Data Science and Statistics in Research: unlocking the power of your data

Session 3.5: Clustering

Introduction

In this session we will carry out some hierarchical and k-means clustering on sample data in `R`.

Data

To demonstrate how to perform hierarchical and k-means clustering in `R` we will use the `iris` and `mtcars` datasets. We have seen the `mtcars` dataset previously.

The `iris` dataset consists of 150 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). We load this dataset, using the `data()` function.

```
# Loading iris dataset
data(iris)
```

Make sure you are familiar with the contents of this dataset before continuing on with the rest of this practical, by typing `?iris` into `R`.

Clustering

Clustering is a statistical technique which creates groupings within data. Objects within a cluster are more similar than those in other clusters. We will investigate two methods of clustering today:

- Hierarchical clustering, and
- K-means clustering.

Hierarchical clustering

A method of clustering, known as hierarchical clustering, is available through the `hclust` function in `R`. This method starts out by putting each observation into its own separate cluster. It examines all the distances between all the observations and pairs together the two closest ones to form a new cluster. This process is repeated until there is one single cluster.

For these methods, the dendrogram is the main graphical tool for gaining an insight into a cluster solution. When you use `hclust` to perform a cluster analysis, you can see the dendogram by passing the result of the clustering to the `plot` function. We will see some examples of this during the session.

Example 1: `mtcars` dataset

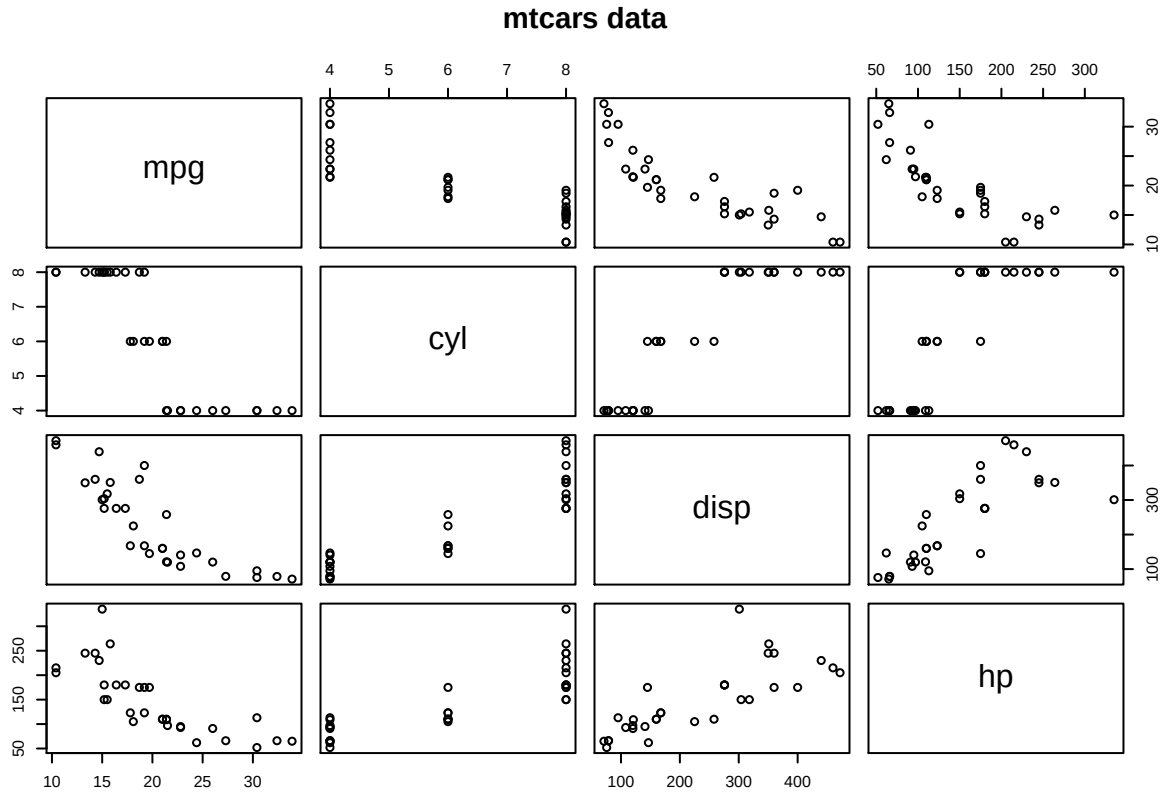
We'll investigate this clustering approach for the `mtcars` dataset, which consists of fuel consumption and other aspects of automobile design and performance for 32 cars from the 1973-74 Motor Trend US magazine. We load this dataset, using the `data()` function.

```
# Loading mtcars dataset
data(mtcars)
```

