PROJECT: PRODUCT SEGMENTATION USING HIERARCHICAL CLUSTERING ON BREWDOG DATA

Programming Language: R

Author: Gaurav Kumar

1. EXECUTIVE SUMMARY:

1.1. Background of company:

BrewDog is a multinational brewery company which produces wide variety of beers. It produces 800,000 hectolitres of beer annually(Wikipedia, 2022).

1.2. Consulting Tasks:

BrewDog has provided dataset of 199 different beers and requires assistance in clustering the similar beers for marketing them to customers.

1.3. Recommendations:

A dendrogram has been developed using clustering which has segmented beers into 4 clusters. This can be used to market beers to customer according to their needs thus bringing personalisation.

2. HANDLING MISSING DATA:

2.1. Analysing the characteristics of data:

2.1.1. Introduction:

Here, we will analyse the data provided by BrewDog using R programming language. We will use functions which include summary(), md.pattern() under MICE package, aggr() under VIM package, matrixplot() under VIM package. We will check the basic characteristics of data such as datatype, min, max, mean etc.

2.1.2. summary() function:

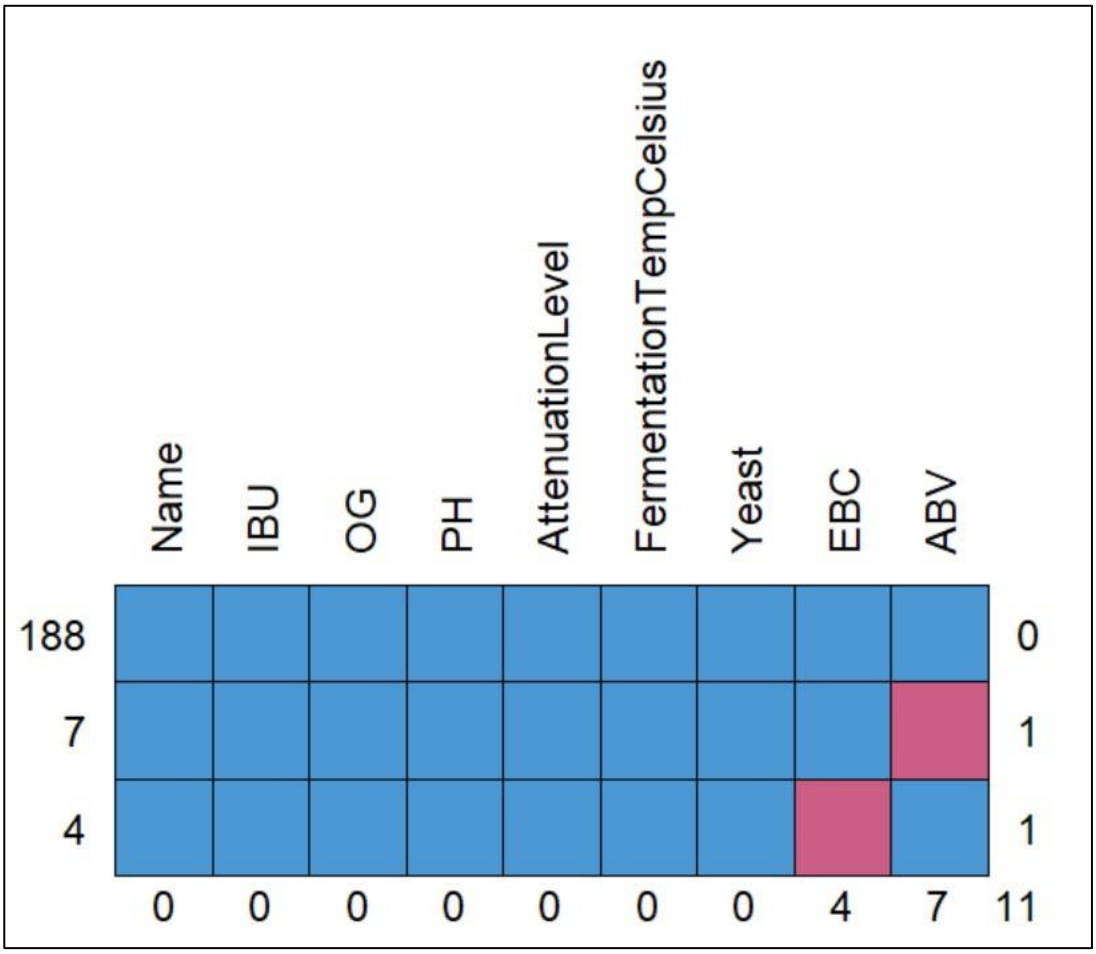
```
> summary(brewdog)
     Name                ABV               IBU                OG
 Length:199         Min.   : 0.500   Min.   :   0.00   Min.   :1007
 Class :character   1st Qu.: 5.200   1st Qu.:  40.00   1st Qu.:1048
 Mode  :character   Median : 7.200   Median :  55.00   Median :1065
                    Mean   : 7.675   Mean   :  67.48   Mean   :1065
                    3rd Qu.: 9.000   3rd Qu.:  75.00   3rd Qu.:1080
                    Max.   :41.000   Max.   :1085.00   Max.   :1156
                    NA's   :7
      EBC                PH        AttenuationLevel FermentationTempCelsius
 Min.   :  2.00   Min.   :3.200   Min.   : 28.60   Min.   : 9.00
 1st Qu.: 17.50   1st Qu.:4.400   1st Qu.: 76.60   1st Qu.:19.00
 Median : 30.00   Median :4.400   Median : 80.70   Median :19.00
 Mean   : 71.66   Mean   :4.409   Mean   : 80.30   Mean   :19.36
 3rd Qu.: 83.00   3rd Qu.:4.400   3rd Qu.: 83.25   3rd Qu.:21.00
 Max.   :500.00   Max.   :5.200   Max.   :102.30   Max.   :99.00
 NA's   :4
    Yeast
 Length:199
 Class :character
 Mode  :character
```

