Lab 2 Explore Iris Dataset

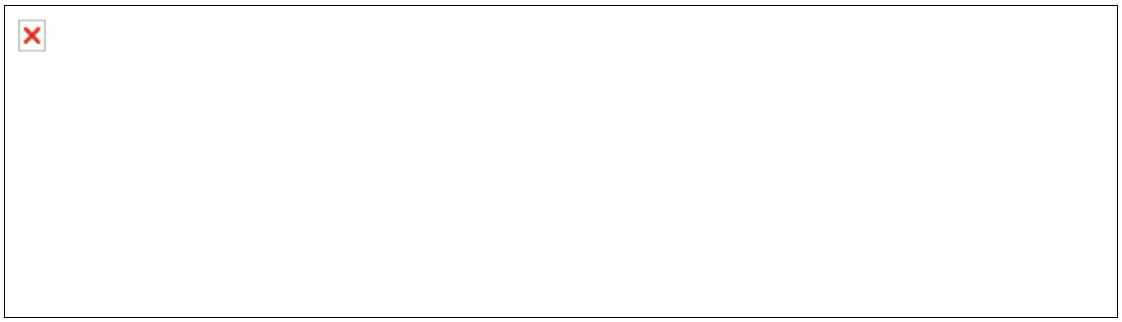
Objective of this lab

1. Install Rattle
2. Exploration and basic visualization using R/Rattle
3. Start with Case #1

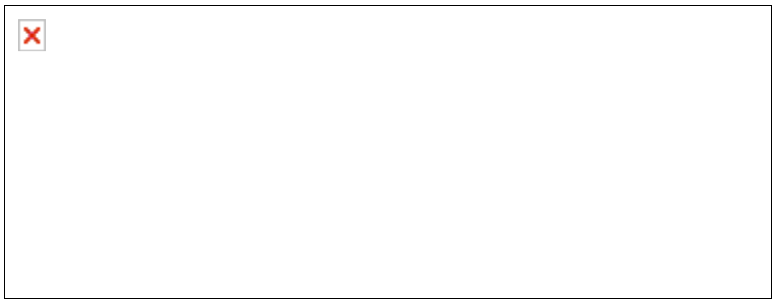
Iris dataset - Introduction

The dataset consists of 50 samples from each of three species of Iris flowers (Iris setosa, Iris virginicaand Iris versicolor). Four features(variables) were measured from each sample, they are the **length** and the **width** of sepal and petal, in centimeters. It is introduced by Sir Ronald Fisher in 1936.

3 Species



Flower Parts



Explore the *iris* Dataset with R

Load Data

The *iris* flower data set is included in R. It is a data frame with 150 cases (rows) and 5 variables (columns) named Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, and Species.

First, load iris data to the current workspace

```
data(iris)
```

What is in the dataset?

Check dimensionality, the dataset has 150 rows(observations) and 5 columns (variables)

```
dim(iris)
```

```
## [1] 150   5
```

Another way to get the dim is to use ncol or nrow:

```
ncol(iris)
```

```
## [1] 5
```

```
nrow(iris)
```

```
## [1] 150
```

Variable names or column names

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

You can also use this command

```
colnames(iris)
```

Structure of the dataframe, note that the difference between *num* and *Factor*

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1
```

Subsetting the dataset I: Numerical Index and Variable Names

Get the first 5 rows/obs

```
iris[1:5, ]
```

```
##   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
```

Get Sepal.Length of the first 10 rows

```
iris[1:10, "Sepal.Length"]
```

```
##  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
```

or alternatively

```
iris$Sepal.Length[1:10]
```

```
##  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
```

or you can use the column index instead of variable names. Since Sepal.Length is the first column:

```
iris[1:10, 1]
```

```
##  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
```

Subsetting the dataset II: Logical Conditions

There are 3 ways to get subset of data that satisfy certain logical conditions. These kind of operations are called filtering in Excel. Knowing any one of these well is enough. Do not worry about memorizing the syntax, you can always look them up.

Suppose we want to get the **subset of data that has Sepal.Length > 5 and Sepal.Width > 4**.

Use subset function