Make sure you are familiar with the contents of this dataset before continuing on with the rest of this practical, by typing `?mtcars` into `R`.

Let's first look at the `pairs` function to produce a matrix of scatterplots. We will choose the first four variables.

```
pairs(mtcars[,1:4], main = "mtcars data")
```



This shows us the correlations between pairs, which may help us identify clusters.

Distance matrix

To find out the dissimilarity between two cars in the `mtcars` dataset, say Honda Civic and Camaro Z28, we can calculate the distance between them with the `dist` function:

```
x <- mtcars["Honda Civic",] #ext
y <- mtcars["Camaro Z28",]
dist(rbind(x, y))
```

```
            Honda Civic
Camaro Z28    335.8883
```

This tells us that there is a distance between these two cars of 335.8883.

We can repeat this, comparing the Camaro Z28 and the Pontiac Firebird cars in the dataset.

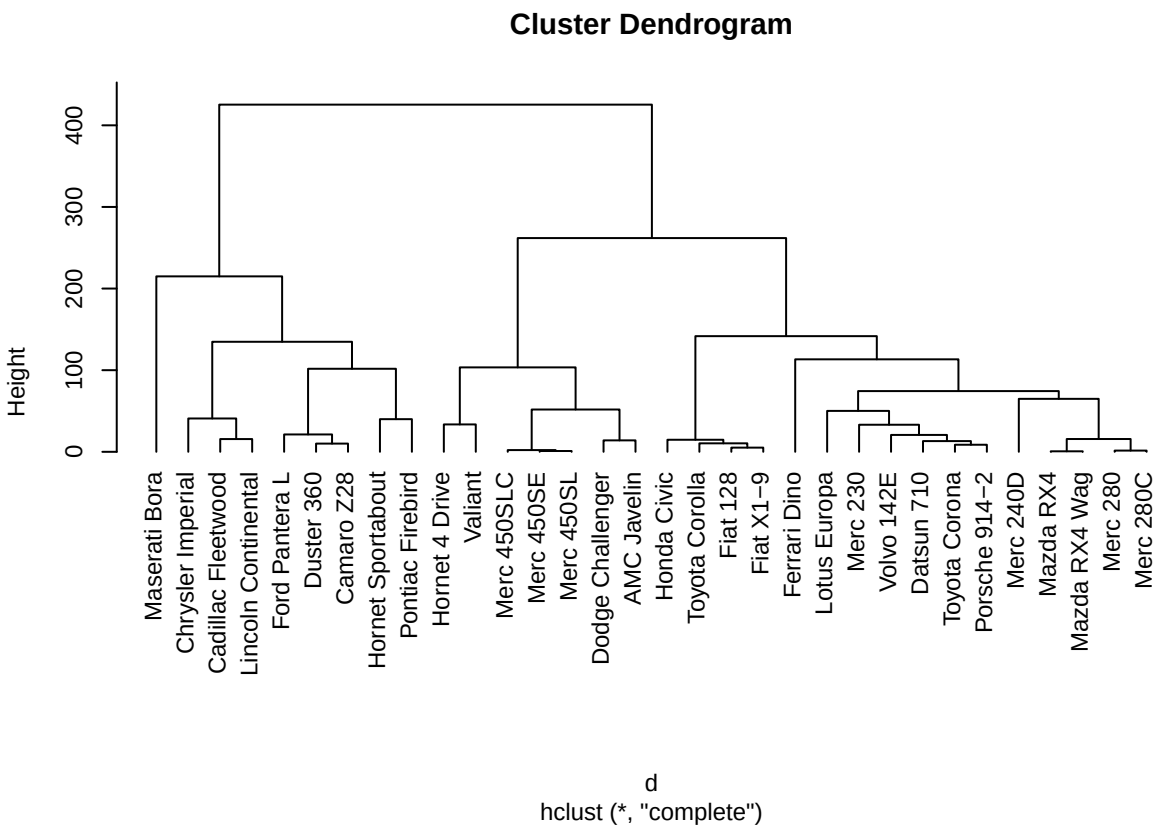
```
z <- mtcars["Pontiac Firebird",]
dist(rbind(y, z))
                 Camaro Z28
Pontiac Firebird   86.26658
```

