Grouping the Penguins: A Clustering Analysis on the Penguin dataset.

GIFT E. IDAMA
Computer and Information Sciences Department
Towson University,
Towson, MD, USA.
gidama1@students.towson.edu

Abstract: Many real-world systems can be studied in terms of pattern recognition tasks, so that proper use (and understanding) of machine learning methods in practical applications becomes essential. While many classification methods have been proposed, there is no consensus on which methods are more suitable for a given dataset. As a consequence, it is important to comprehensively compare methods in many possible scenarios. In this context, we performed a systematic comparison of 3 well-known clustering methods available in the R language assuming normally distributed data. In order to account for the many possible variations of data, we considered artificial datasets with several tunable properties.

1. INTRODUCTION:

Clustering is an unsupervised learning problem.
It is often used as a data analysis technique for discovering interesting patterns in data, such as groups of customers based on their behavior. An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labeled responses. Generally, it is used as a process to find, meaningful, structure, explanatory, underlying processes, generative features, and groupings inherent in a set of examples. There are many clustering algorithms to choose from and no single best clustering algorithm for all cases. Instead, it is a good idea to explore a range of clustering algorithms and different configurations for each algorithm.

1.1 Importance of Clustering:

Clustering is very much important as it determines the intrinsic grouping among the unlabeled data present. There are no criteria for good clustering. It depends on the user, what is the criteria they may use which satisfy their need.

2. DESCRIPTIONS OF THE PENGUIN DATASET:

The dataset contain data for 344 penguins. There are 3 different species of penguins in the dataset collected from 3 islands in the palmer archipelago, Antarctica. The species include Adelie, Chinstrap, and Gentoo. It also shows the island on which these species can be found. The 3 islands in our dataset are Torgersen, Biscoe and Dream Island. There are about 344 observations, and 8 variables present in our dataset. It also contains the physical characteristics of each specie as well as the genders of the penguins.

The dataset contains 3 categorical variables and 5 continuous variables. The variables include specie, island, bill length, bill depth, flipper length, body mass, sex and year. 3 variables are categorical (sex, island, species) while the rest are numeric.

Penguins Data Column Definition

- **species** a factor denoting penguin species (Adelie, Chinstrap and Gentoo)
- **island** a factor denoting island in Palmer Archipelago, Antarctica (Biscoe, Dream or Torgersen)
- **bill_length_mm** a number denoting bill length (millimeters)
- **bill_depth_mm** a number denoting bill depth (millimeters)
- **flipper_length_mm** an integer denoting flipper length (millimeters)
- **body_mass_g** an integer denoting body mass (grams)
- **sex** a factor denoting penguin sex (female, male)

3. PROJECT OBJECTIVES

This project looks to compare various clustering algorithms on the penguin dataset and compare the results of each algorithm. For this project, 3 main algorithms would be considered.

1. K-means clustering
2. Hierarchical clustering
3. Model based clustering

Before we dive into analyzing each individual algorithm, we must for scale our data set because clustering algorithms are only able to cluster on continuous variables rather than categorical variables. we need to get rid of those categorical variables present in our dataset by scaling the data set. Also, another factor to handle is dealing with the missing values present in the dataset.

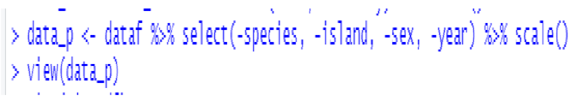
In the next couple of sections, we are going to handle the missing values in the dataset, the scale the dataset before analyzing the 3-clustering algorithm we will be using in this study.

3.1 Handling Missing values.

The missing values in each column does not affect our dataset significantly, so eliminating these missing values should be a straightforward step. We can make use of the **"na.omit() function" to** remove the rows with **NA's.** After calling the function, we arrive at a data frame with no missing values present.

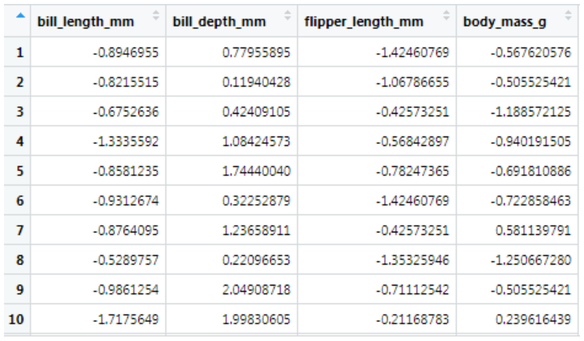
3.2 Dealing with categorical variables.

As mentioned earlier, clustering is only done on continuous variable, so first step in carrying out clustering analysis is removing these redundant variable/columns from the data set and the scaling the data set in case of any large values.



```
> data_p <- dataf %>% select(-species, -island, -sex, -year) %>% scale()
> view(data_p)
```

After removing the redundant variables and scaling the data set, we get the scaled data frame shown below.