```
subset(x = iris, subset = Sepal.Length > 5 & Sepal.Width > 4)
```

You can omit the x = and sebset = part

```
subset(iris, Sepal.Length > 5 & Sepal.Width > 4)
```

Use logical vectors

```
iris[which(iris$Sepal.Length > 5 & iris$Sepal.Width > 4), ]
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 16          5.7         4.4          1.5         0.4  setosa
## 33          5.2         4.1          1.5         0.1  setosa
## 34          5.5         4.2          1.4         0.2  setosa
```

Use SQL statement

```
install.packages("sqldf")
library(sqldf)
sqldf("select * from iris where Sepal_Length > 5 and Sepal_Width > 4
```

Subsetting the dataset III: Random Sample

The following code random sample (without replacement) 90% of the original dataset and assgin them to a new variable *iris_sample*.

```
iris_sample = iris[sample(nrow(iris), nrow(iris) * 0.9), ]
```

Summary Statistics and Plots

Summary statistics of every variable

```
summary(iris)
```

```
##   Sepal.Length   Sepal.Width    Petal.Length   Petal.Width
##  Min.   :4.30   Min.   :2.00   Min.   :1.00   Min.   :0.1
##  1st Qu.:5.10   1st Qu.:2.80   1st Qu.:1.60   1st Qu.:0.3
##  Median :5.80   Median :3.00   Median :4.35   Median :1.3
##  Mean   :5.84   Mean   :3.06   Mean   :3.76   Mean   :1.2
##  3rd Qu.:6.40   3rd Qu.:3.30   3rd Qu.:5.10   3rd Qu.:1.8
##  Max.   :7.90   Max.   :4.40   Max.   :6.90   Max.   :2.5
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

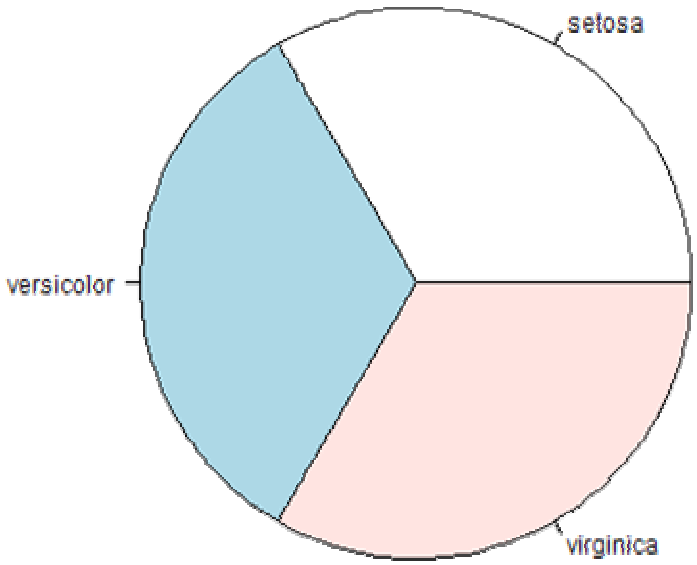
Frequency of species

```
table(iris$Species)
```

```
##
##     setosa versicolor  virginica
##         50         50         50
```