This time, we obtain a distance between the two cars of 86.26658. As the distance between Camaro Z28 and Pontiac Firebird (86.26658) is smaller than the distance between Camaro Z28 and Honda Civic (335.8883), we conclude that Camaro Z28 is more similar to Pontiac Firebird than to Honda Civic.a

The distance matrix can also be computed for the full set of cars in `mtcars` as:

```
d <- dist(as.matrix(mtcars))
```

```
# prepare hierarchical cluster
hc = hclust(d)
# very simple dendrogram
plot(hc, hang=-1) #hang places the labels all at the same level
```



How many clusters do you observe here?

Example 2: `iris` dataset

The `iris` dataset (included with `R`) contains four measurements for 150 flowers representing three species of iris (Iris setosa, versicolor and virginica).

After loading the dataset, we can inspect the data in R like this:

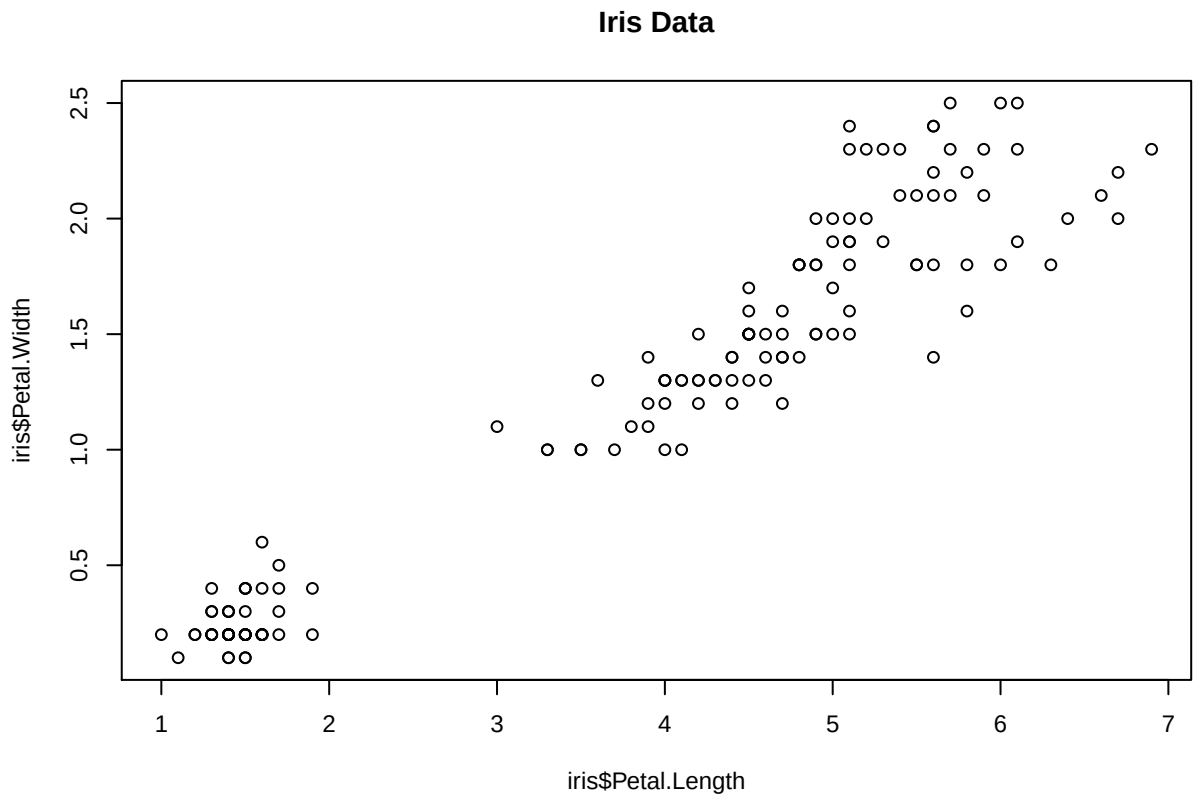
```
head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

This shows us the first five rows of the dataset. We can see the column names of this dataset.

