between where the clusters split is also more clearly defined in the complete-linkage and Ward D2 algorithms making it easier to see sub-clusters and ascertain cut heights if required. The heights of the fusing within each cluster are of similar heights and as such easier to visualise where the clusters and cuts to form them might be.

8. [2 Marks]: Redraw the dendrogram(s) for the selected algorithm(s) in Item 7, now using the class labels that you stored separately in Item 1 to label the observations (as disposed along the horizontal axis of the dendrogram). Do some prominent clusters in the dendrogram(s) correspond approximately to the classes (i.e., two sub-types of leukemia)? Comment/explain. Hint: you can control the size of the labels using argument cex of function plot() (so they don't appear cluttered/superposed, e.g. cex = 0.6), and you may then need to zoom in to properly see the smaller labels in the figures.

```
# Complete Linkage for Question 8

hc_complete_Golab_TC <- hclust(Golub_Data_Predictors_Dist, method = "complete")
plot(hc_complete_Golab,
     main="Complete Linkage with True Classes",
     xlab="",
     sub ="",
     cex = 0.4,
     hang = -1,
     labels = Golab_Classe)

rect.hclust(hc_complete_Golab,
            k = 2,
            border = 2:3)
```



```r
# Ward D2 for Question 8

hc_wardD2_Golab_TC <- hclust(Golub_Data_Predictors_Dist, method = "ward.D2")
plot(hc_wardD2_Golab,
     main="Ward D2 with True Classes",
     xlab="",
     sub ="",
     cex = 0.4,
     hang = -1,
     labels = Golab_Classe)

rect.hclust(hc_wardD2_Golab,
            k = 2,
            border = 2:3)
```




Do some prominent clusters in the dendrogram(s) correspond approximately to the classes (i.e., two sub-types of leukemia)? Comment/explain. >In the complete linkage algorithm the Class labels the accuracy of the algorithm when compared to the class labels is around 86%. The precision of class 1, the ALL sub-type of leukemia, is low at 60% and as such is not likely to be a good predictor of the ALL sub-type of leukemia. Where the algorithm performs well is predicting the Class 2, the AML sub-type of leukemia. The algorithm may be able to be used to screen for only this sub-type as precision is 100%. This level of precision may also only apply to this group of observations and as such

9. The 1868 predictors have not been normalised before computing the distance matrix in Item 2. Normalisation is a non-trivial aspect in unsupervised clustering, as there is no ground truth to assess whether or not it improves performance. On the one hand, it may prevent variables with wider value ranges

from dominating distance computations, but on the other hand it may distort clusters by removing natural differences in variance that help characterize them as clusters. Normalisation thus becomes an aspect of Exploratory Data Analysis when it comes to clustering: the analyst will usually generate and try to interpret results both with normalised and non-normalised versions of the data. The type of normalisation depends on the application in hand. Here, we are computing Euclidean distance between rows of the data set, so the type of normalisation that applies is typically the so-called z-score normalisation of columns, where each column is re-scaled to have zero mean and standard deviation of 1. In this item, you are first asked to normalise the data this way before computing the distance matrix in Item 2 (hint: you can use function scale() to achieve this). Then:

a. [2 Marks]: Recompute the dendrograms in items 3 to 6. For each type of linkage algorithm, comparing the respective results with versus without normalisation, does normalisation improve the interpretability of the dendrogram (according to the same criteria/aspects considered in item 7)? Comment/explain. Note: Do NOT use any class label information to answer this item.

