**Partition Clustering**

**K-means**

The k-means algorithm is very simple and basically consists of two steps. It is initialized by a random choice of cluster centres, e.g. a random selection of objects in the data set or random values within the range for each variable. The steps are:

1. Pick an initial set of K centroids (this can be random or any other means)
2. For each data point, assign it to the member of the closest centroid according to the given distance function
3. Adjust the centroid position as the mean of all its assigned member data points. Go back to (2) until the membership isn't change and centroid position is stable.
4. Output the centroids.

In this tutorial we want to show how to use K-means in R with Iris Data example. We can show the iris data with this command, just type "iris" for show the all data

head(iris)



Or we can use command "names" for show the label/column names

names(iris)



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

x = iris[,-5]

y = iris$Species

Create kmeans model with this command: (You need to put the number how many cluster you want, in this case I use 3 because we already now in iris data we have 3 classes)

kc <- kmeans(x,3)

type "kc" or kmeans model for show summary

kc



After we know the result, we need to know how many error and missing data, so we need to compare the clustering result with the species/classes iris data. We use table for comparison

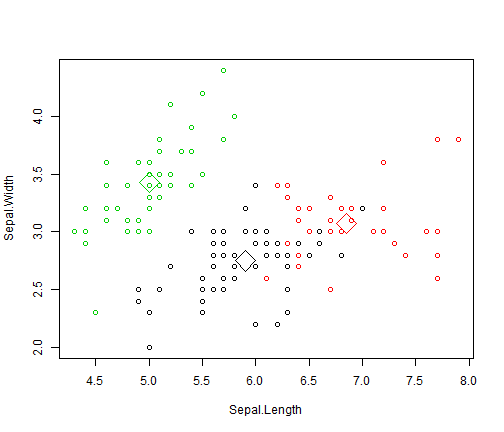
table(y,kc$cluster)



For plotting we can use plot function. In this case we plot the Sepal length as x-axis and Sepal Width as y-axis, you may choose different.

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)



**K-medoids**

library(cluster)

# Warning: package 'cluster' was built under R version 3.1.2, and contains pam function (k-medoids)

data(iris)

iris2 <- iris

str(iris2)

iris2$Species <- NULL

str(iris2)

pam.result <- pam(iris2, 3)

table(pam.result$clustering, iris$Species)