```
colnames(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
[5] "Species"
```

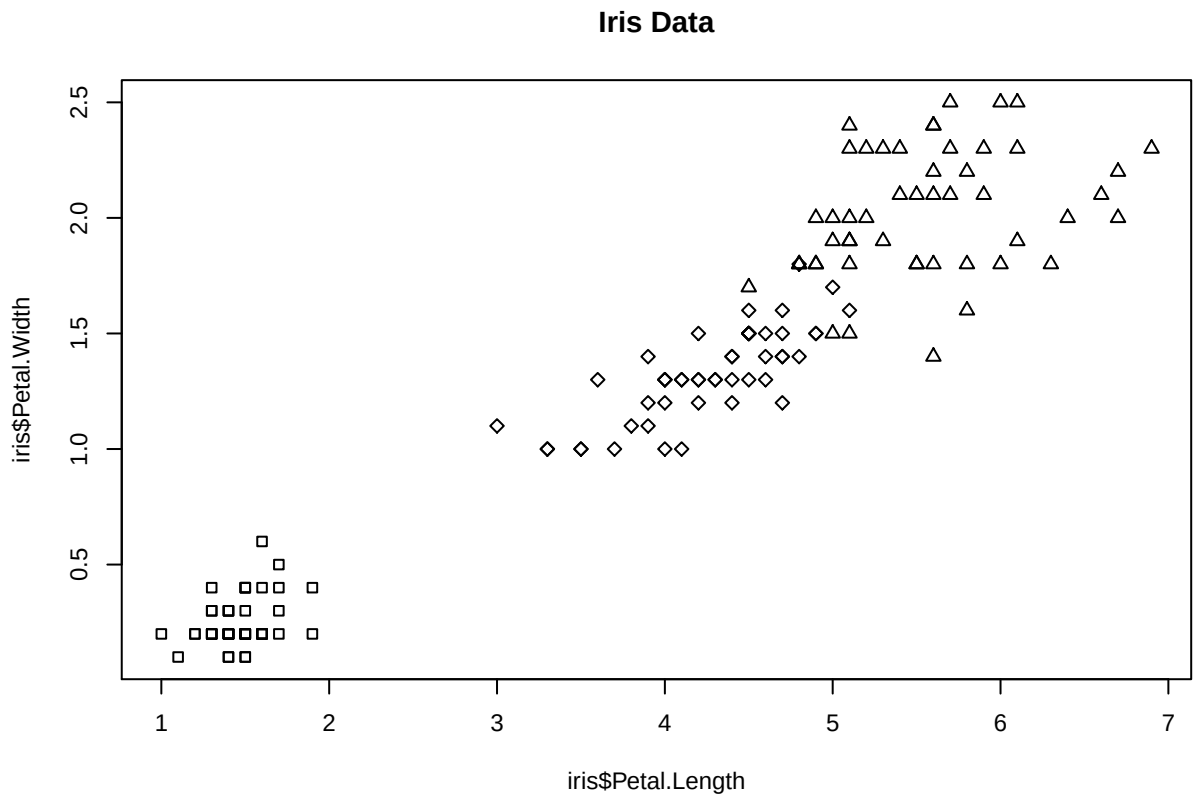
Let's do a simple scatter plot of petal length vs. petal width:

```
plot(iris$Petal.Length, iris$Petal.Width, main="Iris Data")
```



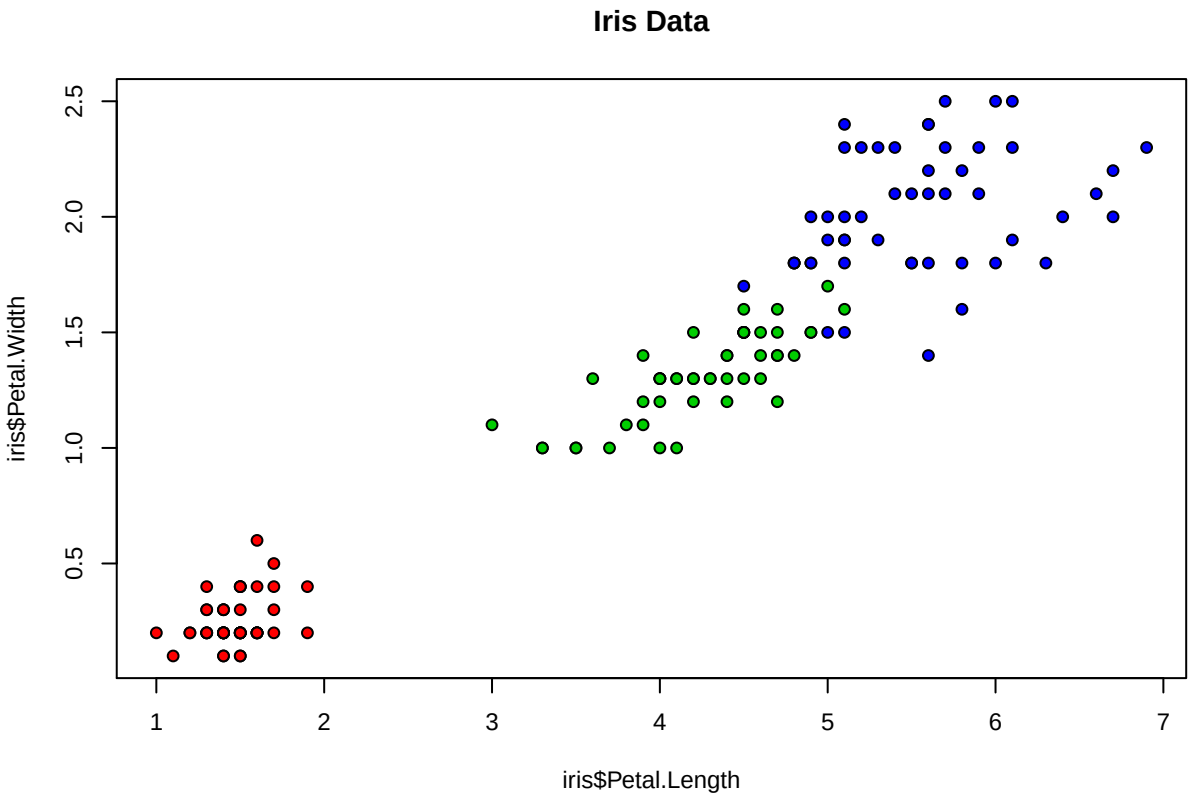
We are looking for patterns or clusters in this data. It may be helpful to mark or colour in the points by species.