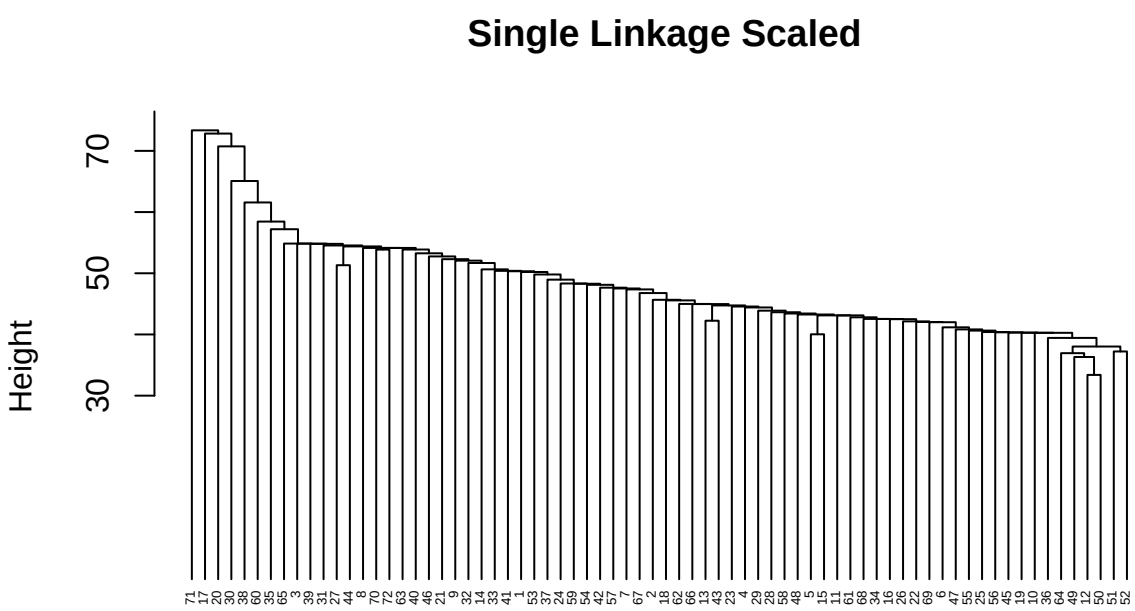
```
# used as.data.frame as scale formatted as a matrix
Golab_Data_Predictors_Scaled <- as.data.frame(scale(Golub_Data_Predictors))

# checking a random variable for correctness
round(mean(Golab_Data_Predictors_Scaled$"199"),0)
```

```
## [1] 0
```

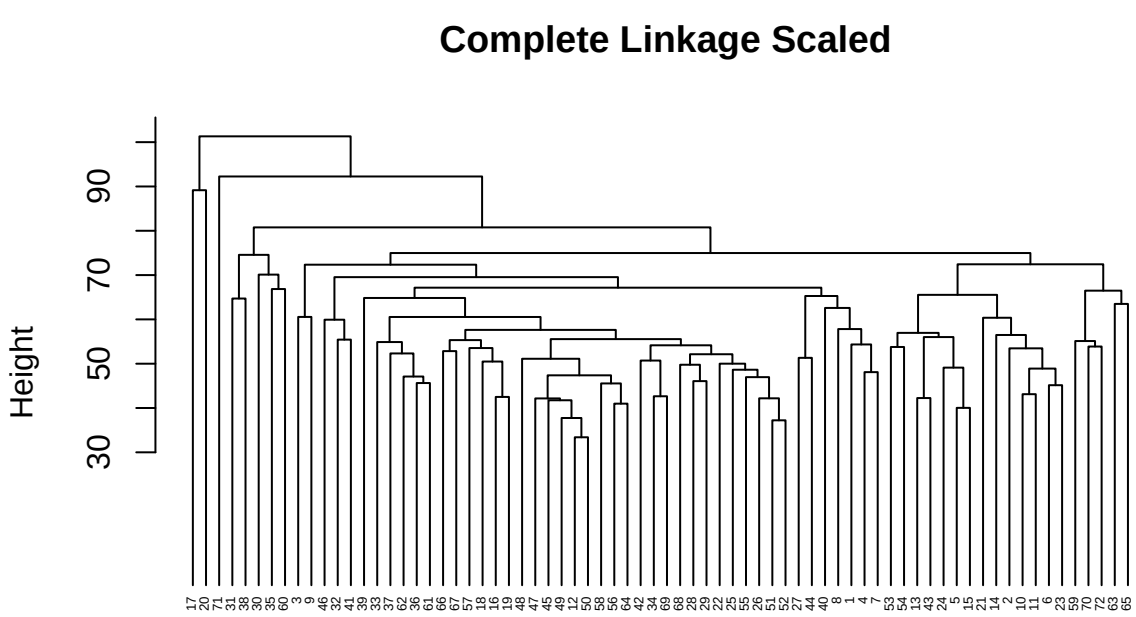
```
sd(Golab_Data_Predictors_Scaled$"199")
```

```
## [1] 1
```

