FACHHOCHSCHULE
WIENER NEUSTADT
University of Applied Sciences – Austria

Cluster Analysis(HÜ2)

Multivariate Statistic

Bernhard Spangl, Dipl.-Ing. Dr.

Students name:

Marjan Aziminezhad

Solveig-Stella Billes

Date: Mai 2023

Task description for the 2nd homework assignment on 'Cluster Analysis'.

1. Load the dataset 'promoters.data' into the R workspace. There should be 106 observations.
2. Clean the dataset if necessary.
3. Choose a suitable distance measure for this dataset and program a function that calculates the chosen distance for any two words.
4. Compute a dissimilarity matrix for all observed DNA sequences (words) in the dataset.
5. Use the computed dissimilarity matrix to cluster the individual genes using an appropriate clustering algorithm. Reflect on why the k-means algorithm is not suitable for nominal/ordinal scaled variables.
6. Validate your results with the existing assignment already present in the dataset.
7. Choose another distance measure (or multiple measures) (you can also use a pre-programmed distance measure available in R) and repeat steps 4 to 6. Were you able to improve your results?
8. Submit your runnable R script (as a text file with the extension .R) and a comprehensible report (as a PDF file) that includes R code, R output, and the generated graphics via Moodle by June 4th, 2023.

Please consider the explanations as comments in code.

1. Hamming distance

```
#Loading Librarys
library(readr)
```

```
## Warning: Paket 'readr' wurde unter R Version 4.2.2 erstellt
```

```
library(cluster)
library(rmarkdown)
```

```
## Warning: Paket 'rmarkdown' wurde unter R Version 4.2.2 erstellt
```

```
library(stringdist)
```

```
## Warning: Paket 'stringdist' wurde unter R Version 4.2.2 erstellt
```

```
library(factoextra)
```

```
## Warning: Paket 'factoextra' wurde unter R Version 4.2.3 erstellt
## Lade nötiges Paket: ggplot2
## Warning: Paket 'ggplot2' wurde unter R Version 4.2.2 erstellt
## Welcome! Want to learn more? See two factoextra-related books at https:/
/goo.gl/ve3WBa
```

```
# Step 1: Load the dataset 'promoters.data' into the R workspace.
promoters <- read_csv("promoters.data", col_names = FALSE)
```

```
## Rows: 106 Columns: 3
## ── Column specification ──────────────────────────────────────────────────
────────
## Delimiter: ","
## chr (3): X1, X2, X3
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
```