From above, following points can be noted:

a. There are 9 columns in data.
b. Column “Name” and “Yeast” have datatype as character and rest all columns have numeric data.
c. Column “ABV” has 7 missing values.
d. Column “EBC” has 4 missing values.
e. Also we can see Min , Max, Mean, Median, First Quartile and Third Quartile data for columns having numeric data.

2.1.3. md.pattern() function under MICE package:

Command: md.pattern(brewdog, rotate.names = TRUE)



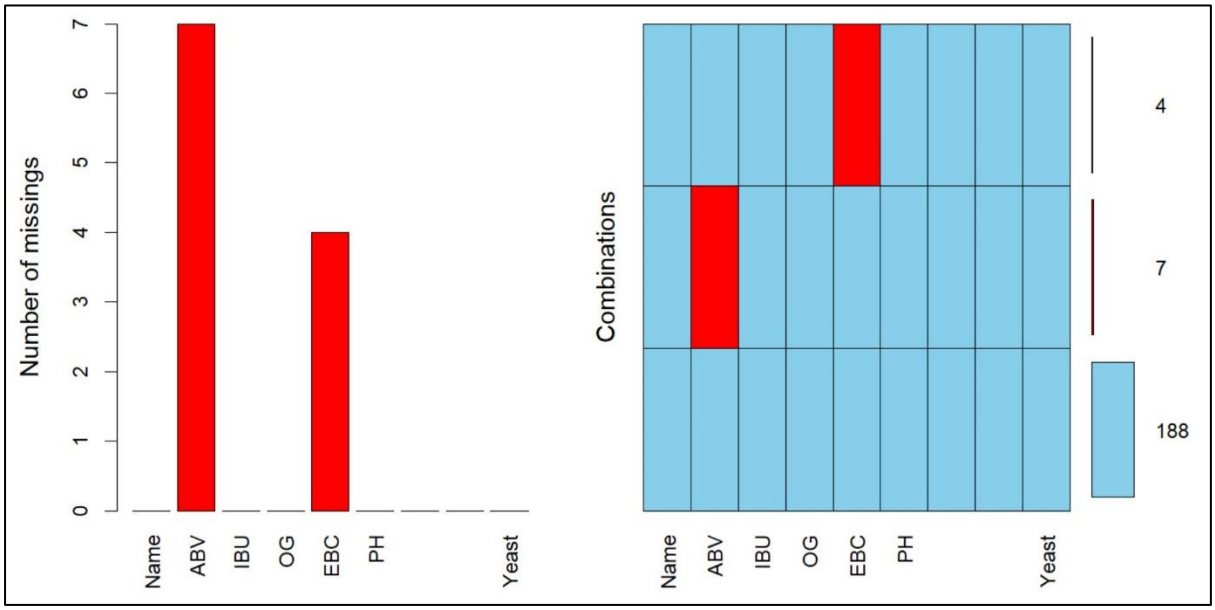
From above, following points can be noted:

a. 188 rows have complete data. This means they have data in all 9 columns.
b. ABV data is missing in 7 rows.
c. EBC data is missing in 4 rows.
d. There is no row where both ABV and EBC are missing.

The information about missing data is same what is provided by summary() function. Here, we get to know additional information that there is no row where both ABV and EBC are missing.