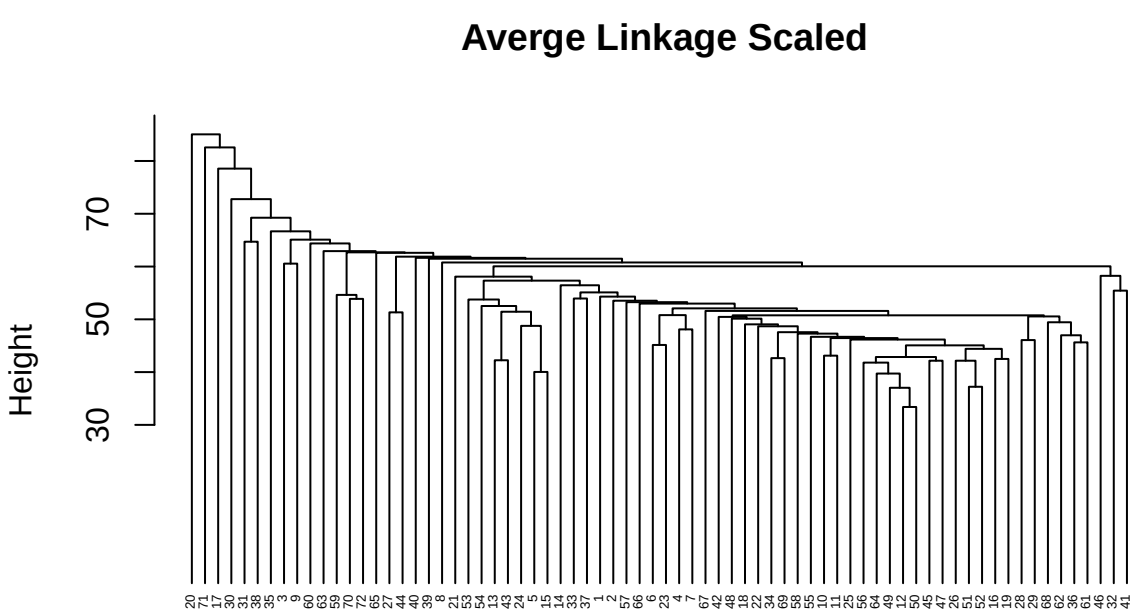
```
Golub_Data_Predictors_Dist_Scaled <- dist(Golab_Data_Predictors_Scaled, method = "euclidean")

hc_single_Golab_Scaled <- hclust(Golub_Data_Predictors_Dist_Scaled, method = "single")
plot(hc_single_Golab_Scaled,
     main=" Single Linkage Scaled",
     xlab="",
     sub ="",
     cex = 0.4,
     hang = -1)
```



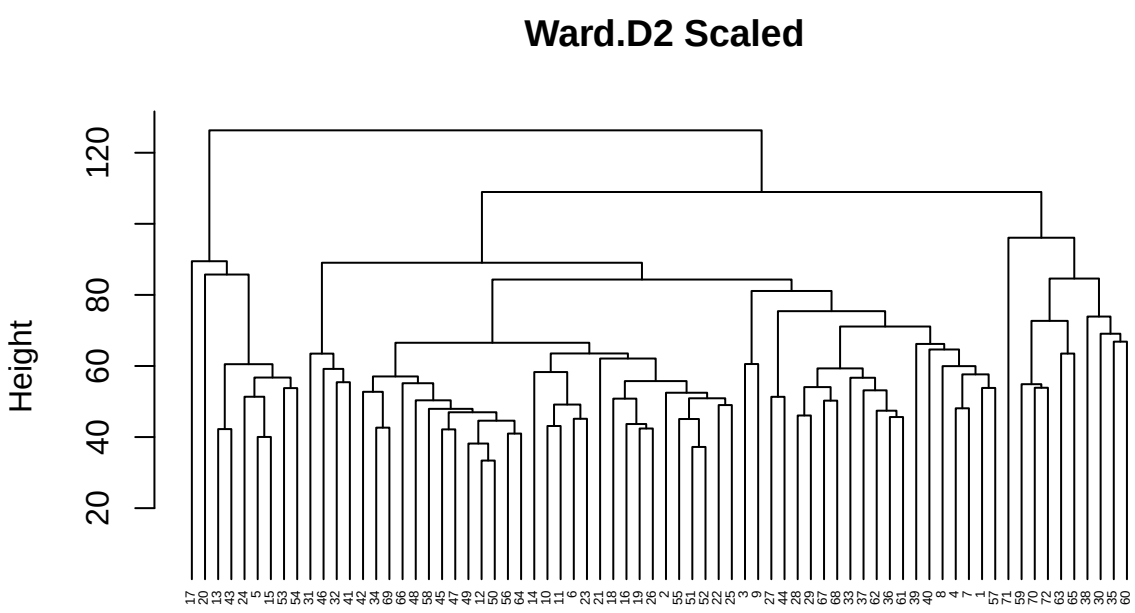

```
hc_complete_Golab_Scaled <- hclust(Golub_Data_Predictors_Dist_Scaled, method = "complete")
plot(hc_complete_Golab_Scaled,
     main=" Complete Linkage Scaled",
     xlab="",
     sub ="",
     cex = 0.4,
     hang = -1)
```




```
hc_average_Golab_Scaled <- hclust(Golub_Data_Predictors_Dist_Scaled, method = "average")
plot(hc_average_Golab_Scaled,
     main=" Averge Linkage Scaled",
     xlab="",
     sub ="",
     cex = 0.4,
     hang = -1)
```




```r
hc_wardD2_Golab_Scaled <- hclust(Golub_Data_Predictors_Dist_Scaled, method = "ward.D2")
plot(hc_wardD2_Golab_Scaled,
     main=" Ward.D2 Scaled",
     xlab="",
     sub ="",
     cex = 0.4,
     hang = -1)
```