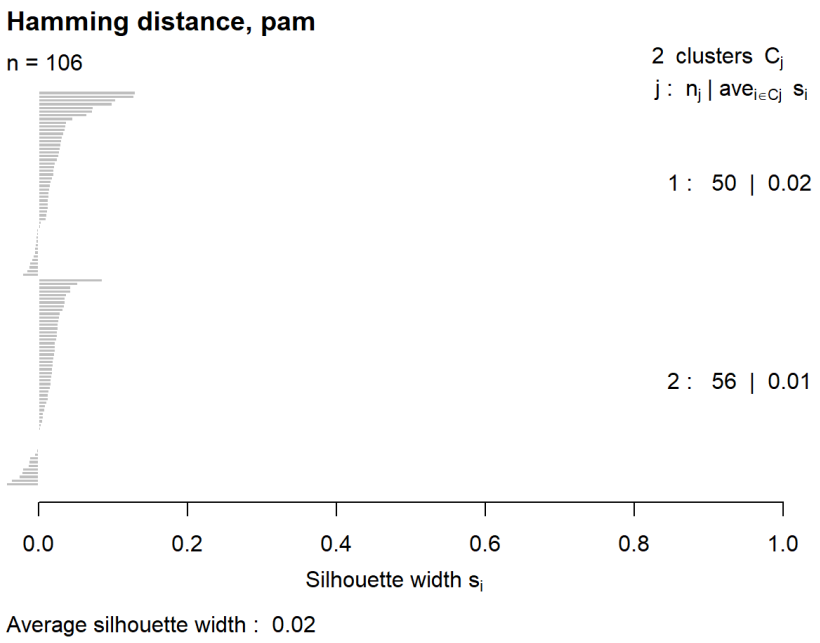
```
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
# Step 2: Clean the dataset if necessary.


# cleaning not necessary because there are no missing data.

'The Hamming distance is commonly used in bioinformatics for comparing DNA
sequences because it provides a simple count of the number of substitutions
required to transform one sequence into another. It is particularly useful
when the sequences being compared are of the same length and have no insert
ions or deletions'

# Step 3: Define a function to calculate the Manhattan distance for two wor
ds.

calculate_hamming_distance <- function(str1, str2) {

  distance <- stringdist::stringdistmatrix(c(str1), c(str2), method = "hamm
ing")[1]

  return(distance)

}

'"osa": Optimal String Alignment. It calculates the minimum number of inser
tions, deletions, and substitutions required to transform one string into a
nother.

"lv": Levenshtein distance. It calculates the minimum number of single-char
acter edits (insertions, deletions, and substitutions) needed to transform
one string into another.

"dl": Damerau-Levenshtein distance. It is similar to Levenshtein distance b
ut also allows for transpositions of adjacent characters.

"hamming": Hamming distance. It measures the number of positions at which t
wo strings of equal length differ.

"lcs": Longest Common Subsequence. It measures the length of the longest su
bsequence shared by two strings.

"qgram": Q-gram distance. It calculates the dissimilarity between two strin
gs based on the number of shared q-grams (substrings of length q).

"cosine": Cosine distance. It calculates the dissimilarity between two stri
ngs based on the cosine similarity of their vector representations.

"jaccard": Jaccard distance. It measures the dissimilarity between two stri
ngs based on the size of their shared n-grams (substrings of length n) rela
tive to the size of their union.

"jw": Jaro-Winkler distance. It measures the dissimilarity between two stri
ngs based on the number of matching characters and the transpositions of ad
jacent characters.

"soundex": Soundex distance. It calculates the dissimilarity between two st
rings based on the Soundex encoding, which represents the phonetic pronunci
ation of words.'

# Step 4: Compute the dissimilarity matrix using Hamming distance for all o
bserved DNA sequences.

sequences <- c(promoters$X3)

hamming_distance_matrix <- stringdist::stringdistmatrix(sequences, sequence
s, method = "hamming")
```

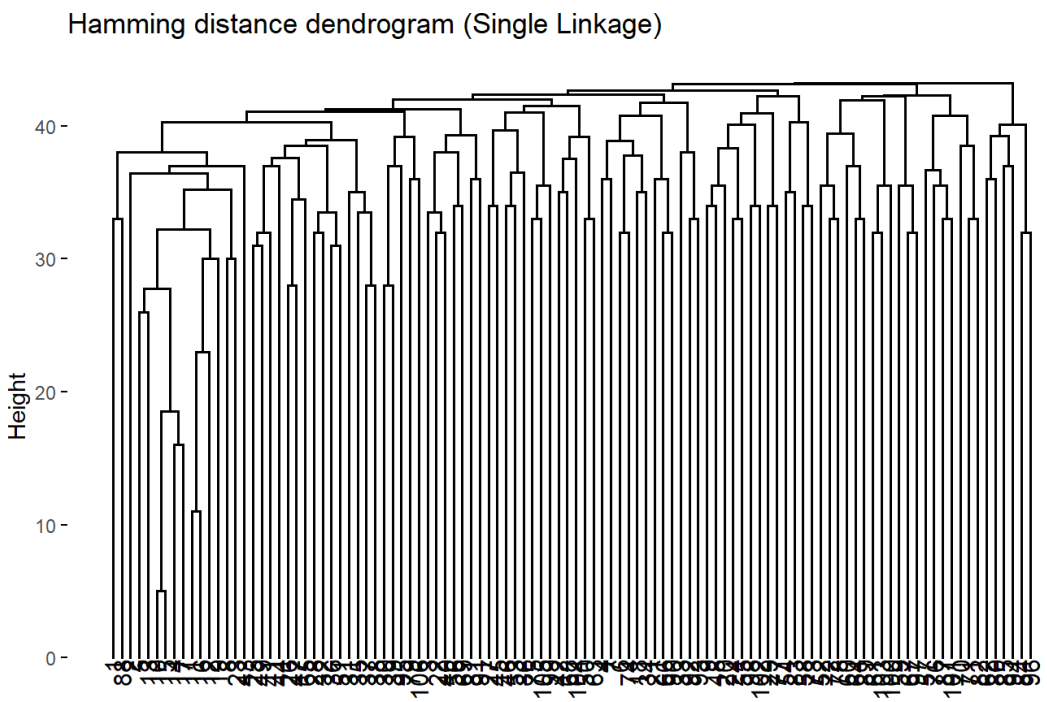
```r
# Convert the distance matrix to a dissimilarity object
hamming_dissimilarity <- as.dist(hamming_distance_matrix)


# Step 5: Cluster the genes using the Hamming distance using different Link
age methods.
hamming_clusters_single <- agnes(hamming_dissimilarity, method = "single",
diss = TRUE)
hamming_clusters_complete <- agnes(hamming_dissimilarity, method = "complet
e", diss = TRUE)
hamming_clusters_average <- agnes(hamming_dissimilarity, method = "average"
, diss = TRUE)


# Step 6: Plot the Hamming distance clusters using different linkage method
s.
plot_dendrogram_Single <- fviz_dend(hamming_clusters_average, main = "Hammi
ng distance dendrogram (Single Linkage)")
```

```
## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "no
ne" instead as
## of ggplot2 3.3.4.
## ℹ The deprecated feature was likely used in the factoextra package.
##   Please report the issue at < ]8;;https://github.com/kassambara/factoex
tra/issues https://github.com/kassambara/factoextra/issues ]8;; >.
```