2.1.4. aggr() function under VIM package:

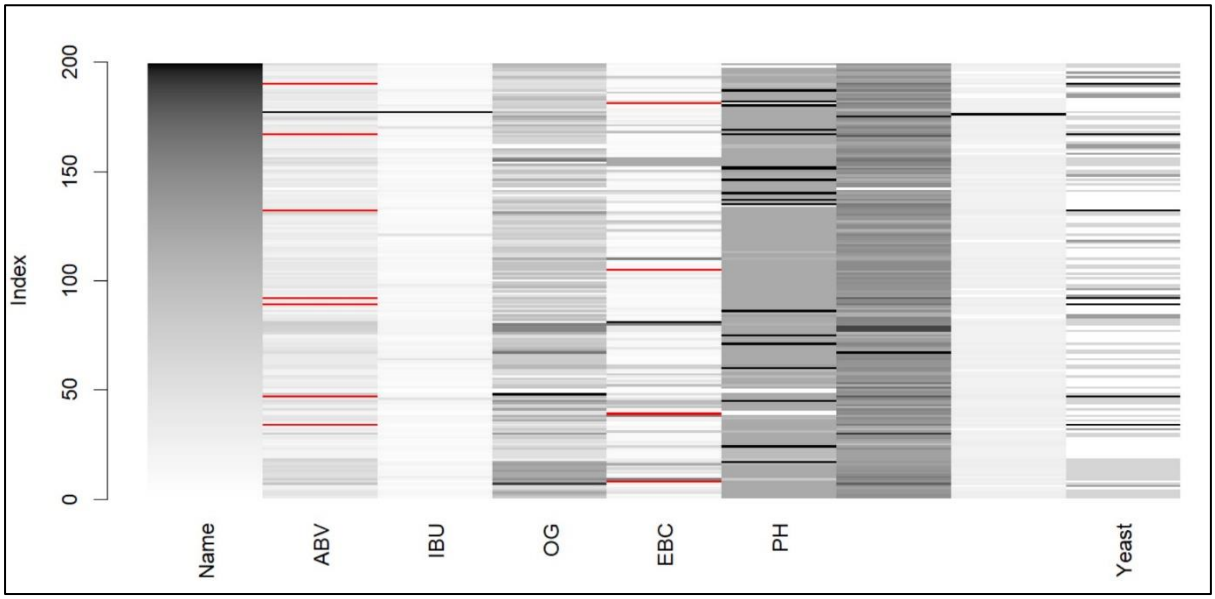
Command: aggr(brewdog, numbers = TRUE, prop = FALSE)



Findings are the same as provided by md.pattern function.

2.1.5. matrixplot() under VIM package:

Command: matrixplot(brewdog)



This is visual representation of each row where red colour denotes missing data.

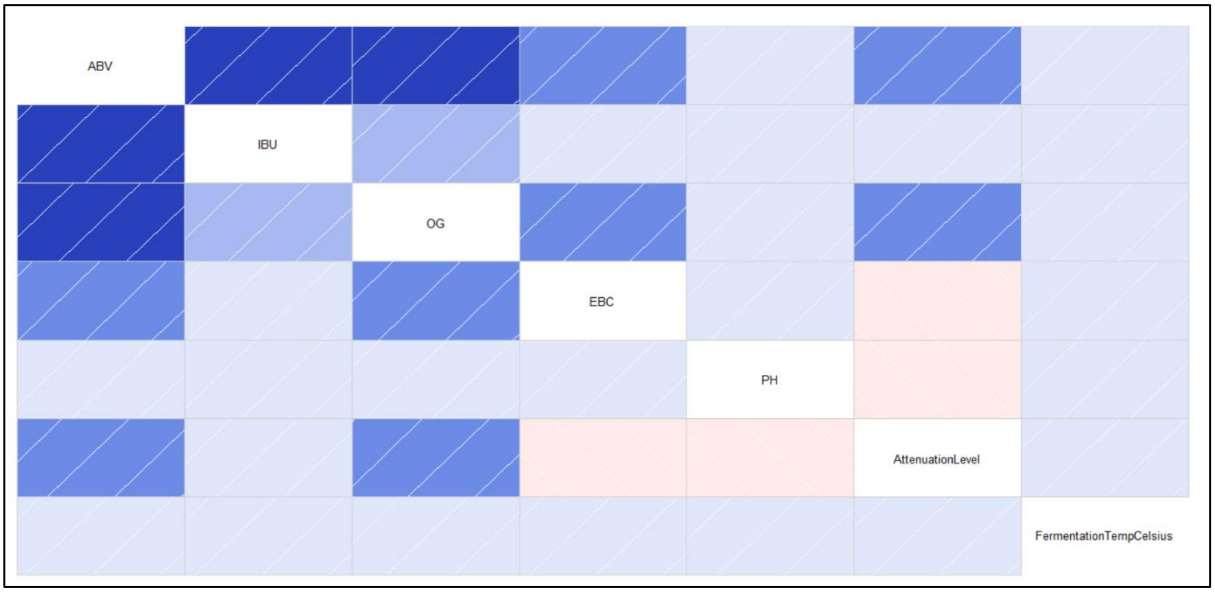
2.2. Identifying the reasons for missing data:

2.2.1. Introduction:

We will use corrgram() function under corrgram package to determine the reasons for missing data. This will enable us to decide which method of handling missing data should be used.