Normalisation of the data does not improve the results for any of the dendrograms. The dendrograms for the single complete and average linkage algorithms are skewed to the left side and form small clusters for k values upto and exceeding 10.

b. [2 Marks]: Your analysis in item 9.a must have been purely based on the visual interpretation of the dendrograms, i.e., in a completely unsupervised way (using no labels). Now, you will use the class labels to confirm whether or not normalisation has helped the clustering algorithms, at least when it comes to a partition extracted via a cut through the dendrogram and with the number of clusters equal to the number of known classes (i.e., two). Specifically, perform a cut to extract a partition with k = 2 clusters from each of the four dendrograms produced in item 9.a, then for each of them compute a contingency table (confusion matrix) of the true class labels by the assigned cluster label, and discuss the correspondence (or lack thereof) between classes and clusters resulting from each linkage algorithm based on the tables. Note: Keep in mind that any permutation of labels in the clustering result does not change the clusters and the clustering result itself, so you should not expect to necessarily see an association between each class label and the very same label in the clustering result.

```
hc_single_Golab_Scaled_TC_cut_2 <- cutree(hc_single_Golab_Scaled, k=2)
(hc_single_Golab_Scaled_confusion_matrix <- table(hc_single_Golab_Scaled_TC_cut_2,
                                                  Golab_Classe,
                                                  dnn = c("Predicted Class", "Actual Class")))
```

```
##                Actual Class
## Predicted Class  1  2
##               1 47 24
##               2  0  1
```

```
hc_complete_Golab_Scaled_TC_cut_2 <- cutree(hc_complete_Golab_Scaled, k=2)
(hc_complete_Golab_Scaled_confusion_matrix <- table(hc_complete_Golab_Scaled_TC_cut_2,
                                                  Golab_Classe,
                                                  dnn = c("Predicted Class", "Actual Class")))
```

```
##                Actual Class
## Predicted Class  1  2
##               1 45 25
##               2  2  0
```

```
hc_average_Golab_Scaled_TC_cut_2 <- cutree(hc_average_Golab_Scaled, k=2)
(hc_average_Golab_Scaled_confusion_matrix <- table(hc_average_Golab_Scaled_TC_cut_2,
                                                  Golab_Classe,
                                                  dnn = c("Predicted Class", "Actual Class")))
```

```
##                Actual Class
## Predicted Class  1  2
##               1 46 25
##               2  1  0
```

```
hc_wardD2_Golab_Scaled_TC_cut_2 <- cutree(hc_wardD2_Golab_Scaled, k=2)
(hc_wardD2_Golab_Scaled_confusion_matrix <- table(hc_wardD2_Golab_Scaled_TC_cut_2,
                                                  Golab_Classe,
                                                  dnn = c("Predicted Class", "Actual Class")))
```

```
##                Actual Class
## Predicted Class  1  2
##               1 38 25
##               2  9  0
```

For single, complete and average linkage algorithms when scaled and cut at k=2 the majority of observations just go into class 1 as the heights of the leaf nodes are very similar. The Ward D2 algortihm leaf nodes heights varied slightly but did not identify any of the negative class (Class 2, AML) correctly. As such scaling the dataset leads to a poorer outcome when compared to the non-scaled dataset.