	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
1	-0.8946955	0.77955895	-1.42460769	-0.567620576
2	-0.8215515	0.11940428	-1.06786655	-0.505525421
3	-0.6752636	0.42409105	-0.42573251	-1.188572125
4	-1.3335592	1.08424573	-0.56842897	-0.940191505
5	-0.8581235	1.74440040	-0.78247365	-0.691810886
6	-0.9312674	0.32252879	-1.42460769	-0.722858463
7	-0.8764095	1.23658911	-0.42573251	0.581139791
8	-0.5289757	0.22096653	-1.35325946	-1.250667280
9	-0.9861254	2.04908718	-0.71112542	-0.505525421
10	-1.7175649	1.99830605	-0.21168783	0.239616439

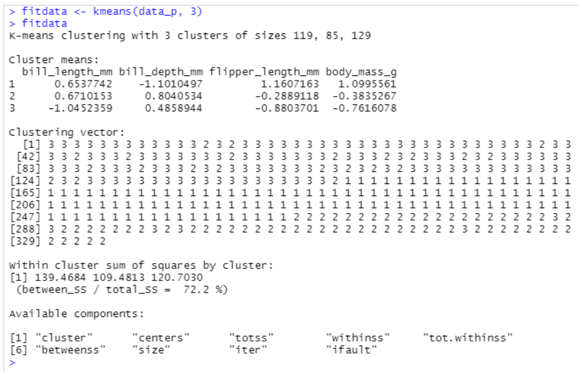
Fig 1: The scaled penguin dataset

Now that the dataset is scaled, we can now test our different clustering algorithm on the scaled dataset. As mentioned earlier, 3 clustering algorithms are considered in this study, k-means clustering, hierarchical clustering and model-based clustering. We start with the k-means clustering.

4. K-MEANS CLUSTERING

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labelled, outcomes.

First, we are going to fit the scaled data into a model and analyze the result (Here, we are assuming k = 3).



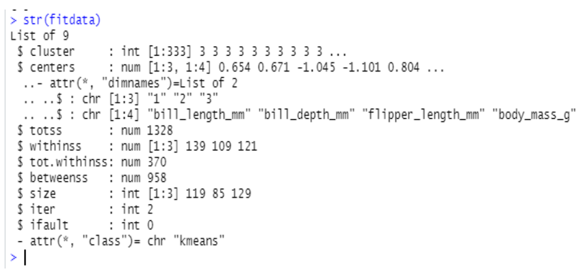
```
> fitdata <- kmeans(data_p, 3)
> fitdata
K-means clustering with 3 clusters of sizes 119, 85, 129

Cluster means:
  bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
1      0.6537742    -1.1010497         1.1607163   1.0995561
2      0.6710153     0.8040534        -0.2889118  -0.3835267
3     -1.0452359     0.4858944        -0.8803701  -0.7616078

Clustering vector:
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3
 [42] 3 3 2 3 3 3 2 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 2 3 3 3 2 3 2 3 3 3 2 3 2 3 3 3 3 3
 [83] 3 3 3 2 3 3 3 2 3 3 3 2 3 2 3 3 3 3 3 3 3 2 3 2 3 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3
[124] 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[165] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[206] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[247] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2
[288] 3 2 2 2 2 2 2 2 3 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2
[329] 2 2 2 2 2

Within cluster sum of squares by cluster:
[1] 139.4684 109.4813 120.7030
 (between_SS / total_SS =  72.2 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"      
>
```

Now, lets analyze our result/output. From the figure above, we have 3 clusters of 119, 85, 129 (These are the number of observations in each cluster). Next, we have the cluster means that is, the mean of each variable in each cluster. For instance, in cluster 1, the mean of bill length is 0.6537742 and the mean of bill depth is -1.1010497 while in cluster 2, mean of bill length is 0.6710153 and mean of bill depth is 0.8040534 and so on.

The clustering vector shows which cluster each individual observation falls in, for instance, observation 1 falls in cluster 3, observation 2 falls in cluster 3 while observation 3 falls in cluster 3 and so on.

Within cluster sum of squares is basically the measurement of how dispersed each cluster are.
Cluster 1 – 47.43819
Cluster 2 – 44.88754
Cluster 3 – 47.33662
between_ss/total_ss = 72.2%

Below is a structure of our fit model (fitdata)



```
> str(fitdata)
List of 9
 $ cluster     : int [1:333] 3 3 3 3 3 3 3 3 3 3 ...
 $ centers     : num [1:3, 1:4] 0.654 0.671 -1.045 -1.101 0.804 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:3] "1" "2" "3"
  .. ..$ : chr [1:4] "bill_length_mm" "bill_depth_mm" "flipper_length_mm" "body_mass_g"
 $ totss       : num 1328
 $ withinss    : num [1:3] 139 109 121
 $ tot.withinss: num 370
 $ betweenss   : num 958
 $ size        : int [1:3] 119 85 129
 $ iter        : int 2
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
> |
```

Fig 3: Structure of the fit model

Creating a plot for the dataframe.

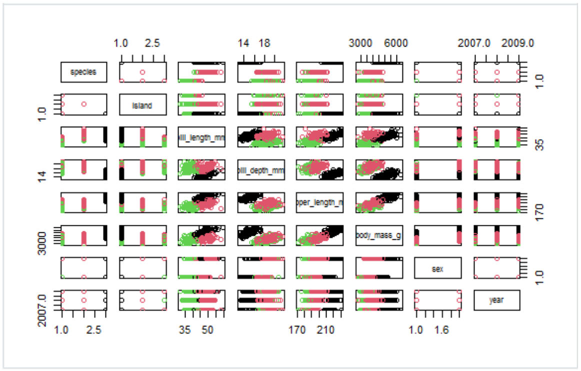
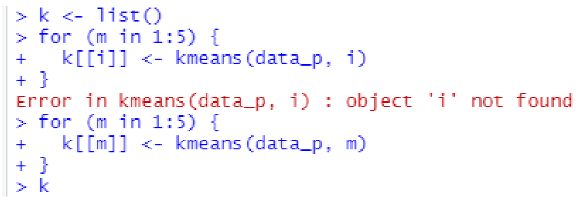


Fig 4: Plot of the dataframe

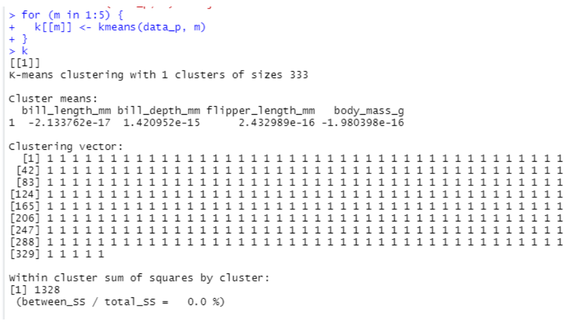
Based off the above chart, the algorithm did a good job separating each clusters but however we don't have a 100% accuracy based on this chart which begs the question, how do we choose the number of clusters (k) to identify and how do we evaluate the performance of the clustering algorithms.