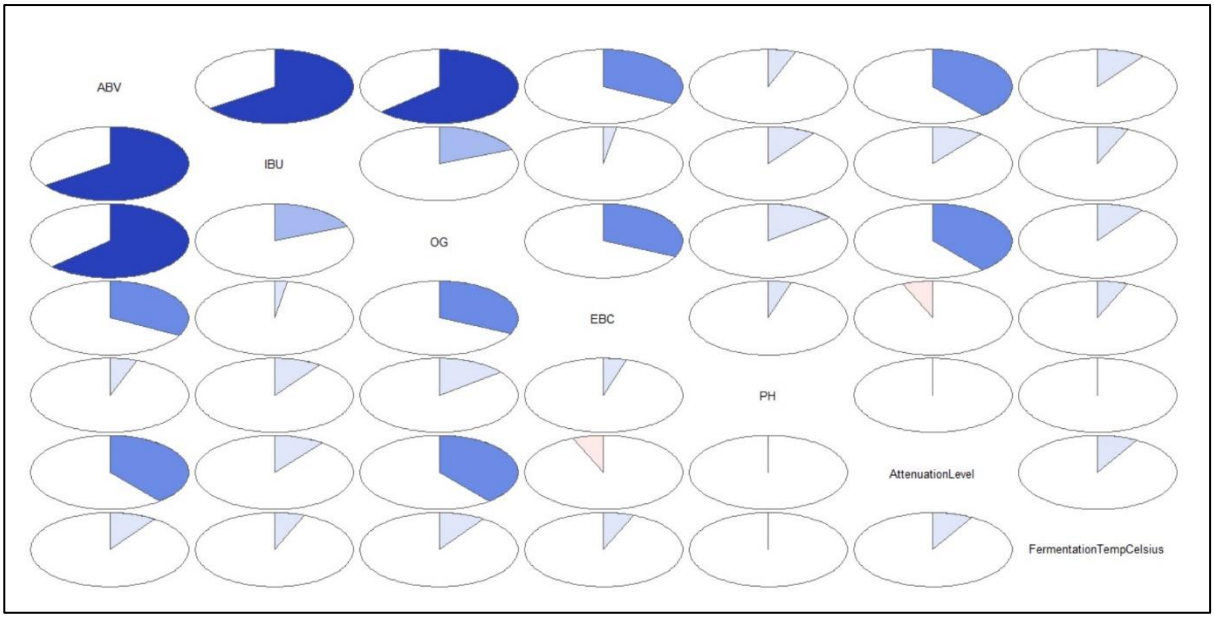
2.2.2. corrgram()

Command: corrgram(brewdog)



Changing the corrgram plot to pie-chart makes it easy to understand.

Command: corrgram(brewdog, lower.panel=panel.pie, upper.panel=panel.pie)



From above, following can be discerned:

a. Strong positive co-relation between ABV and IBU.
b. Strong positive co-relation between ABV and OG.
c. Also, positive co-relation exists between ABV & EBC, OG & EBC, ABV & Attenuation Level, OG & Attenuation Level.

2.2.3. <u>Deciding on type of missing data (MCAR, MAR or MNAR):</u>

2.2.3.1. It is important to decide what is the type of missing data because this will help is choosing which imputation methodology is suitable. The type of missing data can be any of three viz Missing Completely at Random (MCAR), Missing at Random (MAR) and Missing Not at Random (MNAR).

2.2.3.2. From above analysis, it is clear that **co-relation exists** in missing variables ie ABV and EBC. ABV is strongly related to IBU and OG. EBC is related to ABV and OG. Hence, reason for missing data is **not** MCAR.

2.2.3.3. So, it can be **either** MAR or MNAR.

2.2.3.4. To classify the reason for missing data as MNAR, there should be something in the value of missing data itself. Here, missing variables are ABV and EBC. ABV is alcohol by volume and EBC is European Brewery Convention (refers to color). This information requires domain knowledge. It can be gathered by **interviewing the client** or **asking experts** if any possibility exists that data is missing because of value of data itself. The data can be arranged in ascending or descending order to highlight the missing values for client/expert to have a look.

2.2.3.5. For purpose of this project, we are assuming that the reason for missing data is MAR.

2.3. Imputation methodology:

2.3.1. Introduction on imputation methods:

2.3.1.1. So, when reason for missing data is MAR, "deletion" method **cannot** be used to handle missing data as co-relation exist. Therefore, **"imputation"** will be used to handle missing data.

2.3.1.2. There are two ways of imputation ie Simple Imputation and Multiple Imputation. We will deploy both methods and determine which one is appropriate in this case.