## Activity 2 — Clustering Genes [7 Marks]

10. [0.5 Marks]: Read the data set directly from the arff file into a data frame. Then, set aside the rightmost column (containing the class labels) from the data, storing it separately from the remaining data frame (containing the 20 predictors).

```
Yeast_Data <- read.arff("yeast.arff")
Yeast_Classe <- Yeast_Data[ ,21]
Yeast_Data_Predictors <- Yeast_Data[, 1:20]

dim(Yeast_Classe)
```

```
## NULL
```

```
dim(Yeast_Data_Predictors)
```

```
## [1] 205  20
```

11. [1 Mark]: Use the 205×20 data frame to compute a matrix containing all the pairwise correlation based dissimilarities between observations, i.e., a 205 × 205 matrix with dissimilarities between genes according to their 20 expression measurements. Important Note: Co-regulated genes are better characterized by similar trends (shapes) in their gene expression profiles, rather than similar expression levels in terms of their absolute values (magnitudes). In other words, the similarity between genes in terms of their expression profiles for different measurements is better captured by a correlation measure, such as Pearson (standard) correlation (James et al. 2013). In this activity we will use correlation instead of Euclidean distance. However, recall that correlation is a similarity measure that ranges from −1 (lowest possible similarity) to +1 (highest possible similarity). After computing the 205×205 similarity matrix, you have to transform it into a dissimilarity matrix whose values are within 0 (lowest possible dissimilarity) and +1 (highest possible dissimilarity). Once you have this correlation-based dissimilarity (i.e. distance) matrix, you can coerce it into type dist using function as.dist() (as required by the hierarchical clustering methods from package stats in base R).

    Hints: (a) See Section 10.3.2 of (James et al. 2013) for more information about correlation-based distance, and Section 10.5.2 for an example in R (in which the distance matrix is computed within [0, +2] rather than [0, +1] though); (b) make sure you do not confuse correlation between observations (rows) with correlation between measurements (columns — the default when calling e.g. function cor() in R); (c) if your distance matrix has been computed correctly, the distance between the first two observations should be 0.05640425, whereas the distance between the first and last observation should be 0.7821619 (confirm this before you apply as.dist() and proceed with your assignment!).

```
# t function transposes rows into columns
pearson_correlation <- cor(t(Yeast_Data_Predictors), method = "pearson")

# correlation coefficient (r) to a dissimilarity measure (d) transformation is d = (1-r)/2
pearson_correlation_distance <- as.dist((1-pearson_correlation)/2)

# to check if the observations are correct
pearson_correlation_distance_matrix <- as.matrix(pearson_correlation_distance)
pearson_correlation_distance_matrix[1,2]
```

```
## [1] 0.05640425
```

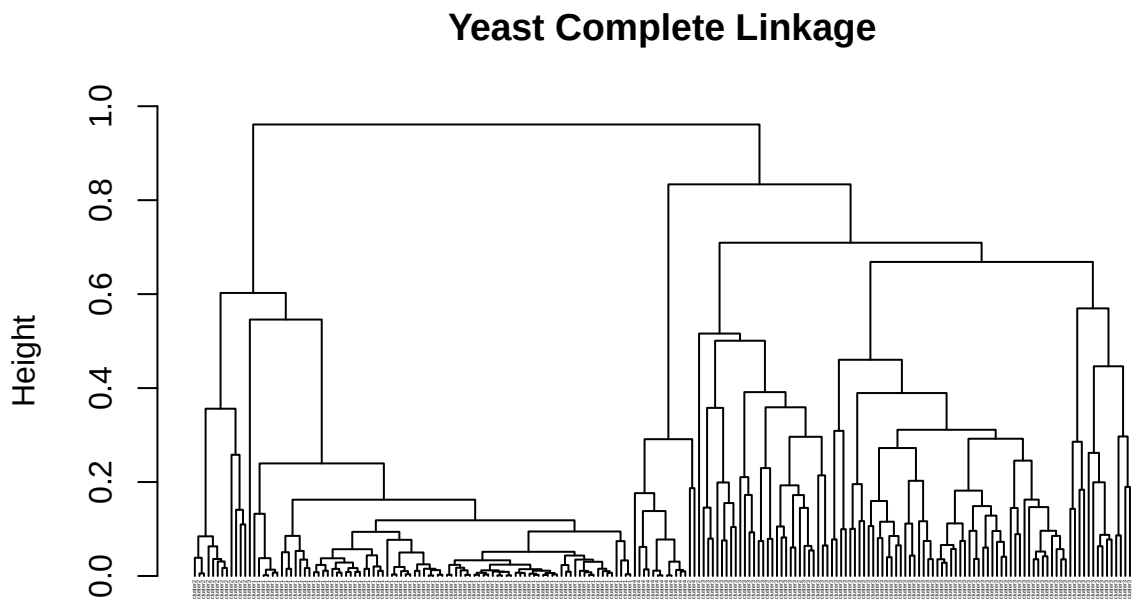
```
pearson_correlation_distance_matrix[1,205]
```

```
## [1] 0.7821619
```

```
reference1 <- "https://www.itl.nist.gov/div898/software/dataplot/refman2/auxillar/pear_dis.htm"
```