# setosa versicolor virginica

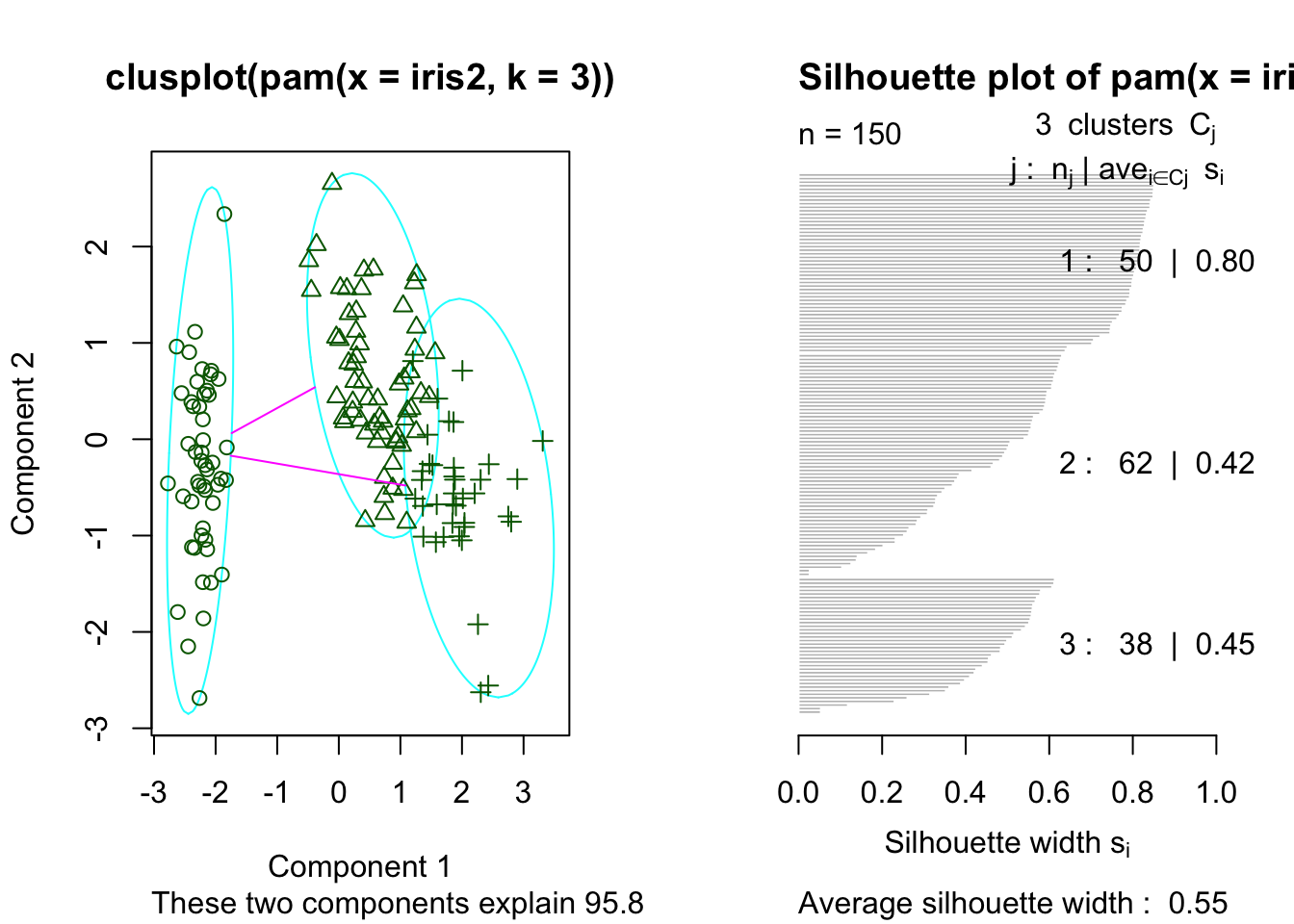
# 1 50 0 0

# 2 0 48 14

# 3 0 2 36

layout(matrix(c(1, 2), 1, 2)) # 2 graphs per page

plot(pam.result)



**WINE DATASET**

The wine data set contains the results of a chemical analysis of wines grown in a specific area of Italy. Three types of wine are represented in the 178 samples, with the results of 13 chemical analyses recorded for each sample. The Type variable has been transformed into a categorical variable.

# install.packages('rattle')

data(wine, package='rattle')

head(wine)

## Type Alcohol Malic Ash Alcalinity Magnesium Phenols Flavanoids

## 1 1 14.23 1.71 2.43 15.6 127 2.80 3.06

## 2 1 13.20 1.78 2.14 11.2 100 2.65 2.76

## 3 1 13.16 2.36 2.67 18.6 101 2.80 3.24

## 4 1 14.37 1.95 2.50 16.8 113 3.85 3.49

## 5 1 13.24 2.59 2.87 21.0 118 2.80 2.69

## 6 1 14.20 1.76 2.45 15.2 112 3.27 3.39

## Nonflavanoids Proanthocyanins Color Hue Dilution Proline

## 1 0.28 2.29 5.64 1.04 3.92 1065

## 2 0.26 1.28 4.38 1.05 3.40 1050

## 3 0.30 2.81 5.68 1.03 3.17 1185

## 4 0.24 2.18 7.80 0.86 3.45 1480

## 5 0.39 1.82 4.32 1.04 2.93 735

## 6 0.34 1.97 6.75 1.05 2.85 1450

**Explore and Pre-processing Data**

Let's see the structure of wine data set

str(wine)

#> 'data.frame':  178 obs. of  14 variables:

#> $ Type           : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...

#> $ Alcohol        : num  14.2 13.2 13.2 14.4 13.2 ...

#> $ Malic          : num  1.71 1.78 2.36 1.95 2.59 1.76 1.87 2.15 1.64 1.35 ...