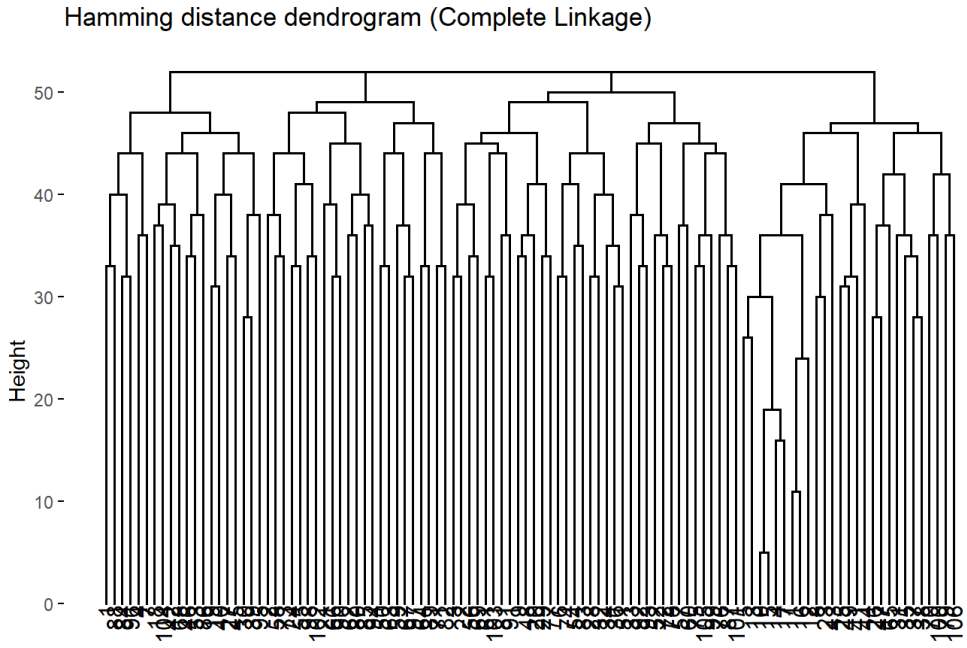
```r
plot_dendrogram_complete <- fviz_dend(hamming_clusters_complete, main = "Ha
mming distance dendrogram (Complete Linkage)")
plot_dendrogram_average <- fviz_dend(hamming_clusters_average, main = "Hamm
ing distance dendrogram (Average Linkage)")
plot(hamming_clusters_pam, main="Hamming distance, pam ")
```