12. [1 Mark]: Provide the distance matrix as input to call the Complete-Linkage clustering algorithm available from base R package stats and plot the resulting dendrogram. Use the class labels that you stored separately in Item 10 to label the observations along the horizontal axis of the dendrogram (but DO NOT use the labels to carry out the clustering itself!). Do some prominent clusters in the dendrogram correspond approximately to the classes (i.e., the four known categories of genes)? Comment/explain. Hint: you can control the size of the labels using argument cex of function plot() (so they don't appear cluttered/superposed, e.g. cex = 0.4), and you may then need to zoom in to properly see the smaller labels in the figures.

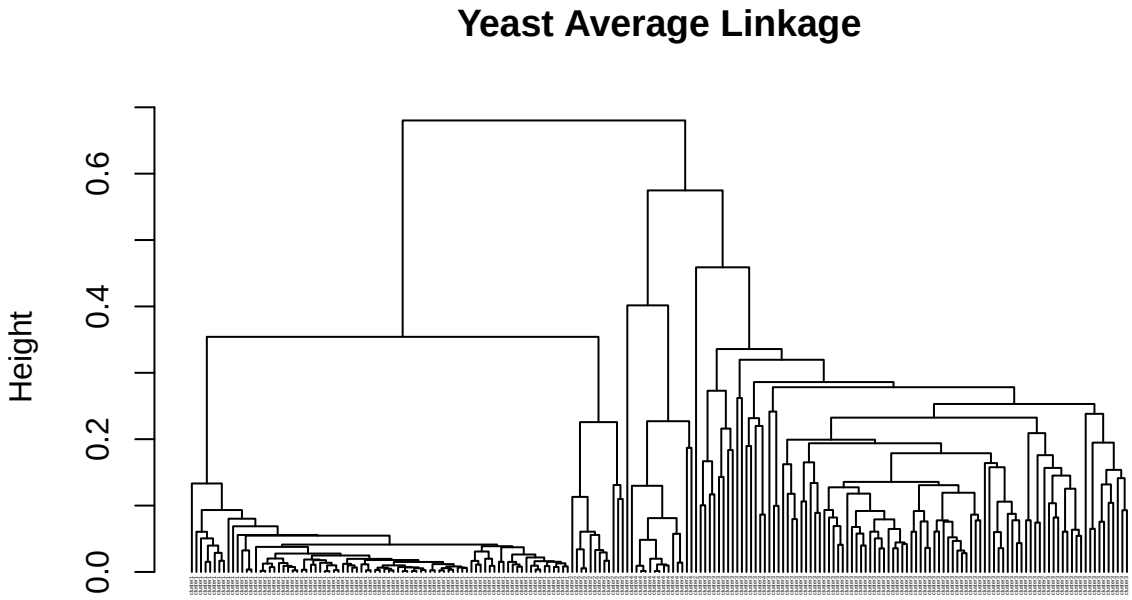
```
hc_complete_Yeast <- hclust(pearson_correlation_distance, method = "complete")
plot(hc_complete_Yeast,
     main="Yeast Complete Linkage",
     xlab="",
     sub ="",
     cex = 0.2,
     hang = -1,
     labels = Yeast_Classe)
```