4.1 How do we choose K.



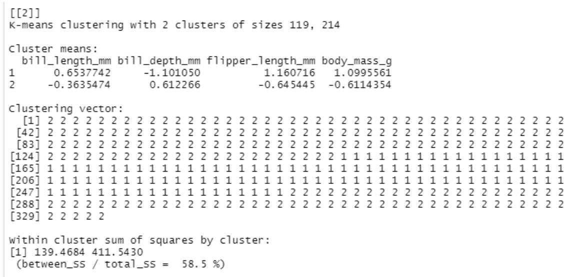
```
> k <- list()
> for (m in 1:5) {
+   k[[i]] <- kmeans(data_p, i)
+ }
Error in kmeans(data_p, i) : object 'i' not found
> for (m in 1:5) {
+   k[[m]] <- kmeans(data_p, m)
+ }
> k
```

The for loop iterates between values range of 1-5 and fit into the k-means model and it will be saved in a list called k as shown above. We use this for loop to compare the sum of squares percentage.



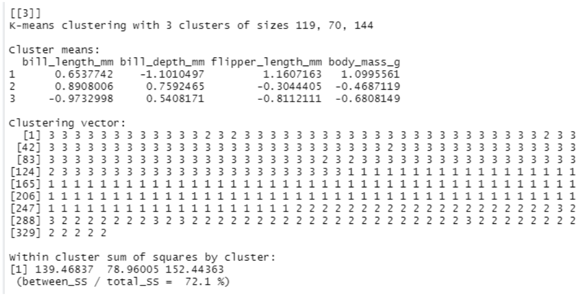
```
> for (m in 1:5) {
+   k[[m]] <- kmeans(data_p, m)
+ }
> k
[[1]]
K-means clustering with 1 clusters of sizes 333

Cluster means:
  bill_length_mm bill_depth_mm flipper_length_mm   body_mass_g
1  -2.133762e-17  1.420952e-15      2.432989e-16 -1.980398e-16

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [42] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [83] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[124] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[165] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[206] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[247] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[288] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[329] 1 1 1 1 1

Within cluster sum of squares by cluster:
[1] 1328
 (between_SS / total_SS =   0.0 %)
```

Fig 5: When k = 1



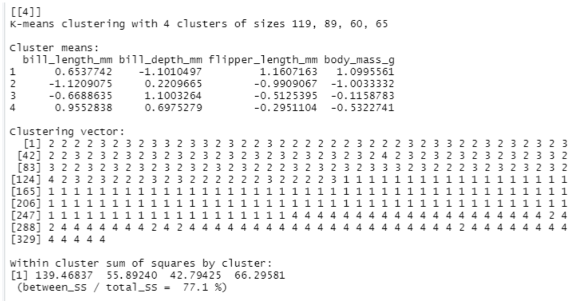
```
[[2]]
K-means clustering with 2 clusters of sizes 119, 214

Cluster means:
  bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
1      0.6537742     -1.101050          1.160716   1.0995561
2     -0.3635474      0.612266         -0.645445  -0.6114354

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [42] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [83] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[124] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[165] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[206] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[247] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[288] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[329] 2 2 2 2 2

Within cluster sum of squares by cluster:
[1] 139.4684 411.5430
 (between_SS / total_SS =  58.5 %)
```

Fig 6: When k = 2



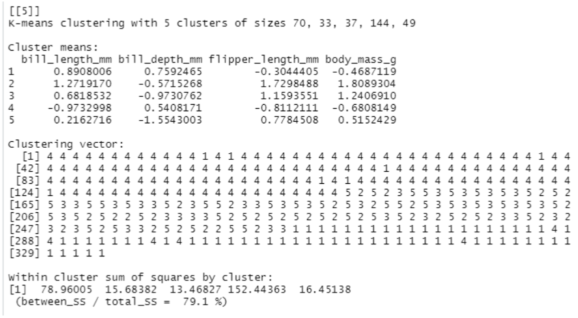
```
[[3]]
K-means clustering with 3 clusters of sizes 119, 70, 144

Cluster means:
  bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
1      0.6537742    -1.1010497         1.1607163   1.0995561
2      0.8908006     0.7592465        -0.3044405  -0.4687119
3     -0.9732998     0.5408171        -0.8112111  -0.6808149

Clustering vector:
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3
 [42] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [83] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[124] 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[165] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[206] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[247] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2
[288] 3 2 2 2 2 2 2 2 3 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2
[329] 2 2 2 2 2

Within cluster sum of squares by cluster:
[1] 139.46837  78.96005 152.44363
 (between_SS / total_SS =  72.1 %)
```