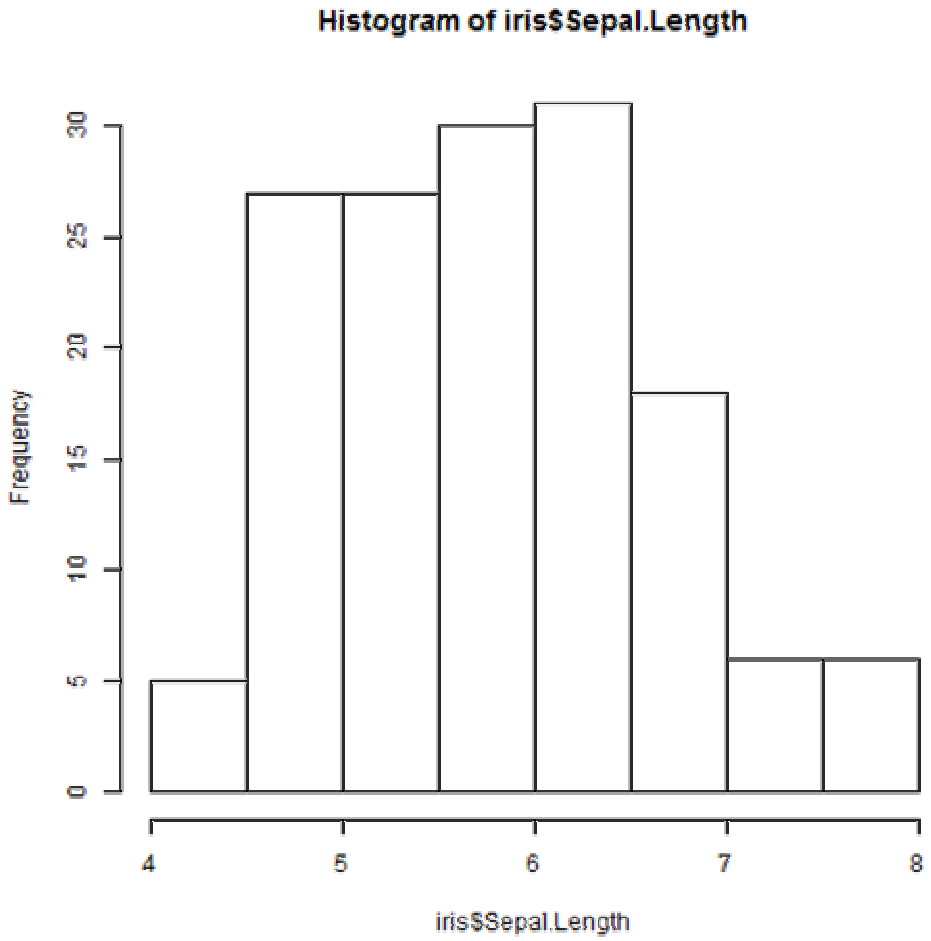
Pie Chart

```
pie(table(iris$Species))
```