#> $ Ash            : num  2.43 2.14 2.67 2.5 2.87 2.45 2.45 2.61 2.17 2.27 ...

#> $ Alcalinity     : num  15.6 11.2 18.6 16.8 21 15.2 14.6 17.6 14 16 ...

#> $ Magnesium      : int  127 100 101 113 118 112 96 121 97 98 ...

#> $ Phenols        : num  2.8 2.65 2.8 3.85 2.8 3.27 2.5 2.6 2.8 2.98 ...

#> $ Flavanoids     : num  3.06 2.76 3.24 3.49 2.69 3.39 2.52 2.51 2.98 3.15 ...

#> $ Nonflavanoids  : num  0.28 0.26 0.3 0.24 0.39 0.34 0.3 0.31 0.29 0.22 ...

#> $ Proanthocyanins: num  2.29 1.28 2.81 2.18 1.82 1.97 1.98 1.25 1.98 1.85 ...

#> $ Color          : num  5.64 4.38 5.68 7.8 4.32 6.75 5.25 5.05 5.2 7.22 ...

#> $ Hue            : num  1.04 1.05 1.03 0.86 1.04 1.05 1.02 1.06 1.08 1.01 ...

#> $ Dilution       : num  3.92 3.4 3.17 3.45 2.93 2.85 3.58 3.58 2.85 3.55 ...

#> $ Proline        : int  1065 1050 1185 1480 735 1450 1290 1295 1045 1045 ...

Wine data set contains 1 categorical variables (label) and 13 numerical variables. But these numerical variables is not scaled, we can use scale function for scaling and centering data and then assign it as training data.

data.train <- scale(wine[-1])

Data is already centered and scaled.

summary(data.train)

#>   Alcohol             Malic

#> Min.   :-2.42739   Min.   :-1.4290

#> 1st Qu.:-0.78603   1st Qu.:-0.6569

#> Median : 0.06083   Median :-0.4219

#> Mean   : 0.00000   Mean   : 0.0000

#> 3rd Qu.: 0.83378   3rd Qu.: 0.6679

#> Max.   : 2.25341   Max.   : 3.1004

#>      Ash             Alcalinity

#> Min.   :-3.66881   Min.   :-2.663505

#> 1st Qu.:-0.57051   1st Qu.:-0.687199

#> Median :-0.02375   Median : 0.001514

#> Mean   : 0.00000   Mean   : 0.000000

#> 3rd Qu.: 0.69615   3rd Qu.: 0.600395

#> Max.   : 3.14745   Max.   : 3.145637

#>   Magnesium          Phenols

#> Min.   :-2.0824   Min.   :-2.10132

#> 1st Qu.:-0.8221   1st Qu.:-0.88298

#> Median :-0.1219   Median : 0.09569

#> Mean   : 0.0000   Mean   : 0.00000

#> 3rd Qu.: 0.5082   3rd Qu.: 0.80672

#> Max.   : 4.3591   Max.   : 2.53237

#>   Flavanoids      Nonflavanoids

#> Min.   :-1.6912   Min.   :-1.8630

#> 1st Qu.:-0.8252   1st Qu.:-0.7381

#> Median : 0.1059   Median :-0.1756

#> Mean   : 0.0000   Mean   : 0.0000

#> 3rd Qu.: 0.8467   3rd Qu.: 0.6078

#> Max.   : 3.0542   Max.   : 2.3956

#> Proanthocyanins        Color

#> Min.   :-2.06321   Min.   :-1.6297

#> 1st Qu.:-0.59560   1st Qu.:-0.7929

#> Median :-0.06272   Median :-0.1588

#> Mean   : 0.00000   Mean   : 0.0000

#> 3rd Qu.: 0.62741   3rd Qu.: 0.4926

#> Max.   : 3.47527   Max.   : 3.4258

#>      Hue              Dilution

#> Min.   :-2.08884   Min.   :-1.8897

#> 1st Qu.:-0.76540   1st Qu.:-0.9496

#> Median : 0.03303   Median : 0.2371

#> Mean   : 0.00000   Mean   : 0.0000

#> 3rd Qu.: 0.71116   3rd Qu.: 0.7864

#> Max.   : 3.29241   Max.   : 1.9554

#>    Proline

#> Min.   :-1.4890

#> 1st Qu.:-0.7824

#> Median :-0.2331

#> Mean   : 0.0000

#> 3rd Qu.: 0.7561

#> Max.   : 2.9631

**Model Fitting**

We can use NbClust function to determine what is the best number of clusteres k for K-Means