```
plot(iris$Petal.Length, iris$Petal.Width, pch=c(22,23,24)[unclass(iris$Species)], main="Iris Data")
```



It looks as if we may have some pattern or cluster emerging here - it appears that the three species have different petal sizes. We can check this further by adding some colours to our plot.

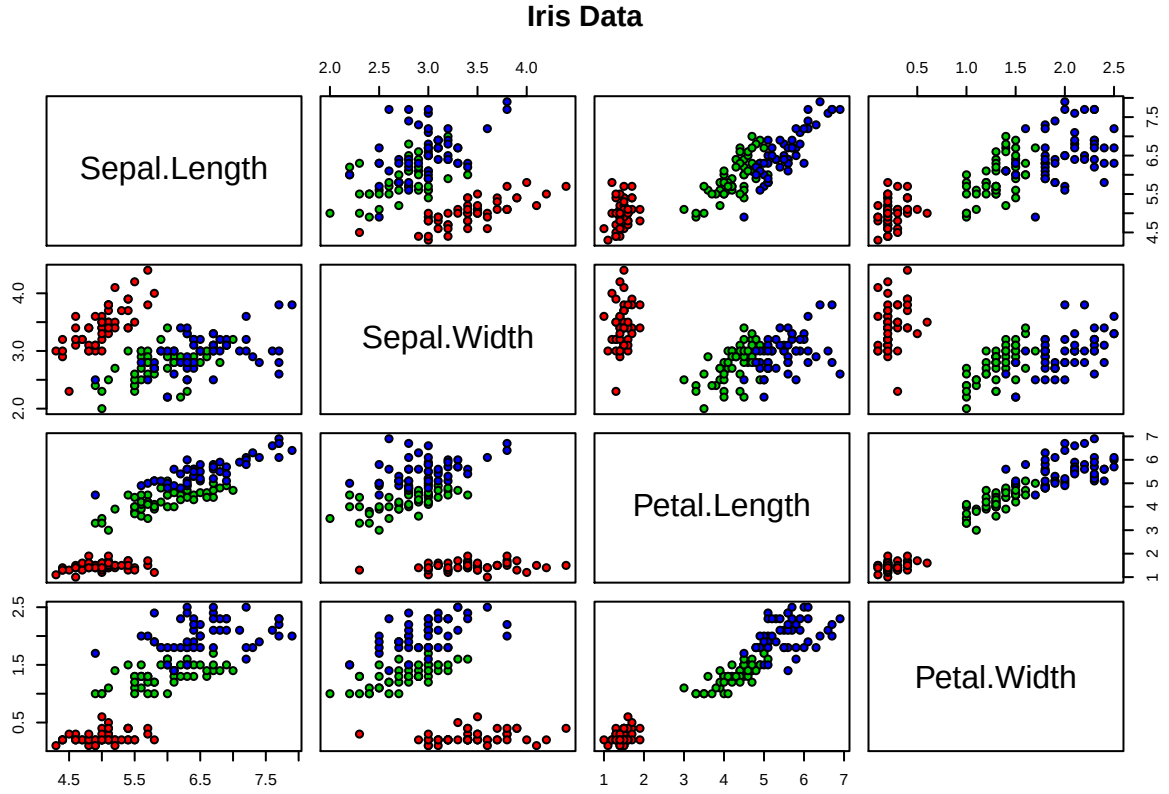
```
plot(iris$Petal.Length, iris$Petal.Width, pch=21, bg=c("red","green3","blue")[unclass(iris$Species)],
     main="Iris Data")
```



Using different colours it seems clear that the three species have very different petal sizes.

How do the other variables behave? We could generate each plot individually, but there is quicker way, using the 'pairs' command on the first four columns to produce a matrix of scatterplots:

```
pairs(iris[1:4], main = "Iris Data", pch = 21, bg = c("red", "green3", "blue")[unclass(iris$Species)])
```



This command shows us the correlations between pairs, which may help us further identify clusters.

It appears that most of the variables could be used to predict the species - except that using the sepal length and width alone would make distinguishing between *Iris versicolor* and *virginica* (green and blue) difficult.

Distance matrix

The measure of distance is an important tool in statistical analysis. It quantifies dissimilarity between sample data.

If we apply the distance computation between all possible pairs of flowers in `iris`, and arrange the result into a symmetric matrix, with the element at the i-th row and j-th column being the distance between the i-th and j-th iris in the the data set, we will have the so-called the distance matrix. It can be computed for the full set of cars in `iris` as:

```
d1 <- dist(as.matrix(iris))
```

Now we wish to use the hierarchical clustering method to identify patterns in the `iris` dataset. Hierarchical clustering is typically used when the number of points are not too high, so we sample 40 data points from the `iris` dataset here.

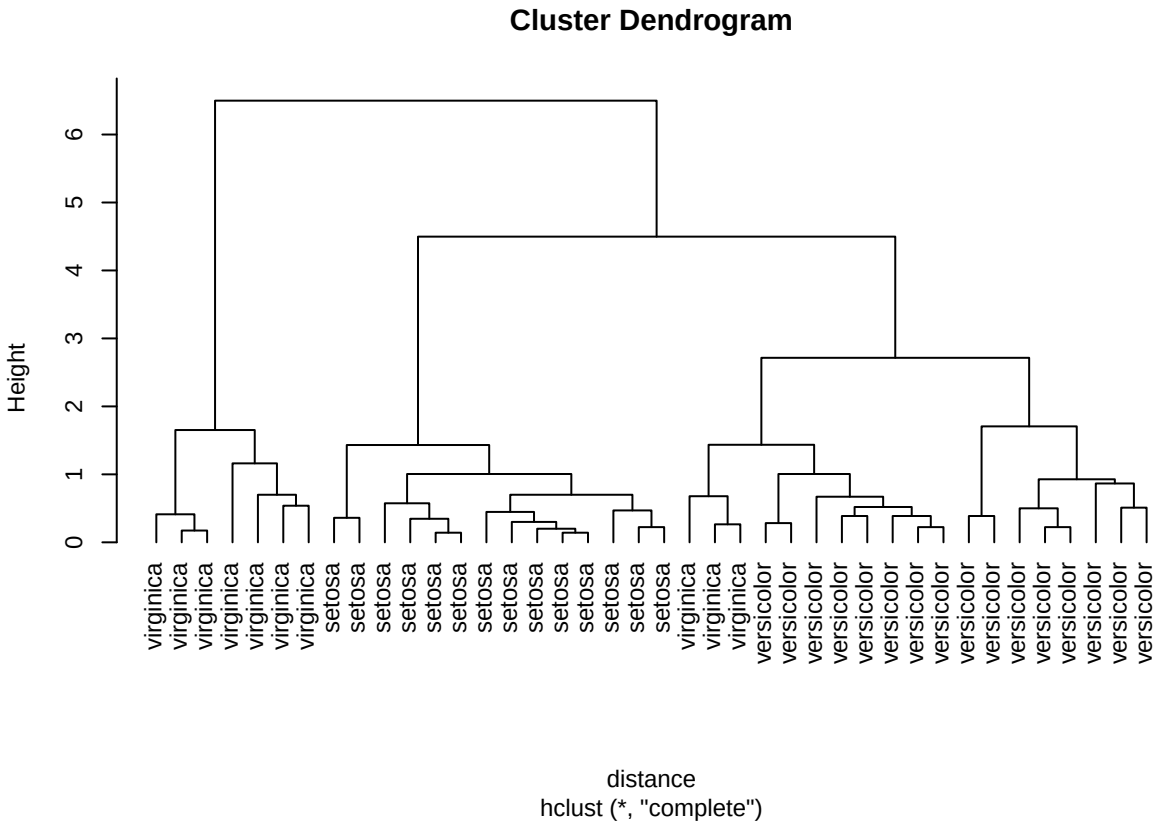
```
#sampling 40 points
sampleiris <- iris[sample(1:150, 40),]

#calculating our distance matrix
distance <- dist(sampleiris[,-5])

#preparing the hierarchical cluster
```

```
cluster <- hclust(distance)

#plotting our dendrogram, with hang=-1 to have all labels at same level
plot(cluster, hang=-1, label=sampleiris$Species)
```



We can identify three main clusters here, confirming our suspicions from our scatter plots. Note that this does not cluster perfectly, ase see some overlap of *virginica* and *versicolor*.

We noted earlier that `hclust` is slow and computationally expensive for big data, which was why we sampled from the `iris` dataset for convenience. However, there is another potential approach - k-means clustering.

K-means clustering

In this approach, observations are divided into k groups and reshuffled to form the most cohesive clusters possible according to a given criterion.

Unlike the hierarchical clustering, K-means require that we specify the number of clusters that will be formed in advance.

We implement this method in R using the `kmeans` function. ##Example 1: `iris` dataset We'll return to the `iris` dataset that we worked with earlier.

In this we assign the data from column 1-4 (features) to variable x, and the class (species) to variable y.

```
x = iris[,-5]
y = iris$Species
```

We create a kmeans model with the `kmeans` function. Note that we must always specify the number of clusters. Here, we specify 3 as we know that `iris` data has 3 classes.