Fig 7: When k = 3



```
[[4]]
K-means clustering with 4 clusters of sizes 119, 89, 60, 65

Cluster means:
  bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
1      0.6537742    -1.1010497         1.1607163   1.0995561
2     -1.1209075     0.2209665        -0.9909067  -1.0033332
3     -0.6688635     1.1003264        -0.5125395  -0.1158783
4      0.9552838     0.6975279        -0.2951104  -0.5322741

Clustering vector:
  [1] 2 2 2 2 3 2 3 2 3 3 2 3 3 2 3 2 2 3 2 2 2 2 2 2 3 2 2 2 3 2 3 3 2 2 3 2 3 2 3 2 3
 [42] 2 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 4 2 3 2 3 2 3 2 3 2 3 2 3 3 2
 [83] 3 2 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 2 2 3 2 3 2 3 2 3 3 3 2 3 2 2 2 3 2 3 2 3 2 3 2
[124] 4 2 3 2 3 2 2 2 3 2 3 2 2 2 2 2 2 3 2 2 2 2 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[165] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[206] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[247] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 4
[288] 2 4 4 4 4 4 4 4 2 4 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 4 4 4 4 4 4 4 4
[329] 4 4 4 4 4

Within cluster sum of squares by cluster:
[1] 139.46837  55.89240  42.79425  66.29581
 (between_SS / total_SS =  77.1 %)
```

Fig 8: When k = 4



```
[[5]]
K-means clustering with 5 clusters of sizes 70, 33, 37, 144, 49

Cluster means:
  bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
1      0.8908006     0.7592465        -0.3044405  -0.4687119
2      1.2719170    -0.5715268         1.7298488   1.8089304
3      0.6818532    -0.9730762         1.1593551   1.2406910
4     -0.9732998     0.5408171        -0.8112111  -0.6808149
5      0.2162716    -1.5543003         0.7784508   0.5152429

Clustering vector:
  [1] 4 4 4 4 4 4 4 4 4 4 4 4 1 4 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 1 4 4
 [42] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 [83] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 1 4 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
[124] 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 2 5 2 3 5 5 3 5 3 5 3 5 3 5 2 5 2
[165] 5 3 3 5 5 3 5 3 3 5 2 3 5 5 2 3 3 5 3 5 3 5 2 5 3 2 5 5 2 5 3 5 3 5 3 5 3 5 3 5 2
[206] 5 3 5 2 5 2 2 5 2 3 3 3 3 5 2 5 2 5 2 5 2 5 2 5 2 5 3 5 2 3 2 5 2 5 2 3 3 5 2 3 2
[247] 3 2 3 5 2 5 3 3 2 5 2 5 2 2 5 5 2 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 1
[288] 4 1 1 1 1 1 1 1 4 1 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 1 1 1 1 1 1 1 1
[329] 1 1 1 1 1

Within cluster sum of squares by cluster:
[1]  78.96005  15.68382  13.46827 152.44363  16.45138
 (between_SS / total_SS =  79.1 %)
```

Fig 9: When k = 5

From the following figures, we can see that as the number of clusters increases, the sum of squares percentage also increases. The higher the percentage, the better.

For k = 1, sum of squares was 0% (which isn't good)

For k = 2, sum of squares was 58.5%

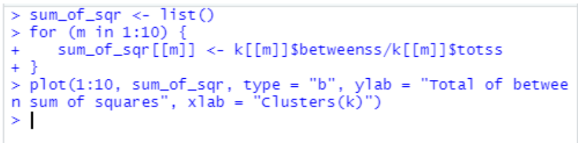
For k = 3, sum of squares was 72.1%

For k = 4, sum of squares was 77.1%

For k = 5, sum of squares was 79.1%

For k = 10, sum of squares was 88.7%

Next let's create a plot for the clusters against the sum of squares. Using a for loop, we find the total sum of squares and plot a graph showing the relation between the sum of squares and the clusters



```
> sum_of_sqr <- list()
> for (m in 1:10) {
+     sum_of_sqr[[m]] <- k[[m]]$betweenss/k[[m]]$totss
+ }
> plot(1:10, sum_of_sqr, type = "b", ylab = "Total of betwee
n sum of squares", xlab = "Clusters(k)")
> |
```

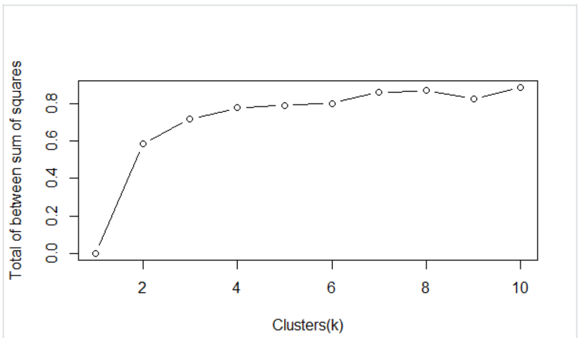


Fig 10: Plot showing the cluster vs sum of sq.

Observing the above chart, we can deduce that the number of clusters(k) is 2 or 3.

NOTE: it is up to the user to select a value for k

Now, let us consider using a method in order to determine the optimal number of clusters.

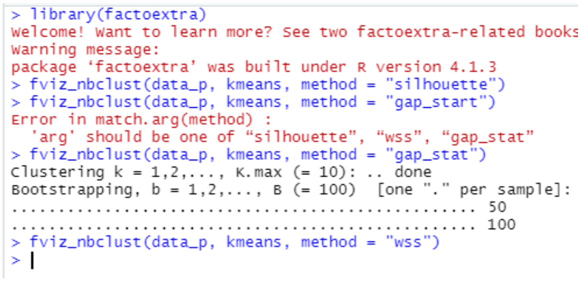
Using a package called factoextra, and using the method "wss", "silhouette" and "gap stat" we plot several graphs showing the optimal number of clusters for each method.