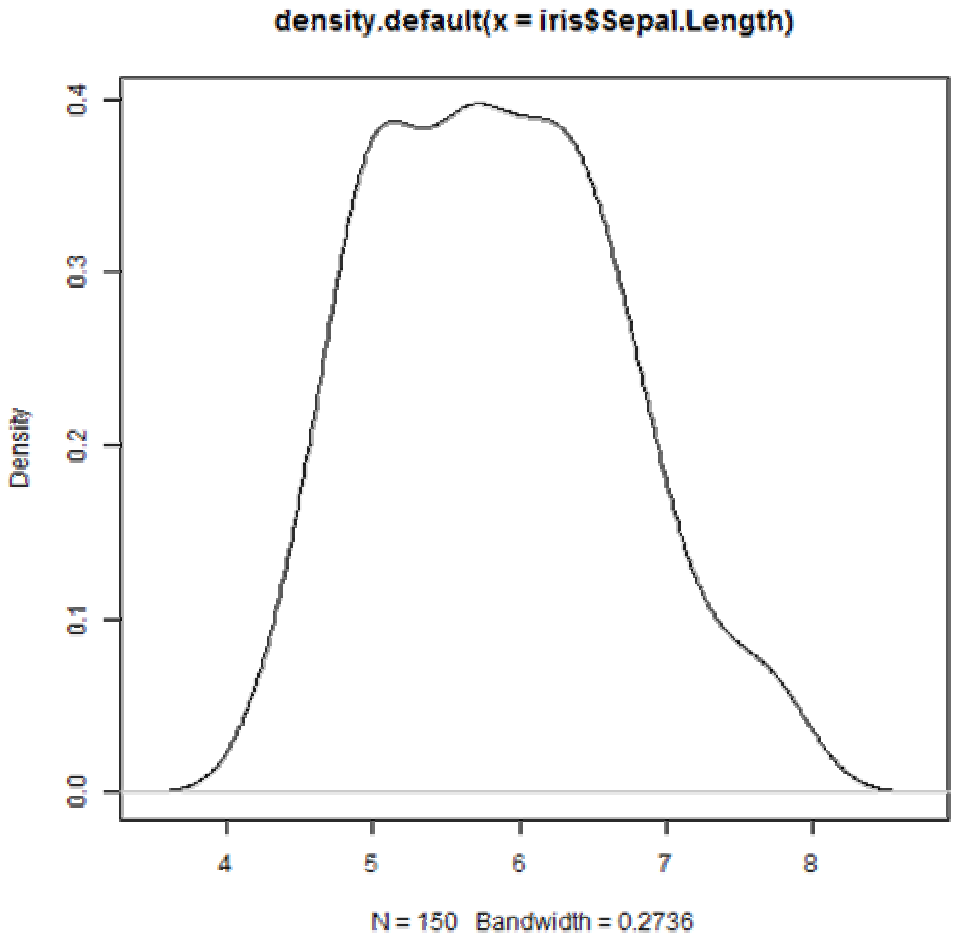


Plot histogram and density plot of Sepal.length

```
hist(iris$Sepal.Length)
```

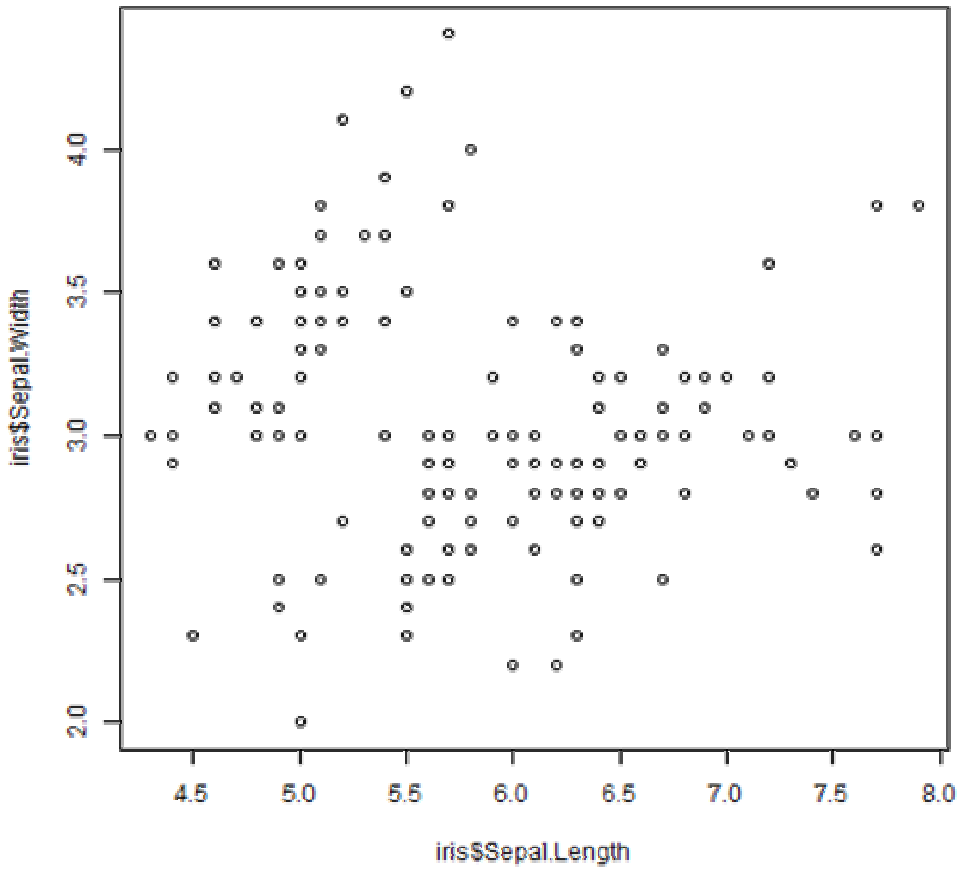



```
plot(density(iris$Sepal.Length))
```