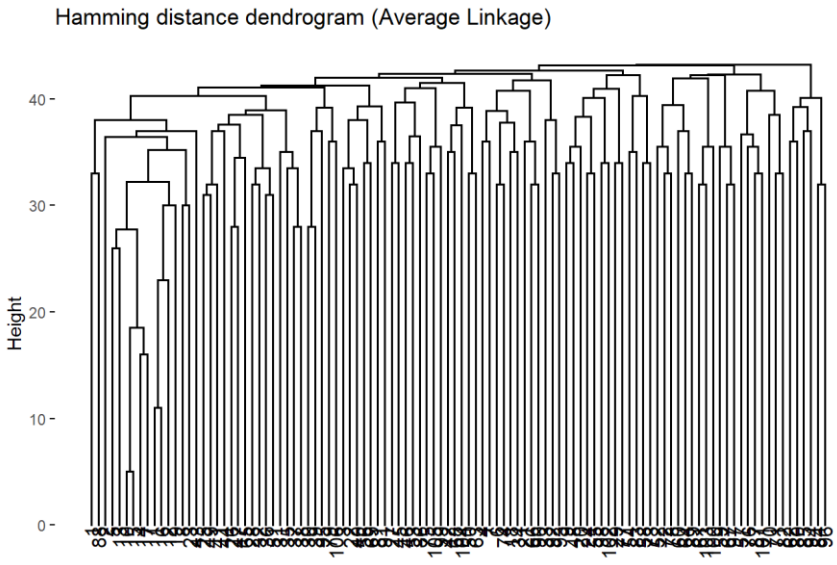
```
# Display the dendrogram plots
print(plot_dendrogram_Single)
```




```
print(plot_dendrogram_complete)
```



```
print(plot_dendrogram_average)
```




```
# Optional: Cross-tabulate the clustering results with the X1 column in the promoters data
table(cutree(hamming_clusters_single, 2), promoters$X1)
##
##      -  +
##   1 51 53
##   2  2  0
table(cutree(hamming_clusters_complete, 2), promoters$X1)
##
##      -  +
##   1  7 13
##   2 46 40
table(cutree(hamming_clusters_average, 2), promoters$X1)
##
##      -  +
##   1 47 53
##   2  6  0
table(hamming_clusters_pam$clustering, promoters$X1)
##
##      -  +
##   1 26 24
##   2 27 29
```

Based on the performance of different clustering algorithms on the given dataset:

Single Agglomerative Coefficient: 0.13

Complete Agglomerative Coefficient: 0.38

Average Agglomerative Coefficient: 0.26

PAM (Partitioning Around Medoids) Average silhouette width: 0.02

A higher value indicates better clustering results, with values closer to 1 representing well-separated clusters.

Based on the results, none of the agglomerative clustering methods (single, complete, and average linkage) provide a correct separation of the data. This suggests that these methods may not be suitable for this dataset or the chosen distance metric.

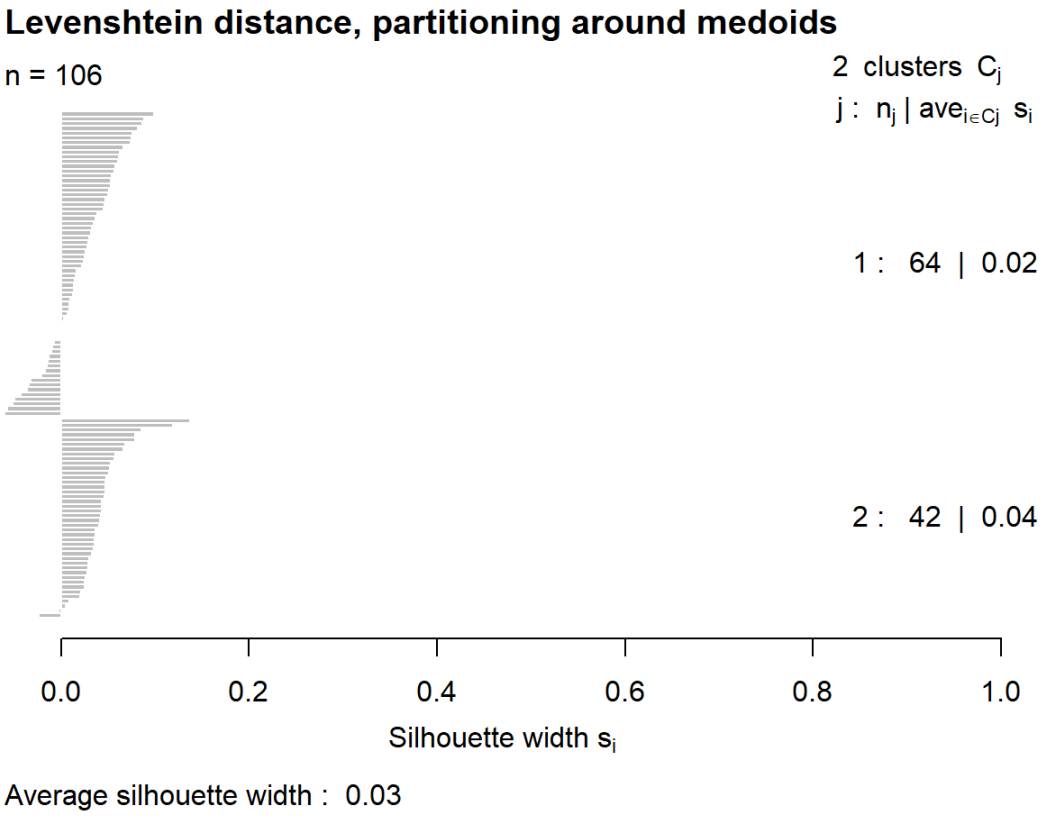
The statement also mentions that using the k-means algorithm is inappropriate for data on an ordinal/nominal scale. This is because k-means assumes continuous variables and calculates means to assign cluster centers, which is not meaningful for ordinal/nominal data. Therefore, the algorithm would not be able to assign meaningful centers to the clusters.