Based on the silhouette method, optimal number of clusters is 2

Based on the gap stat method, optimal number of clusters is 4

Based on the wss method, optimal number of clusters is 2 or 3

The visualizations are shown below.



```
> library(factoextra)
Welcome! Want to learn more? See two factoextra-related books
Warning message:
package ‘factoextra’ was built under R version 4.1.3
> fviz_nbclust(data_p, kmeans, method = "silhouette")
> fviz_nbclust(data_p, kmeans, method = "gap_start")
Error in match.arg(method) :
  'arg' should be one of “silhouette”, “wss”, “gap_stat”
> fviz_nbclust(data_p, kmeans, method = "gap_stat")
Clustering k = 1,2,..., K.max (= 10): .. done
Bootstrapping, b = 1,2,..., B (= 100)  [one "." per sample]:
.................................................. 50
.................................................. 100
> fviz_nbclust(data_p, kmeans, method = "wss")
> |
```

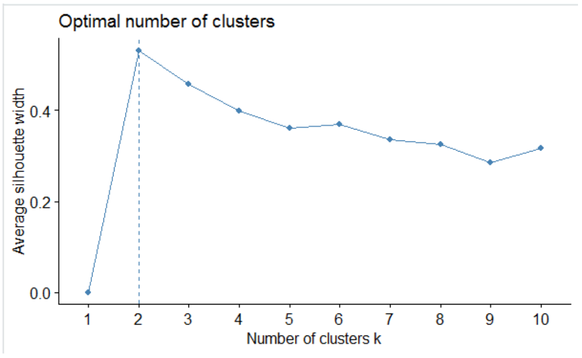


Fig 11: Silhouette method

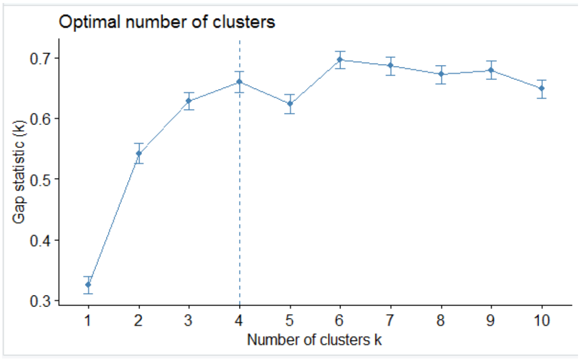


Fig 12: gap stat method.

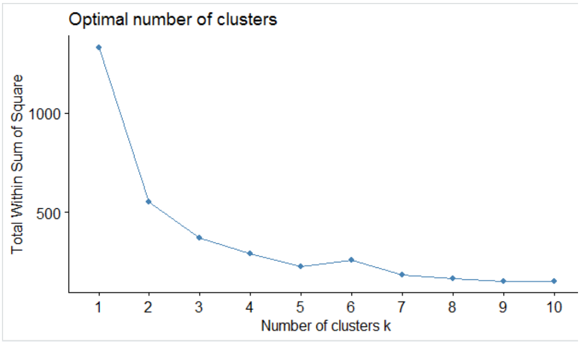
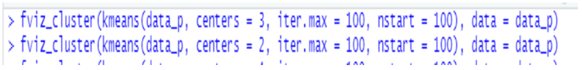


Fig 13: wss method.

Taking into consideration the silhouette method, optimal number of clusters(k) = 2.

Creating a cluster biplot,



```
> fviz_cluster(kmeans(data_p, centers = 3, iter.max = 100, nstart = 100), data = data_p)
> fviz_cluster(kmeans(data_p, centers = 2, iter.max = 100, nstart = 100), data = data_p)
```

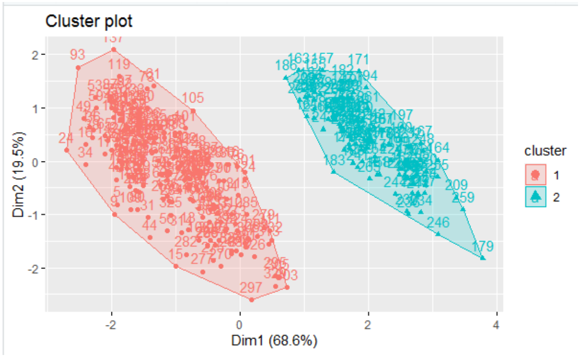


Fig 14: Cluster biplot (k = 2)

NOTE: percent in parenthesis is the amount of variability that is explained by each of the component.

From the plot above, we can see that the observations are well grouped and do not overlap. Let’s consider the wss method and take our k = 3, we arrive at the plot below.

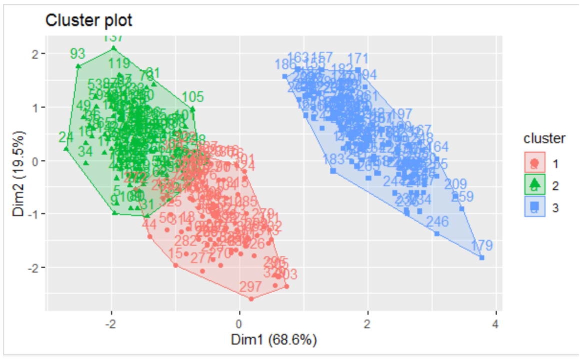
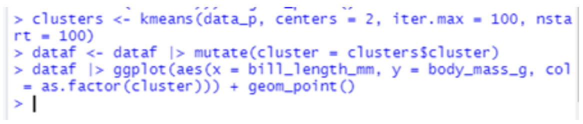


Fig 15: Cluster biplot (k = 3)

Only cluster 3 is well grouped off the other two clusters while cluster 1 and cluster 2 overlap. Let’s create few comparisons in our data frame



```
> clusters <- kmeans(data_p, centers = 2, iter.max = 100, nsta
rt = 100)
> dataf <- dataf |> mutate(cluster = clusters$cluster)
> dataf |> ggplot(aes(x = bill_length_mm, y = body_mass_g, col
 = as.factor(cluster))) + geom_point()
> |
```