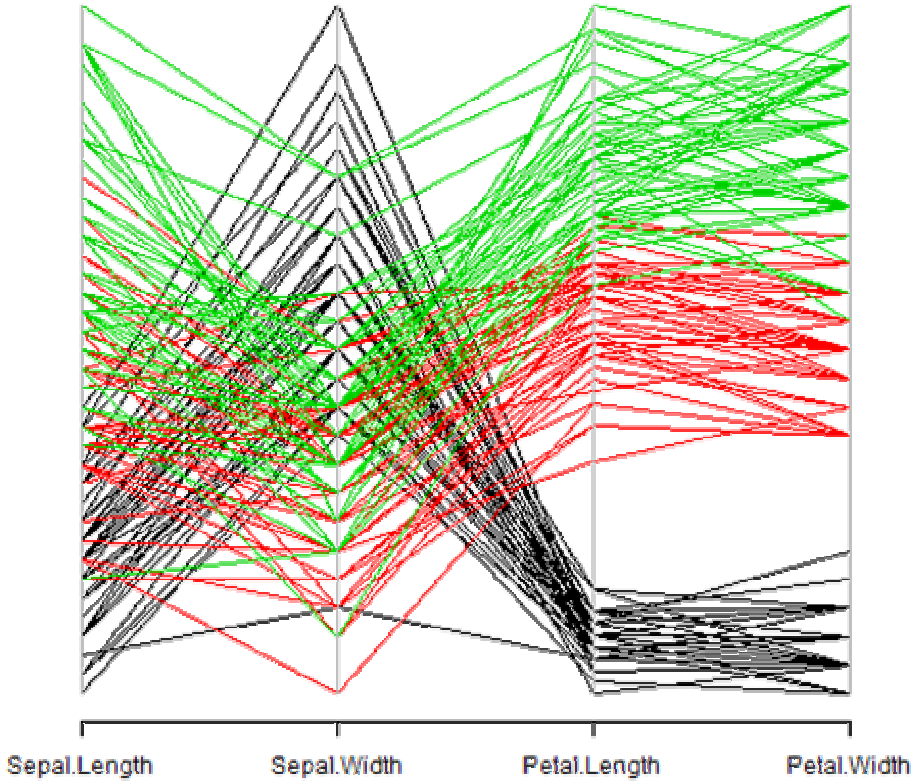
Scatter plot of length and width of sepals

```
plot(iris$Sepal.Length, iris$Sepal.Width)
```



Parallel Coordinates

```
library(MASS)
parcoord(iris[, 1:4], col = iris$Species)
```