2.3.2. Simple Imputation (SI):

2.3.2.1. Here, we are replacing the missing values with **mean** of values of that column to represent the complete dataset.

2.3.2.2. Command:

```
#Simple Imputation
brewdogsi <- brewdog
brewdogsi$ABV[!complete.cases(brewdogsi$ABV)] <- mean(brewdogsi$ABV
                                                      , na.rm =TRUE)
brewdogsi$EBC[!complete.cases(brewdogsi$EBC)] <- mean(brewdogsi$EBC
                                                      , na.rm =TRUE)
summary(brewdogsi)
```

2.3.3. Multiple imputation (MI):

2.3.3.1. Multiple imputation (MI) is carried out using MICE package in R. MI uses repeated simulations to impute missing values.

2.3.3.2. Command:

```
#Multiple Imputation
impute <- mice(brewdog, m = 10, seed = 1234)
impute$imp
impute
brewdogimp <- complete(impute)
summary(brewdogimp)
```

2.3.4. Comparing Simple Imputation and Multiple Imputation:

We will compare SI and MI using summary() function and histogram function.

2.3.4.1. summary() function:

Summary of SI:

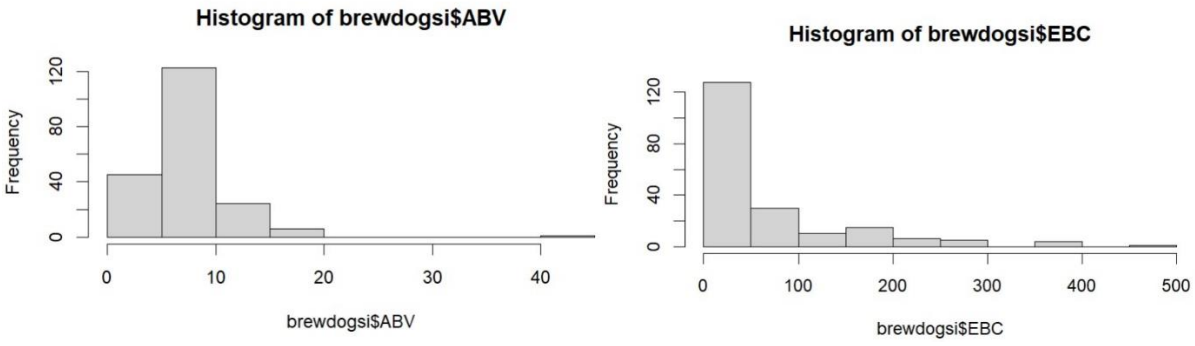
```
> summary(brewdogsi)
     Name               ABV              IBU              OG
 Length:199         Min.   : 0.500   Min.   :   0.00   Min.   :1007
 Class :character   1st Qu.: 5.200   1st Qu.:  40.00   1st Qu.:1048
 Mode  :character   Median : 7.200   Median :  55.00   Median :1065
                    Mean   : 7.675   Mean   :  67.48   Mean   :1065
                    3rd Qu.: 8.650   3rd Qu.:  75.00   3rd Qu.:1080
                    Max.   :41.000   Max.   :1085.00   Max.   :1156
      EBC                PH        AttenuationLevel FermentationTempCelsius
 Min.   :  2.00   Min.   :3.200   Min.   : 28.60   Min.   : 9.00
 1st Qu.: 18.00   1st Qu.:4.400   1st Qu.: 76.60   1st Qu.:19.00
 Median : 30.00   Median :4.400   Median : 80.70   Median :19.00
 Mean   : 71.66   Mean   :4.409   Mean   : 80.30   Mean   :19.36
 3rd Qu.: 79.50   3rd Qu.:4.400   3rd Qu.: 83.25   3rd Qu.:21.00
 Max.   :500.00   Max.   :5.200   Max.   :102.30   Max.   :99.00
    Yeast
 Length:199
 Class :character
 Mode  :character
```

Summary of MI:

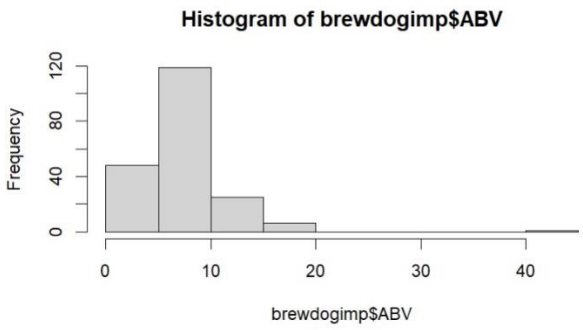
```
> summary(brewdogimp)
     Name               ABV              IBU              OG
 Length:199         Min.   : 0.500   Min.   :   0.00   Min.   :1007
 Class :character   1st Qu.: 5.200   1st Qu.:  40.00   1st Qu.:1048
 Mode  :character   Median : 7.200   Median :  55.00   Median :1065
                    Mean   : 7.626   Mean   :  67.48   Mean   :1065
                    3rd Qu.: 8.900   3rd Qu.:  75.00   3rd Qu.:1080
                    Max.   :41.000   Max.   :1085.00   Max.   :1156
      EBC                PH        AttenuationLevel FermentationTempCelsius
 Min.   :  2.0    Min.   :3.200   Min.   : 28.60   Min.   : 9.00
 1st Qu.: 17.0    1st Qu.:4.400   1st Qu.: 76.60   1st Qu.:19.00
 Median : 30.0    Median :4.400   Median : 80.70   Median :19.00
 Mean   : 71.1    Mean   :4.409   Mean   : 80.30   Mean   :19.36
 3rd Qu.: 83.0    3rd Qu.:4.400   3rd Qu.: 83.25   3rd Qu.:21.00
 Max.   :500.0    Max.   :5.200   Max.   :102.30   Max.   :99.00
    Yeast
 Length:199
 Class :character
 Mode  :character
```

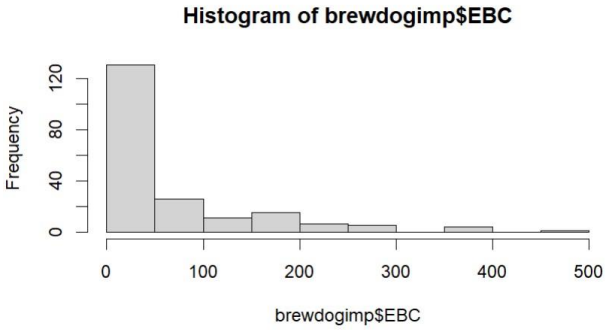
2.3.4.2. <u>Histogram function:</u>

For Simple Imputation:



For Multiple Imputation:





2.3.5. Why Multiple Imputation is selected:

2.3.6.

2.3.6.1. Both simple imputation and multiple imputation are giving near similar results. This is evident from analysing summary data and histograms.

2.3.6.2. Thus, both methods are **equally valid** here.

2.3.6.3. For the purpose of this assignment, Multiple Imputation (MI) method is selected just because MI is statistically strong, and data is not so large that computation takes very long time.

3. CLUSTERING THE DATA:

3.1. Scaling the data:

3.1.1. Firstly, data would be scaled because variables with different measurement units are present in data set.

```
#Using the brewdogimp matrix imputed using MI
brewdogimp
brewdogmi <- brewdogimp
summary(brewdogmi)
#Scaling the data
brewdogmi[,2:8] <- scale(brewdogmi[,2:8], center=TRUE, scale=TRUE)
```

3.2. Creating dissimilarity matrix:

3.2.1. Since the data contains **both** numeric and categorical data, dissimilarity matrix **cannot** be created using "**dist**" function. It is because "**dist**" function calculates Euclidean distance. Categorical values cannot be measured like Euclidean distance.

3.2.2. Distance between Categorical values can be measured using **Gower's** distance using "**daisy"** function in R. "**daisy"** function will automatically use Euclidean distance for numerical data and Gower's distance for categorical data.

3.2.3. Before using the daisy function, "Yeast" categorical column must be converted to factors. Otherwise, daisy function would not work.

```
#Creating dissimilarity matrix using daisy
brewdogmi
brewdogmi$Yeast <- as.factor(brewdogmi$Yeast)
summary(brewdogmi)
bd <- daisy(brewdogmi[2:9])
```

3.3. Selecting the agglomerative clustering methods:

3.3.1. There are **seven** clustering methods viz 'Single', 'Complete', 'Average (UPGMA)', 'Weighted (WPGMA)', 'Ward', 'Centroid' and 'Median'.

3.3.2. The data before us is mix of numeric and categorical data which will influence the selection of method.

3.3.3. 'Ward', 'Centroid' and 'Median' methods **cannot** be used because these methods assume data can use Euclidean distance (Everitt et al., 2010, p.79). Whereas categorical data cannot use Euclidean distance.

3.3.4. Also 'Single' and 'Complete' methods will **not** be used as they do not take into account cluster structure (Everitt et al., 2010, p.79).

3.3.5. Out of 'Average' and 'Weighted', method '**Weighted**' will be used as this method is better when there is likeliness of cluster size being uneven (Everitt et al., 2010, p.79).