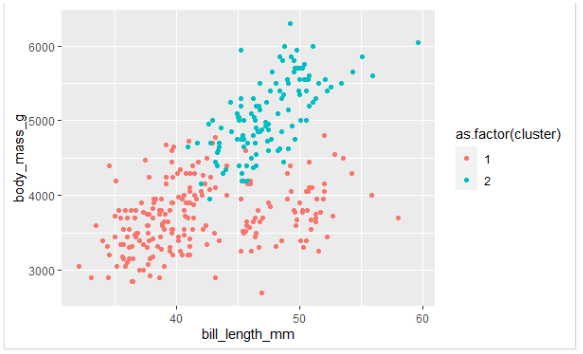


Fig 16: body mass vs bill length

Observations:

- Observations with lower body mass and shorter bill length have been assigned mostly to cluster 1
- Observations with higher body mass and longer bill length have been assigned mostly to cluster 2.

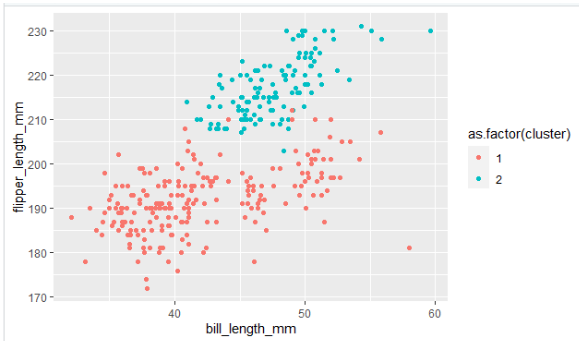


Fig 17: flipper length vs bill length

Observations:

- Shorter flippers and shorter bill length were mostly assigned to cluster 1.
- Long flippers and long bills were mostly assigned to cluster 2.
- Short flippers and long bills were mostly assigned to cluster 1.

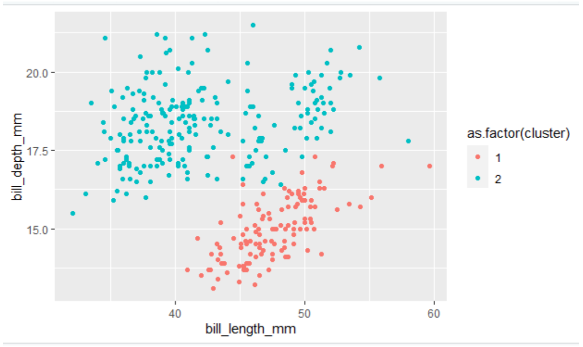


Fig 18: bill depth vs bill length

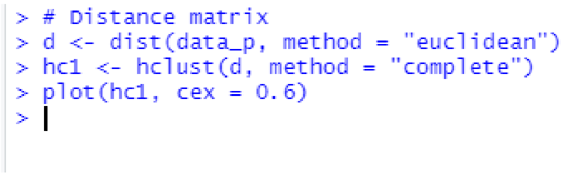
Observations:

- Deep bills and short bills were mostly assigned to cluster 2.
- Shallow bills and long bills were mostly assigned to cluster 1.

5. HIERARCHICAL CLUSTERING.

This starts by placing each observation in its on cluster and then it merges clusters by looking at the differences between adjacent points. It gives the user an idea of how clusters are like each other.

Unlike k-means, for hierarchical clustering, we supply a distance matrix (a matrix of the distance between every point in our data frame to every other point).



```
> # Distance matrix
> d <- dist(data_p, method = "euclidean")
> hc1 <- hclust(d, method = "complete")
> plot(hc1, cex = 0.6)
> 
```

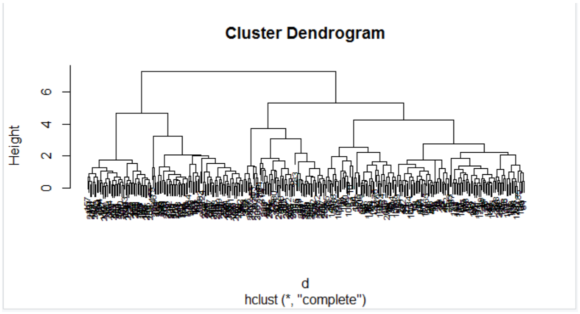
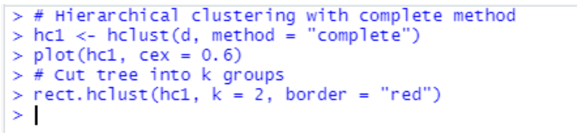


Fig 19: Cluster dendrogram

From the dendrogram, it can be observed that every observation is labelled at the bottom and each one is clustered with its most similar neighbours. As we move higher up the tree, we can notice we've got larger clusters. This is useful if we are interested in some form of hierarchical structure to your dataset.