Chernoff faces of the first 20 obs

```
install.packages("aplpack")
```

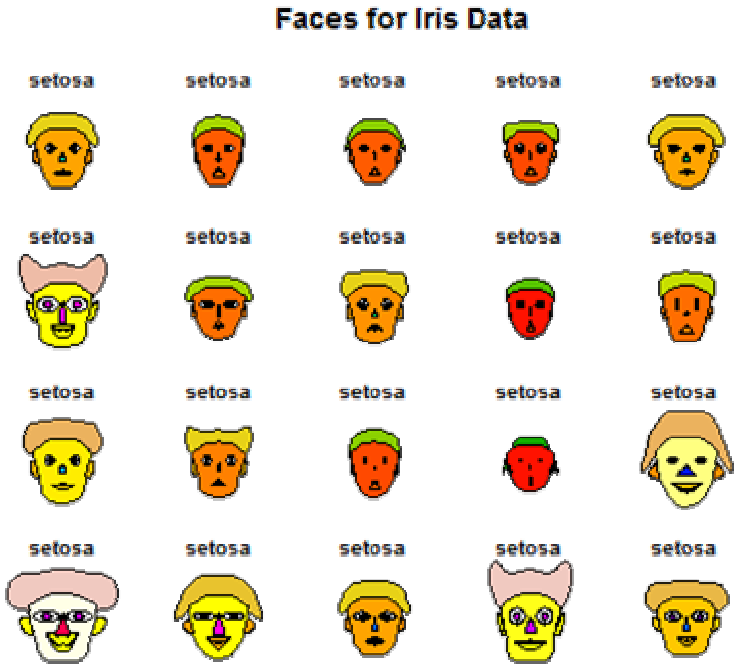
```
library(aplpack)
```

```
## Loading required package: tcltk
```

```
## Loading Tcl/Tk interface ...
```

```
## done
```

```
faces(iris[1:20, 1:4], main = "Faces for Iris Data", label = iris$Sp
```




```
## effect of variables:
##  modified item       Var
##  "height of face   " "Sepal.Length"
##  "width of face    " "Sepal.Width"
##  "structure of face" "Petal.Length"
##  "height of mouth  " "Petal.Width"
##  "width of mouth   " "Sepal.Length"
##  "smiling          " "Sepal.Width"
##  "height of eyes   " "Petal.Length"
##  "width of eyes    " "Petal.Width"
##  "height of hair   " "Sepal.Length"
##  "width of hair    " "Sepal.Width"
##  "style of hair    " "Petal.Length"
##  "height of nose   " "Petal.Width"
##  "width of nose    " "Sepal.Length"
##  "width of ear     " "Sepal.Width"
##  "height of ear    " "Petal.Length"
```