3.4. Performing the agglomerative clustering:

3.4.1. To perform agglomerative clustering "**agnes**" function would be used instead of hclust function.

```
#Performing agglomerative clustering
clust <- agnes(bd, diss = TRUE, method = "weighted")
```

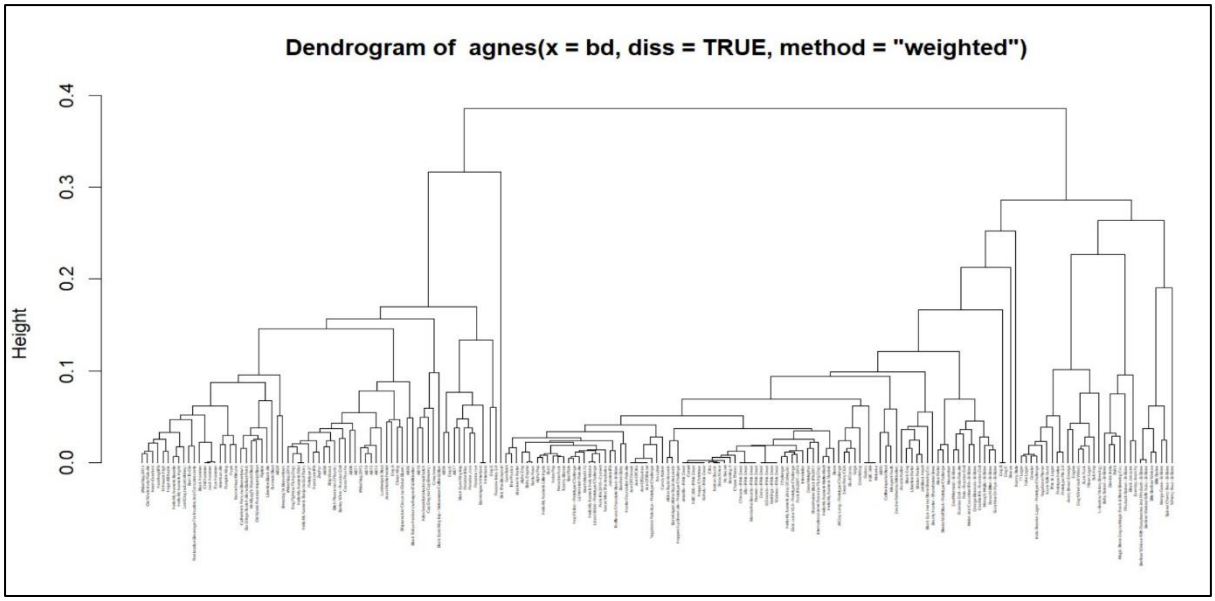
4. PLOTTING THE CLUSTER:

4.1. Plotting the cluster into dendrogram:

Command:

```
#Plotting a dendrogram
plot(clust,labels=brewdogmi$Name, which.plots=2, cex = 0.2, hang = -1)
```

Output:



5. Selecting number of clusters:

5.1. The output of hierarchical clustering is a dendrogram. Here, user has to **choose** the number of clusters.

NbClust is one way to determine the number of clusters. NbClust provides results from 30 indices. In our case, only **Five** indexes (viz frey, mcclain, cindex, silhouette and dunn) can be used because only these 5 indices take dissimilarity matrix as input and no data input.

Command:

```
#Determining the number of clusters
#Using silhouette index
res <- NbClust(diss = bd, distance = NULL, min.nc=2, max.nc=15
               , method = "mcquitty"
               , index = "silhouette")
res$Best.nc
#Using frey index
res1 <- NbClust(diss = bd, distance = NULL, min.nc=2, max.nc=15
               , method = "mcquitty"
               , index = "frey")
res1$Best.nc
#Using mcclain index
res2 <- NbClust(diss = bd, distance = NULL, min.nc=2, max.nc=15
               , method = "mcquitty"
               , index = "mcclain")
res2$Best.nc
#Using cindex
res3 <- NbClust(diss = bd, distance = NULL, min.nc=2, max.nc=15
               , method = "mcquitty"
               , index = "cindex")
res3$Best.nc
#Using dunn
res4 <- NbClust(diss = bd, distance = NULL, min.nc=2, max.nc=15
                , method = "mcquitty"
                , index = "dunn")
res4$Best.nc
```

Results:

Sr No	Index	Number of cluster recommended
1	Frey	1
2	Mcclain	2
3	Cindex	15
4	Silhouette	7
5	Dunn	3

No clarity can be found here regarding number of clusters.

5.1.1. In dendrogram, the **height** at which the leaves join to form a node determines the closeness between the leaves. So, less height means leaves are closer to each other. Alternatively, if they join at greater height, it means they are less similar. So, from visual examination of dendrogram it can be discerned that 4 clusters can be formed.