Hierarchical clustering makes use of two methods

Complete and single method.



```
> # Hierarchical clustering with complete method
> hc1 <- hclust(d, method = "complete")
> plot(hc1, cex = 0.6)
> # Cut tree into k groups
> rect.hclust(hc1, k = 2, border = "red")
> 
```

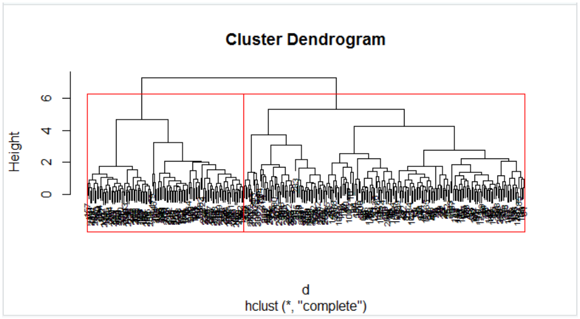
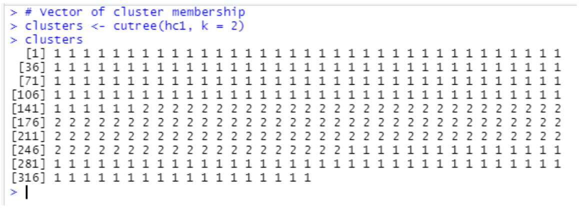


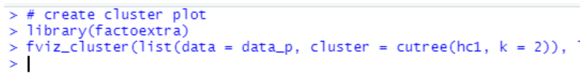
Fig 20: Cluster Dendrogram where k=2

To get the vector of cluster membership and to create the cluster plot, we use the cutree function.



```
> # Vector of cluster membership
> clusters <- cutree(hc1, k = 2)
> clusters
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [71] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[106] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[141] 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[176] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[211] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[246] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[281] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[316] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
> 
```

Fig 21: Vector of cluster membership



```
> # create cluster plot
> library(factoextra)
> fviz_cluster(list(data = data_p, cluster = cutree(hc1, k = 2)), `
> |
```

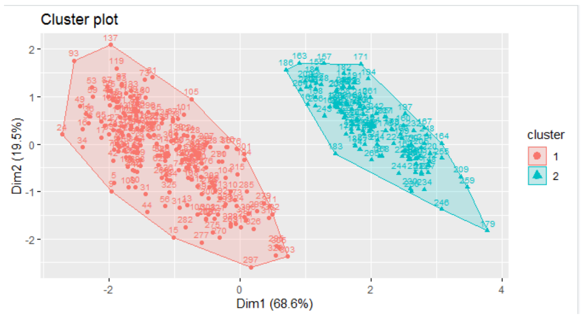
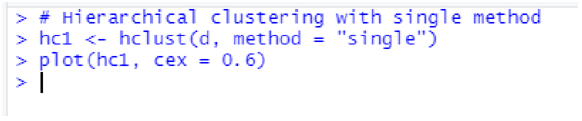


Fig 22: Cluster plot for the complete method

Let us now look at the visualization of the single method.



```
> # Hierarchical clustering with single method
> hc1 <- hclust(d, method = "single")
> plot(hc1, cex = 0.6)
> |
```

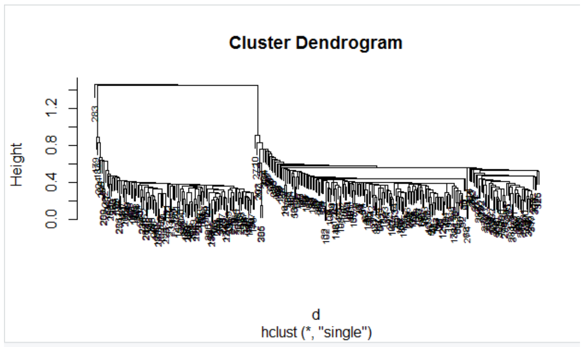
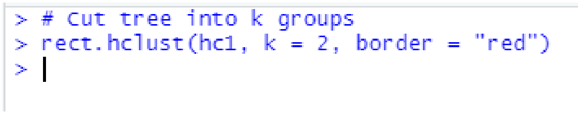


Fig 22: Single method Cluster dendrogram

Splitting it into clusters.



```
> # Cut tree into k groups
> rect.hclust(hc1, k = 2, border = "red")
> |
```

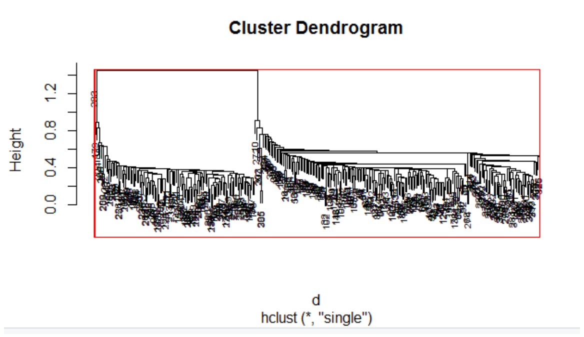
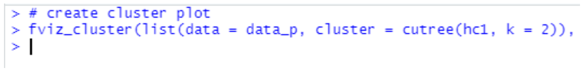


Fig 23: Single method Cluster Dendrogram where k=2

Creating the cluster plot.



```
> # create cluster plot
> fviz_cluster(list(data = data_p, cluster = cutree(hc1, k = 2)),
> |
```

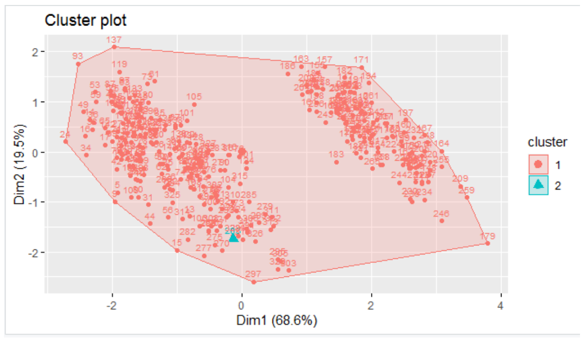


Fig 24: Cluster plot for the single method

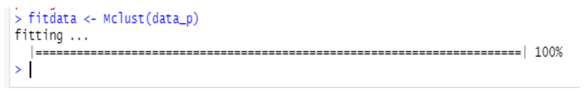
6. MODEL-BASED CLUSTERING.