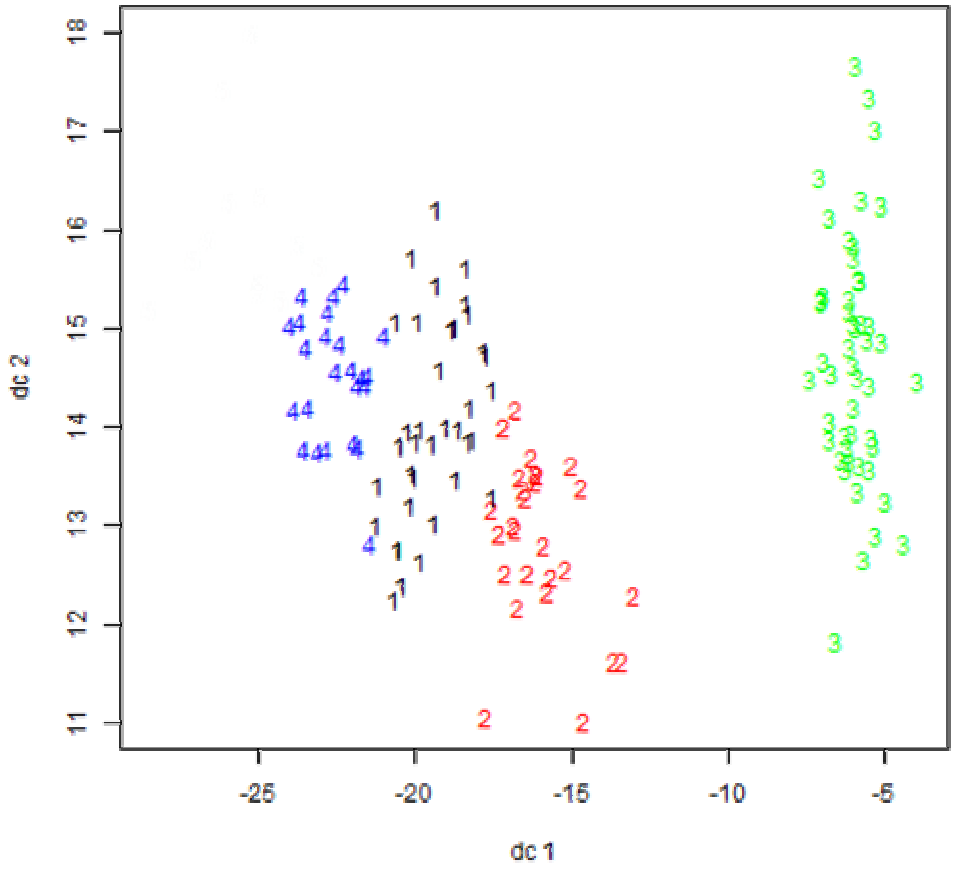
Cluster Analysis

K-means clustering

K-means clustering with 5 clusters, the 'fpc' package provides the 'plotcluster' function. You need to run *install.packages('fpc')* to install it first.

```
library(fpc)
```

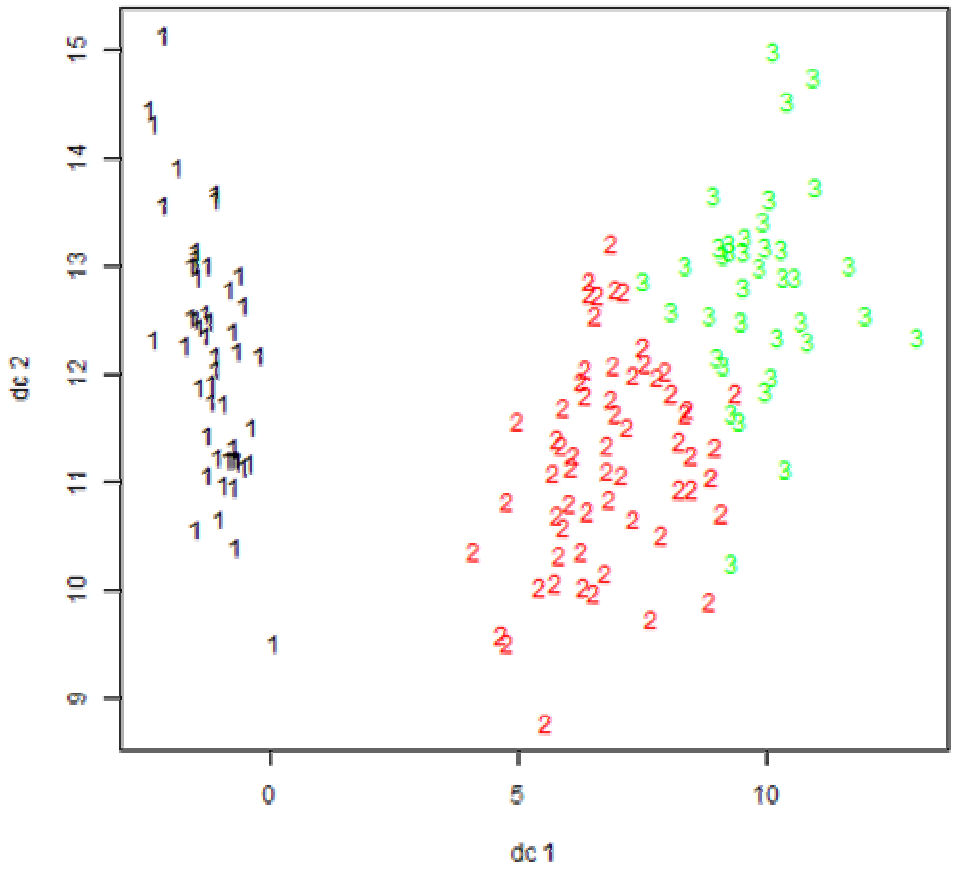
```
fit = kmeans(iris[, 1:4], 5)
plotcluster(iris[, 1:4], fit$cluster)
```



The first argument of the kmeans function is the dataset that you wish to cluster, that is the column 1-4 in the iris dataset, the last column is true category the observation so we do not include it in the analysis; the second argument 5 indicates that you want a 5-cluster solution. The result of the cluster analysis is then assigned to the variable fit, and the plotcluster function is used to visualize the result.