```
kc <- kmeans(x,3)
kc
K-means clustering with 3 clusters of sizes 62, 50, 38

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.901613    2.748387     4.393548    1.433871
2     5.006000    3.428000     1.462000    0.246000
3     6.850000    3.073684     5.742105    2.071053

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
 [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [71] 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 3 3 3
[106] 3 1 3 3 3 3 3 3 1 1 3 3 3 3 1 3 1 3 1 3 3 1 1 3 3 3 3 3 1 3 3 3 3 1 3
[141] 3 3 1 3 3 3 1 3 3 1

Within cluster sum of squares by cluster:
[1] 39.82097 15.15100 23.87947
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
```

After we know the result, we can check for errors or missing data. We can compare the clustering result with the species/classes iris data using the below command.

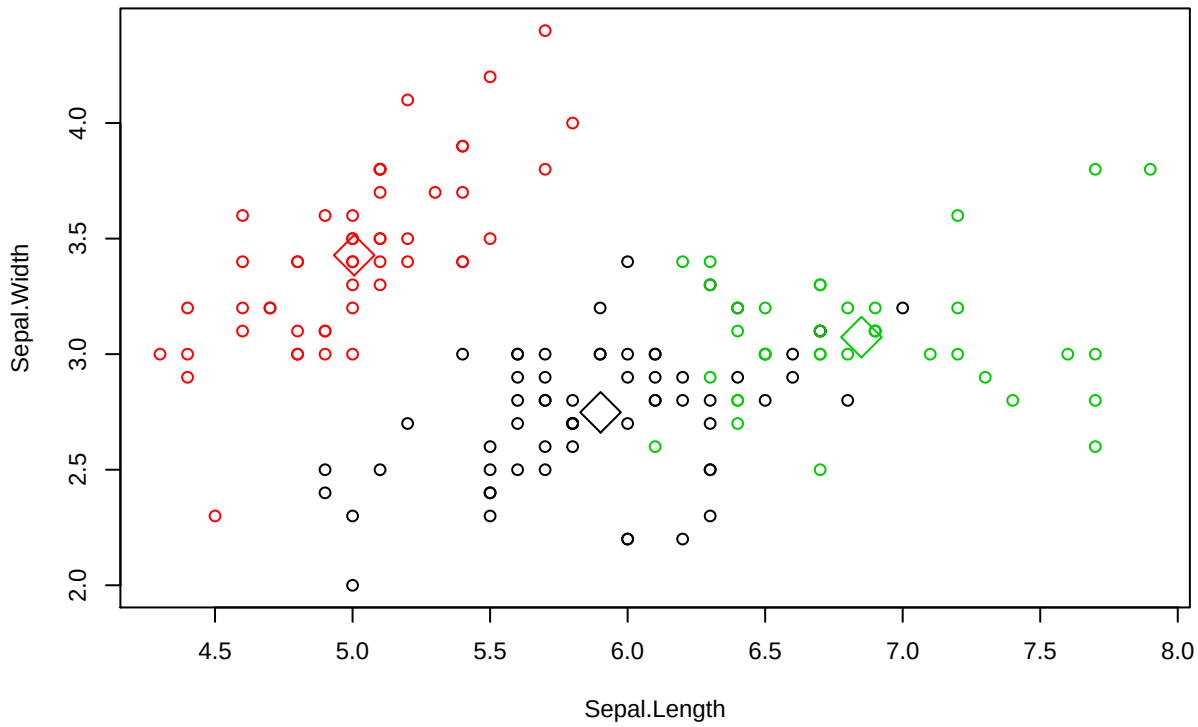
```
table(y,kc$cluster)

y             1  2  3
  setosa      0 50  0
  versicolor 48  0  2
  virginica  14  0 36
```

As we can see, the data belonging to the setosa species got grouped into cluster 3, versicolor into cluster 2, and virginica into cluster 1. The algorithm wrongly classified two data points belonging to versicolor and fourteen data points belonging to virginica.

We can plot this simply through the `plot` command, and colour by cluster.

```
plot(x[c("Sepal.Length", "Sepal.Width")], col=kc$cluster)
points(kc$centers[,c("Sepal.Length", "Sepal.Width")], col=1:3, pch=23, cex=3)
```



We can see the clusters clearly here. Note again, that this does not cluster perfectly. We see some overlap of *virginica* and *versicolor*, as we did in the hierarchical clustering approach.

Activity

*Repeat this process for the `mtcars` dataset. How many clusters do you choose in this case, and why?