There are other sophisticated algorithms for instance, this current one, the model-based clustering. This model-based clustering fits statistical models to the data to look for different distributions in the parameter space and based on those distributions, finds whether observation in our dataset are likely to have come from the distributions that are mapped in the data.

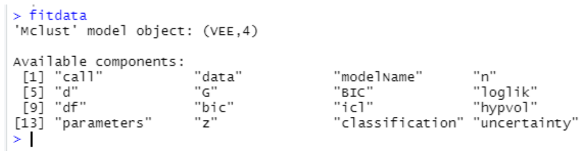
To carry out model-based clustering, we use the package called mclust. Mclust fits several different models to the data and it not only picks the statistical model that fits the data, it also fits the version of that model with the number of clusters that it thinks best explains the data.

When plotting the chart for the fitdata model, it gives us a set of options to select from.

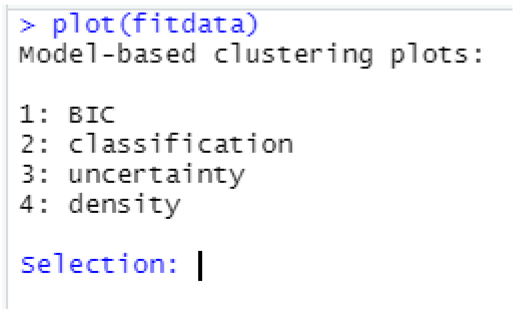
BIC, Classification, uncertainty, and density.



```
> fitdata <- Mclust(data_p)
fitting ...
  |========================================================================| 100%
> |
```



```
> fitdata
'Mclust' model object: (VEE,4)

Available components:
 [1] "call"           "data"           "modelName"      "n"
 [5] "d"              "G"              "BIC"            "loglik"
 [9] "df"             "bic"            "icl"            "hypvol"
[13] "parameters"     "z"              "classification" "uncertainty"
> |
```



```
> plot(fitdata)
Model-based clustering plots:

1: BIC
2: classification
3: uncertainty
4: density

Selection: |
```

BIC:

x-axis is the number of clusters the algorithm has tried to identify.

y-axis is the BIC

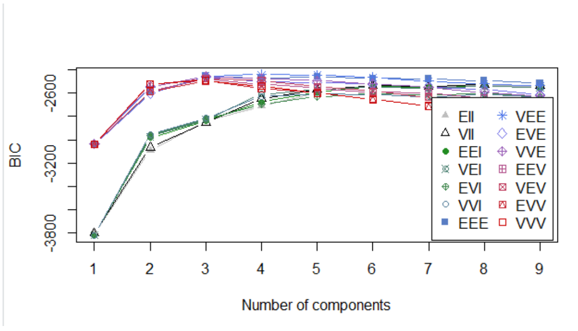


Fig 23: BIC chart

This is basically a measurement of how much variability is explained in the data. The higher the value, the better the model.

Those lines represent several different statistical models that the software has fitted to my data. It looks at the model which performs the best that is, the one with the highest BIC which in this case is the VEE and EEE.

Classification:

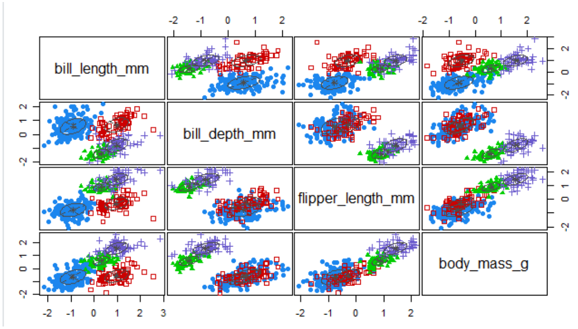


Fig 24: Classification plot

We can see the algorithm has classified these into 4 different clusters unlike in the k-means or hierarchical clustering where the user picks the number of clusters.

Uncertainty:

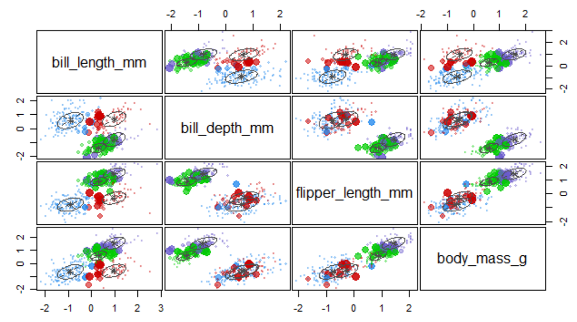


Fig 24: Uncertainty plot.

This shows us our own subject and cluster membership but draws in large observations for which cluster membership is uncertain. It is quite normal to have individual observations that does not fit particularly into any cluster. But in this case, we have several observations that does not fit into any cluster.

Density:

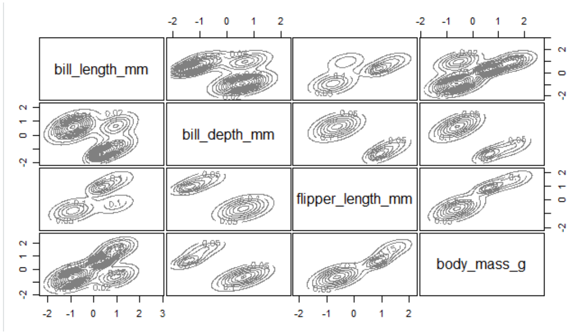


Fig 25: Density plot

This is a representation of the final model that was used to model the data.

7. CONCLUSION:

Unlike k-means and hierarchical clustering, this model-based clustering gives no flexibility to the data scientist as to the number of clusters that are identified or to how the group membership is assigned. In a way, it is a good thing as it removes the subjectivity of it. It doesn't allow you to tune or select the parameters.