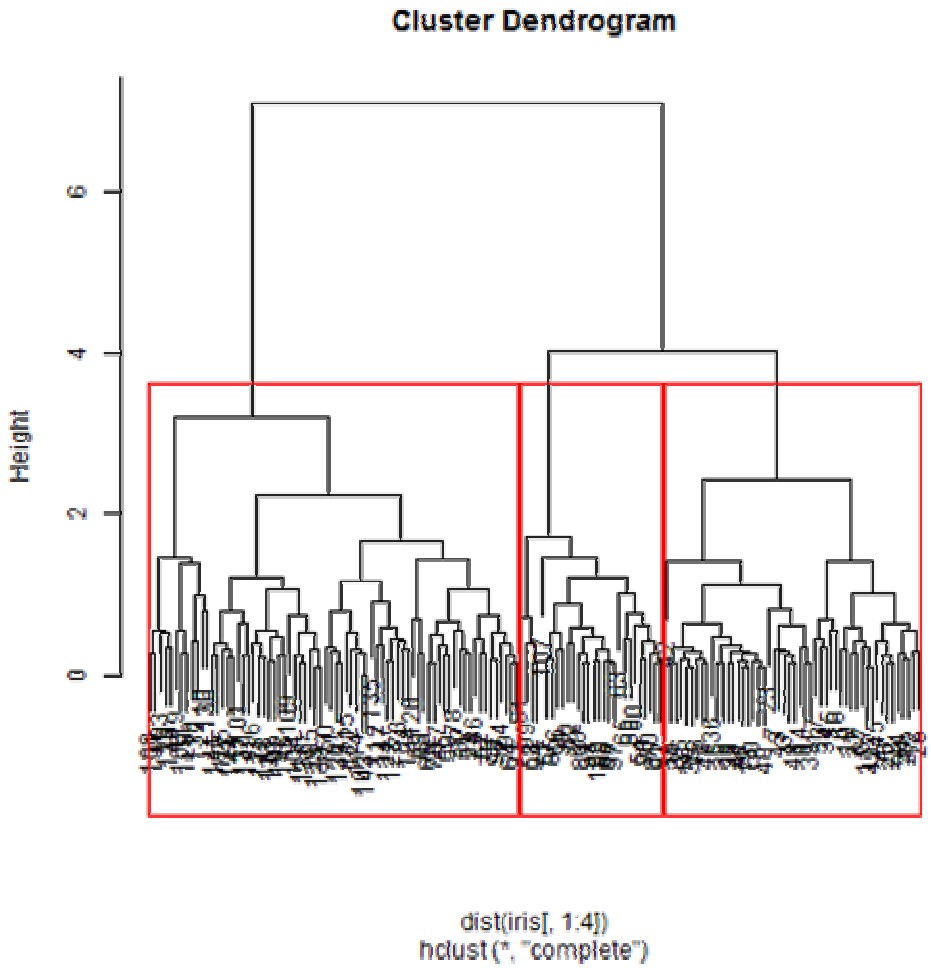
Do you think it is a good solution? Try it with 3 cluster.

```
kmeans_result = kmeans(iris[, 1:4], 3)
plotcluster(iris[, 1:4], kmeans_result$cluster)
```



Hierarchical clustering

```
hc_result = hclust(dist(iris[, 1:4]))
plot(hc_result)
# Cut Dendrogram into 3 Clusters
rect.hclust(hc_result, k = 3)
```



3 things happened in the first line. First dist(iris[, 1:4]) calculates the distance matrix between observations (how similar the observations are from each other judging from the 4 numerical variables). Then hclust takes the distance matrix as input and gives a hierarchical cluster solution. At last the solution is assigned to the variable hc_result. In hierarchical clustering you do not need to give the number of how many clusters you want, it depends on how you cut the dendrogram.

Analysis Using Rattle

You can follow this video:
http://www.youtube.com/watch?v=7P16fj0tqa4&hd=1