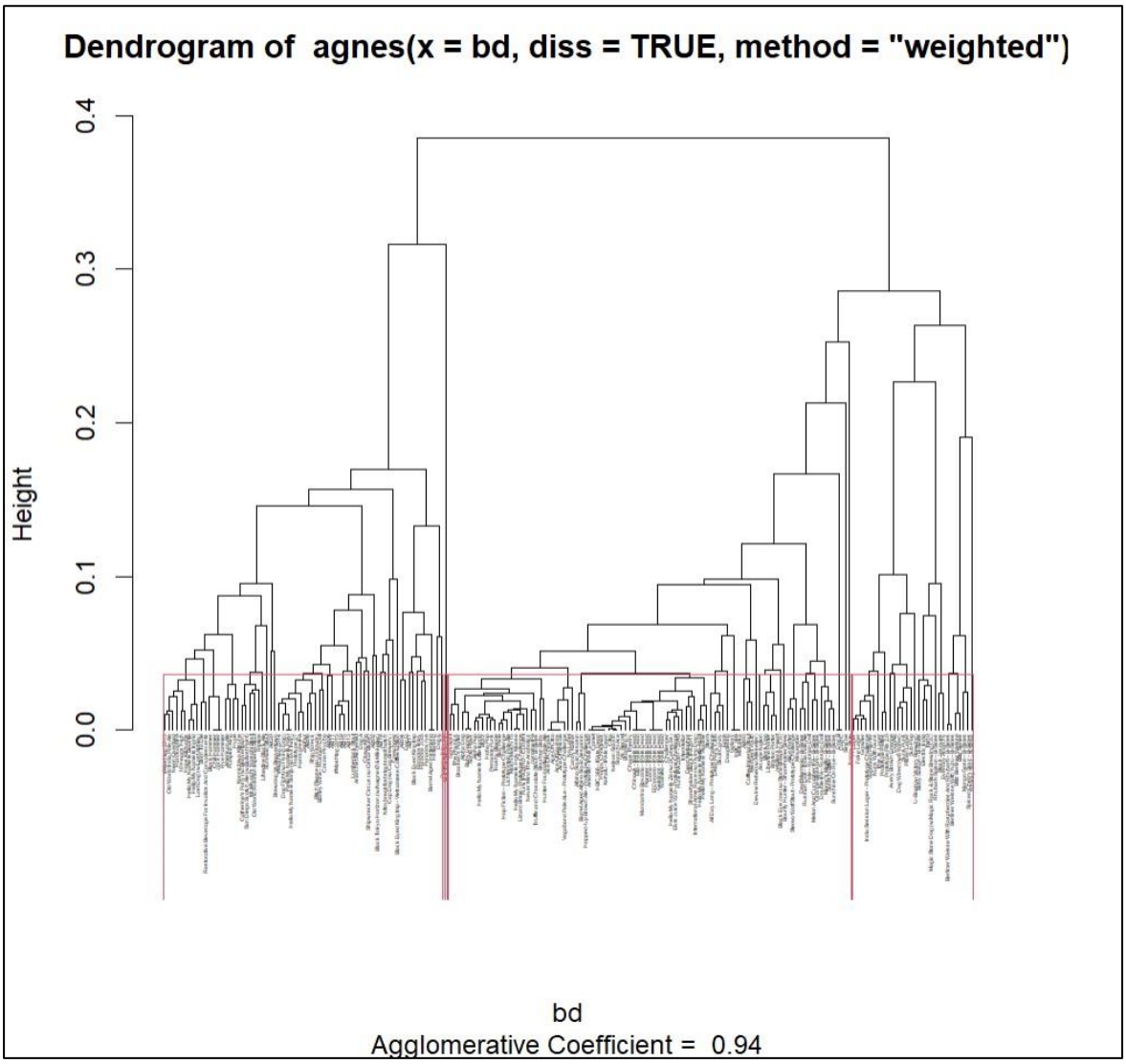
Command:

```
#Choosing number of clusters as 4
cluster <- data.frame(brewdogimp, clusterNo = cutree(clust, k = 4))
library("dplyr")
cluster %>% group_by(clusterNo) %>% count(clusterNo)
cluster[c(1,9,10)][order(cluster[10]),]
#Creating rectangle for k=4
rect.hclust(clust,4)
```

5.1.2. Number of values distributed cluster wise:

```
> cluster %>% group_by(clusterNo) %>% count(clusterNo)
# A tibble: 4 x 2
# Groups:   clusterNo [4]
  clusterNo     n
      <int> <int>
1         1    69
2         2    99
3         3    30
4         4     1
```

Dendrogram with 4 clusters:



6. CONCLUSION:

In order to derive value from the clustering performed above, client intervention is required. Now, data will be analysed cluster wise with involvement of client to see what the areas of similarity in each cluster are. We can look at “Yeast” also because there are only 4 varieties of yeast.

The practical implementation of this clustering can be that when a customer arrives in a bar he/she can be offered 4 samples of beers one from each cluster as taster. The one which is liked by customer will determine which cluster to choose to make offering to customer.