Additionally, the alternative approach of using the Partitioning Around Medoids (PAM) algorithm did not work well in this case, as indicated by the low average silhouette width of 0.02. This suggests that PAM did not produce well-defined clusters.

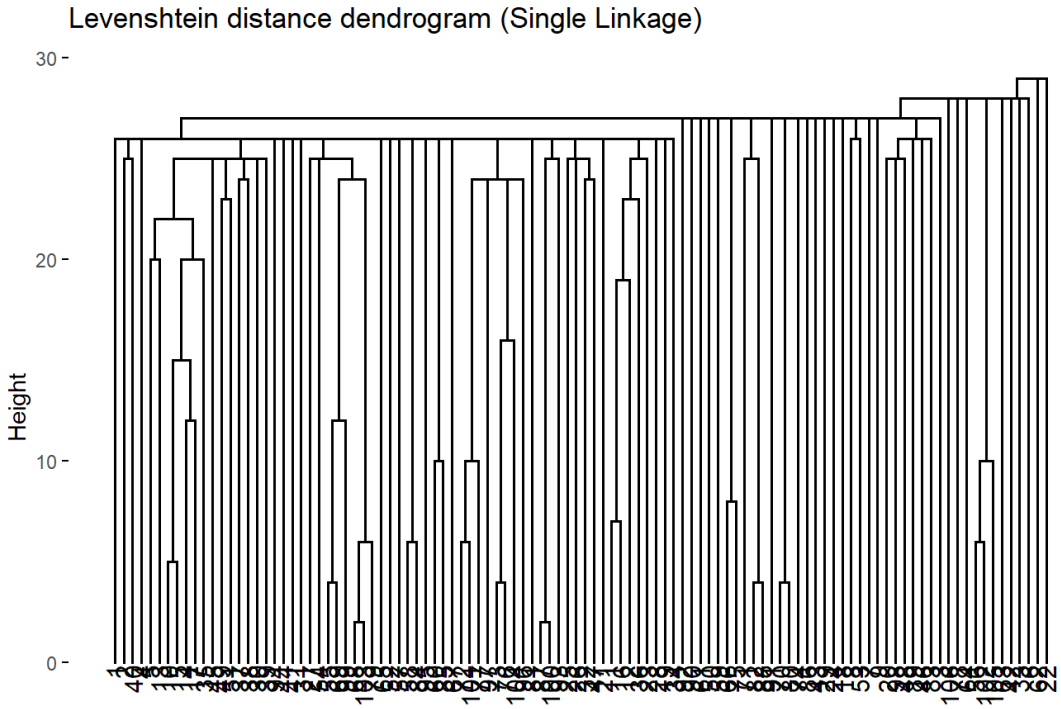
2. Levenshtein distance

```
#second Method:


# Step 3: Define a function to calculate the Levenshtein distance for two D
NA sequences.

calculate_levenshtein_distance <- function(seq1, seq2) {

  distance <- stringdist::stringdistmatrix(c(seq1), c(seq2), method = "lv")
[1]

  return(distance)

}


# Step 4: Compute the dissimilarity matrix using Levenshtein distance for a
ll observed DNA sequences.

sequences <- c(promoters$X3)

levenshtein_distance_matrix <- stringdist::stringdistmatrix(sequences, sequ
ences, method = "lv")


# Step 5: Cluster the genes using the Levenshtein distance.

levenshtein_clusters_single <- agnes(levenshtein_distance_matrix, method =
"single", diss = TRUE)

levenshtein_clusters_complete <- agnes(levenshtein_distance_matrix, method
= "complete", diss = TRUE)
```