Clusters have formed along the horizonal axis of the dendrogram, this dendrogram could be split into 3 main cluster with one containing the majority of clusters 1 and 2 with 3 and 4 seperately clustered. There is no cut that would seperate them into the 4 known categories of genes

13. [1 Mark]: Repeat 12, now with Average-Linkage.

```
hc_average_Yeast <- hclust(pearson_correlation_distance, method = "average")
plot(hc_average_Yeast,
     main="Yeast Average Linkage",
     xlab="",
     sub ="",
     cex = 0.2,
     hang = -1,
     labels = Yeast_Classe)
```

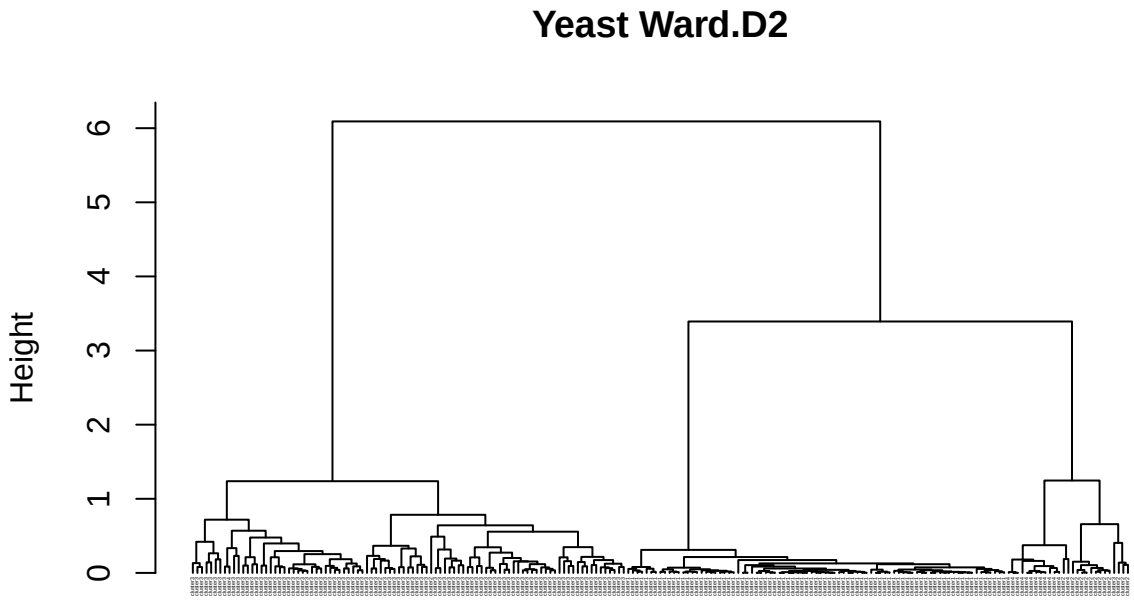



If split at k=6, cluster 1 and 2 would be seperated into their own groups and 3 and 4 would have 2 clusters with the majority of members being assigned to those clusters.

14. [1 Mark]: Repeat 12, now with Ward's. Note: Make sure you use variant ward.D2 of the Ward's algorithm as per our course notes and lab sheet.

```
hc_wardD2_Yeast <- hclust(pearson_correlation_distance, method = "ward.D2")
plot(hc_wardD2_Yeast,
     main="Yeast Ward.D2",
```

```
     xlab="",
     sub ="",
     cex = 0.2,
     hang = -1,
     labels = Yeast_Classe)
```