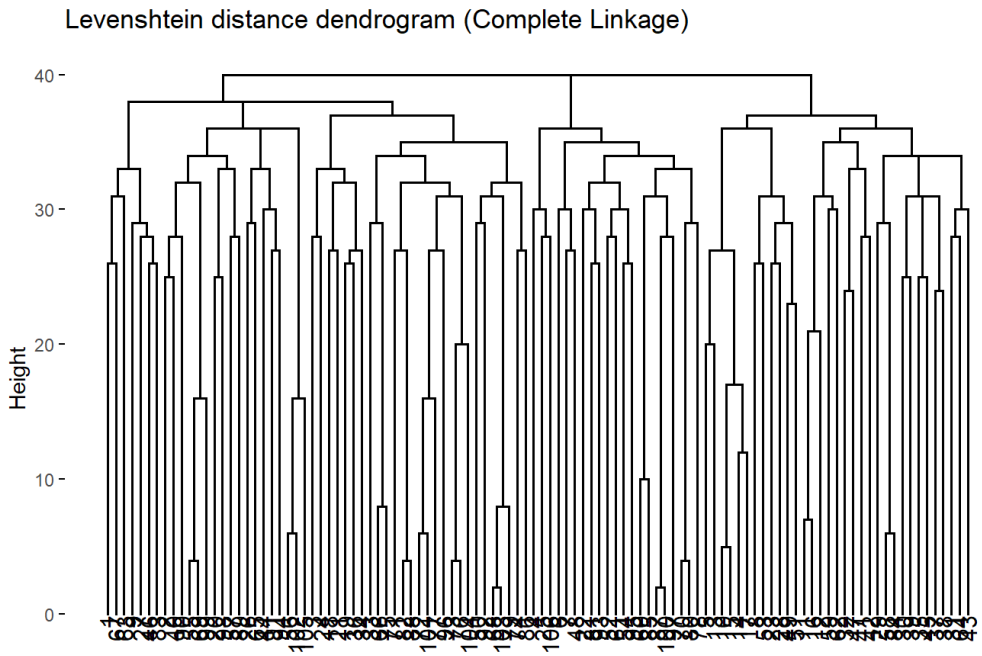
```
levenshtein_clusters_average <- agnes(levenshtein_distance_matrix, method = "average", diss = TRUE)

levenshtein_clusters_average_pam <- pam(levenshtein_distance_matrix, k=2, diss=TRUE)


# Step 6: Plot the Levenshtein distance clusters using different linkage methods.

plot_dendrogram_single <- fviz_dend(levenshtein_clusters_single, main = "Levenshtein distance dendrogram (Single Linkage)")

plot_dendrogram_complete <- fviz_dend(levenshtein_clusters_complete, main = "Levenshtein distance dendrogram (Complete Linkage)")

plot_dendrogram_average <- fviz_dend(levenshtein_clusters_average, main = "Levenshtein distance dendrogram (Average Linkage)")


plot(levenshtein_clusters_average_pam, main="Levenshtein distance, partitioning around medoids")
```