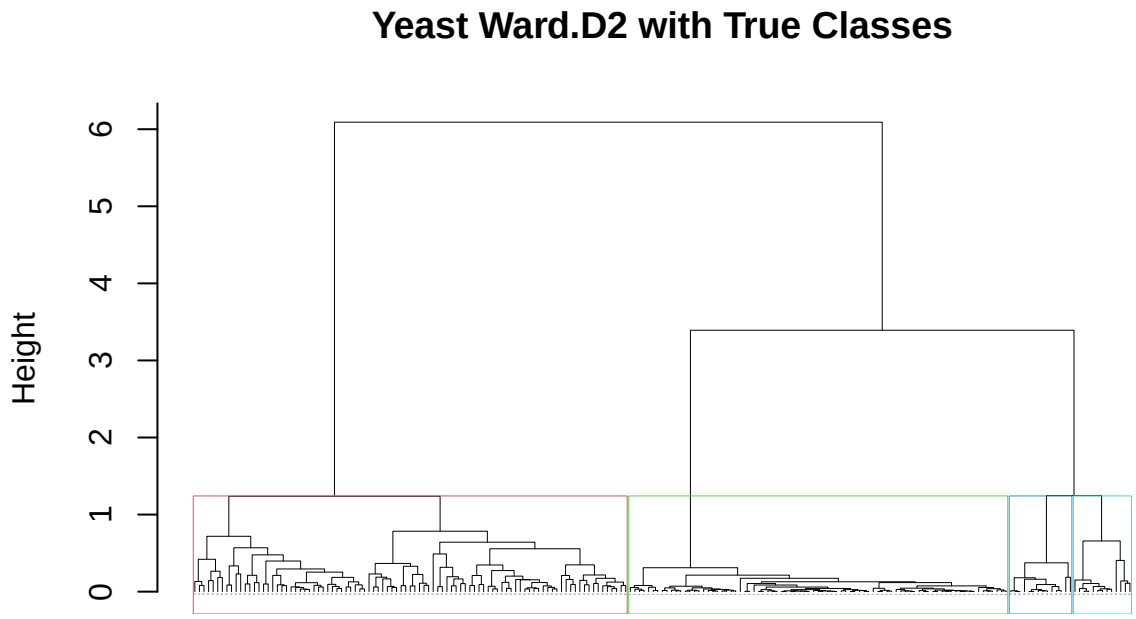



>A cut can be made at k=4 which will seperate the 4 clusters where the overwhelming majority is of the same type of Yeast.

15. [2.5 Marks]: Perform a cut through the dendrogram to extract a partition with k = 4 clusters from the clustering hierarchy produced by the Ward's method in item 14, then compute a contingency table (confusion matrix) of the true class labels by the assigned cluster label, and explain in detail the correspondence/association (or lack thereof) between classes (known categories) and discovered/extracted clusters revealed by the table.

```
pars <- par() # default parameters to reset to

# sets all line weights to 0.1
par(lwd=0.1)

levels(Yeast_Classe) <- c("1", "2", "3", "4")

wardD2_Yeast_Cut_4_Classes <- as.factor(cutree(hc_wardD2_Yeast, k=4))

hc_wardD2_Yeast_True_Classes <- hclust(pearson_correlation_distance, method = "ward.D2")
```

```
plot(hc_wardD2_Yeast_True_Classes,
     main="Yeast Ward.D2 with True Classes",
     xlab="",
     sub ="",
     cex = 0.1,
     hang = -1,
     labels = Yeast_Classe)

rect.hclust(hc_wardD2_Yeast_True_Classes,
            k = 4,
            border = 2:5)
```




```
(confusion_matrix <- table(wardD2_Yeast_Cut_4_Classes,
                           Yeast_Classe,
                           dnn = c("Predicted Class", "Actual Class")))
```

```
##                 Actual Class
## Predicted Class  1  2  3  4
##               1 83  0  0  0
##               2  0 12  1  0
##               3  0  2 92  1
##               4  0  1  0 13
```

```r
diag_CM <- diag(confusion_matrix)

row_Sums = apply(confusion_matrix, 1, sum) # number of instances per class

col_Sums = apply(confusion_matrix, 2, sum) # number of predictions per class

(accuracy <- sum(diag_CM)/sum(confusion_matrix))
```

```
## [1] 0.9756098
```

```r
precision = diag_CM / col_Sums

recall = diag_CM / row_Sums

PC_table <- data.frame(Cluster = 1:4,
                       Precision = round(precision, 2),
                       Recall = round(recall, 2))

knitr::kable(PC_table)
```

| Cluster | Precision | Recall |
|---|---|---|
| 1 | 1.00 | 1.00 |
| 2 | 0.80 | 0.92 |
| 3 | 0.99 | 0.97 |
| 4 | 0.93 | 0.93 |

```r
par(lwd=pars$lwd) # reset parameter to default
```

The accuracy of the Ward D2 algorithm on the yeast dataset is around 97.5%. With precision and recall all above 90% with the exception of Cluster 2's precision which is at 80%. Therefore the Ward D2 would have a strong probability of predicting the type of yeast from a new sample using the same methodology.