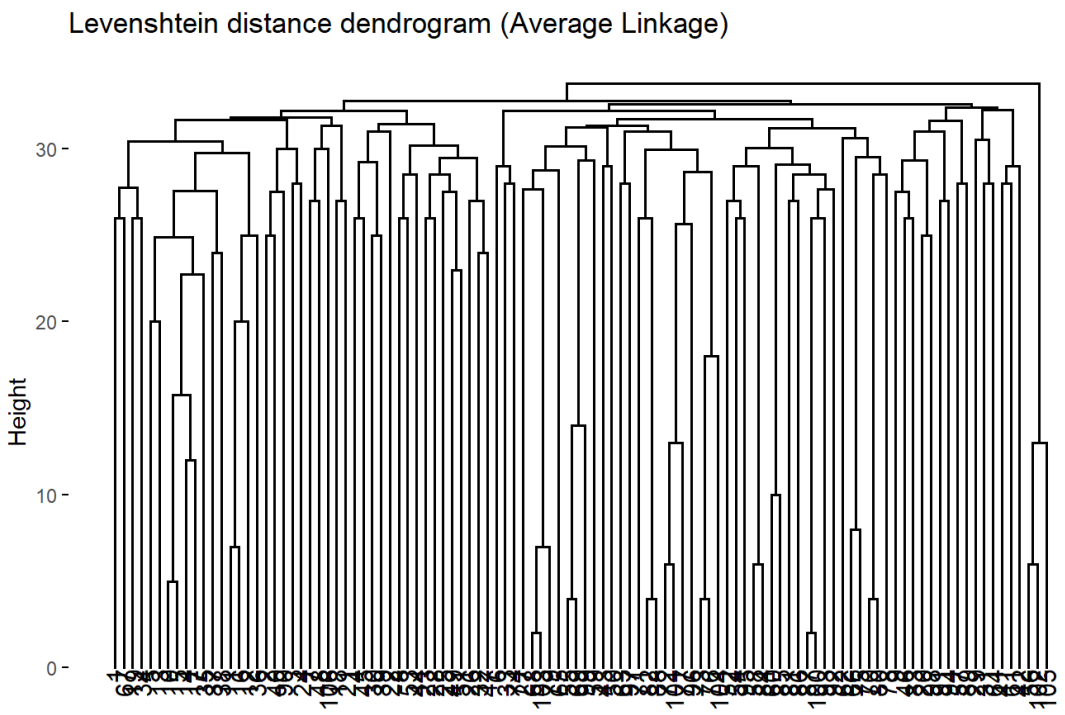
```
# Display the dendrogram plots
print(plot_dendrogram_single)
```




```
print(plot_dendrogram_complete)
```




```
print(plot_dendrogram_average)
```




```
# Optional: Cross-tabulate the clustering results with the X1 column in the promoters data
table(cutree(levenshtein_clusters_single, 2), promoters$X1)
```

```
##
##      -  +
##   1 53 52
##   2  0  1
```

```
table(cutree(levenshtein_clusters_complete, 2), promoters$X1)
```

```
##
##      -  +
##   1 34 18
##   2 19 35
```

```
table(cutree(levenshtein_clusters_average, 2), promoters$X1)
```

```
##
##      -  +
##   1 50 53
##   2  3  0
```

```
table(levenshtein_clusters_average_pam$clustering, promoters$X1)
```

```
##
##      -  +
##   1 16 48
```

Based on the results obtained using the Levenshtein distance and different clustering algorithms, the following performance metrics were observed:

Average Agglomerative Coefficient: 0.4

Single Agglomerative Coefficient: 0.32

Complete Agglomerative Coefficient: 0.48

PAM (Partitioning Around Medoids) Average silhouette width: 0.03

The Average Agglomerative Coefficient, Single Agglomerative Coefficient, and Complete Agglomerative Coefficient provide measures of the quality of the clustering solutions obtained from the agglomerative clustering algorithms. A higher value indicates better clustering results, with values closer to 1 representing well-separated clusters. The PAM Average silhouette width measures the compactness and separation of clusters, with higher values indicating better-defined clusters.

According to the results, the best classification performance was achieved using the Levenshtein distance and the PAM algorithm, with an Average Agglomerative Coefficient of 0.4 and a PAM Average silhouette width of 0.03. However, it is mentioned that there are 21 incorrectly classified observations.

The statement also suggests that achieving better performance with such short sequences would be challenging unless a scoring matrix reflecting the underlying biological laws of nucleic acid substitutions was used and the compared sequences were longer (hundreds of base pairs).

Concolusion

Based on these results, we can see that the Levenshtein distance method generally outperformed the Hamming distance method in terms of clustering quality. The Levenshtein distance method achieved higher values for the agglomerative coefficients (Single, Complete, and Average), indicating better separation and distinction between the clusters. Additionally, the PAM Average silhouette width was also higher for the Levenshtein distance, suggesting more compact and well-separated clusters.

These findings suggest that the Levenshtein distance, which considers insertions, deletions, and substitutions between sequences, captures more meaningful dissimilarity information compared to the Hamming distance, which only measures differences in positions. The Levenshtein distance takes into account the similarity and differences in the entire sequence, allowing for a more accurate representation of the dissimilarity between DNA sequences.

Therefore, when clustering DNA sequences or any other data with similar characteristics, it is generally more appropriate to use the Levenshtein distance or other sequence alignment-based methods that account for insertions, deletions, and substitutions, rather than the Hamming distance.