nc <- NbClust(data.train,

              min.nc=2, max.nc=15,

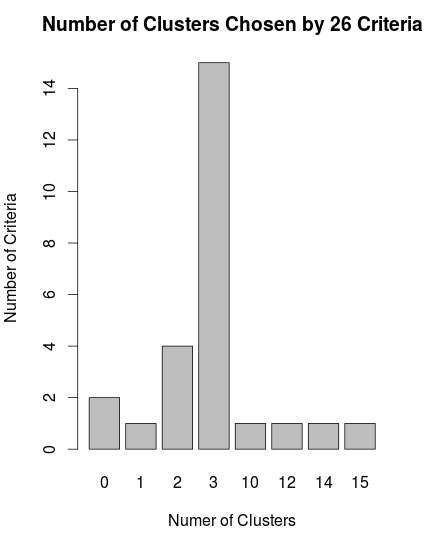
              method="kmeans")

barplot(table(nc$Best.n[1,]),

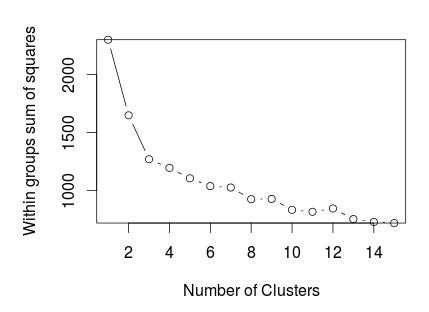
        xlab="Numer of Clusters",

        ylab="Number of Criteria",

        main="Number of Clusters Chosen by 26 Criteria")



According to the graph, we can find the best number of clusters is 3. Beside NbClust function which provides 30 indices for determining the number of clusters and proposes the best clustering scheme, we can draw the sum of square error (SSE) scree plot and look for a bend or elbow in this graph to determine appropriate k.



Both two methods suggest k=3 is best choice for us. It's reasonable if we take notice that the original data set also contains 3 classes.

Fit the model

We now fit wine data to K-Means with k = 3

fit.km <- kmeans(data.train, 3)

Then interpret the result

fit.km

#> K-means clustering with 3 clusters of sizes 51, 65, 62

#>

#> Cluster means:

#>      Alcohol      Malic        Ash Alcalinity

#> 1  0.1644436  0.8690954  0.1863726  0.5228924

#> 2 -0.9234669 -0.3929331 -0.4931257  0.1701220

#> 3  0.8328826 -0.3029551  0.3636801 -0.6084749

#>     Magnesium     Phenols  Flavanoids Nonflavanoids

#> 1 -0.07526047 -0.97657548 -1.21182921    0.72402116

#> 2 -0.49032869 -0.07576891  0.02075402   -0.03343924

#> 3  0.57596208  0.88274724  0.97506900   -0.56050853

#>   Proanthocyanins      Color        Hue   Dilution

#> 1     -0.77751312  0.9388902 -1.1615122 -1.2887761

#> 2      0.05810161 -0.8993770  0.4605046  0.2700025

#> 3      0.57865427  0.1705823  0.4726504  0.7770551

#>      Proline

#> 1 -0.4059428

#> 2 -0.7517257

#> 3  1.1220202

#>

#> Clustering vector:

#>   [1] 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

#>  [26] 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

#>  [51] 3 3 3 3 3 3 3 3 3 2 2 1 2 2 2 2 2 2 2 2 2 2 2 3 2

#>  [76] 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2

#> [101] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 3 2 2 2

#> [126] 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

#> [151] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

#> [176] 1 1 1

#>

#> Within cluster sum of squares by cluster:

#> [1] 326.3537 558.6971 385.6983

#>  (between\_SS / total\_SS =  44.8 %)

#>

#> Available components:

#>

#> [1] "cluster"      "centers"      "totss"

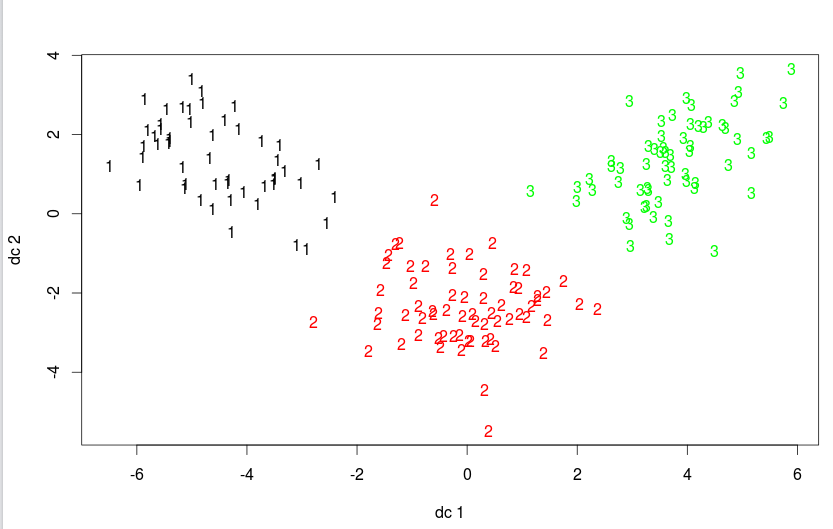
#> [4] "withinss"     "tot.withinss" "betweenss"

#> [7] "size"         "iter"         "ifault"

The result shows information about cluster means, clustering vector, sum of square by cluster and available components. Let's do some visualizations to see how data set is clustered. First, we use plotcluster function from fpc package to draw discriminant projection plot

library(fpc)

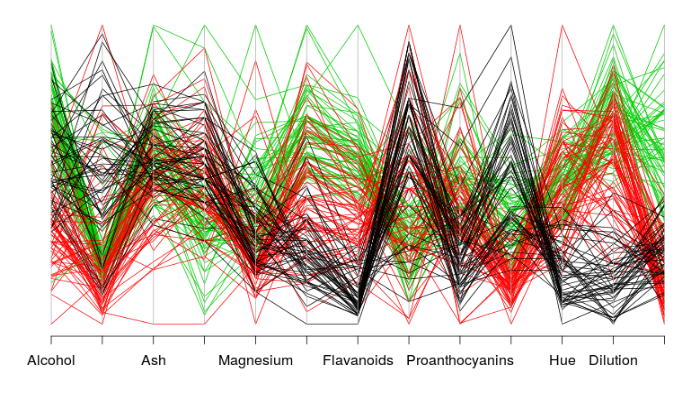
plotcluster(data.train, fit.km$cluster)



We can see the data is clustered very well, there are no collapse between clusters. Next, we draw parallel coordinates plot to see how variables contributed in each cluster

library(MASS)

parcoord(data.train, fit.km$cluster)



We can extract some insights from above graph as black cluster contains wine with low flavanoids value, low proanthocyanins value, low hue value. Or green cluster contains wine which has dilution value higher than wine in red cluster.

Evaluation

Because the original data set wine also has 3 classes, it is reasonable if we compare these classes with 3 clusters fitted by K-Means

confuseTable.km <- table(wine$Type, fit.km$cluster)

confuseTable.km

#>    1  2  3

#> 1  0  0 59

#> 2  3 65  3

#> 3 48  0

We can see only 6 sample is missed. Let's use randIndex from flexclust to compare these two parititions – one from data set and one from result of clustering method.

library(flexclust)

randIndex(ct.km)

#>      ARI

#> 0.897495

It's quite close to 1 so K-Means is good model for clustering wine data set.

HOMEWORK 1

Use the k-medoids clustering method with the wine data shown above

HOMEWORK 2

**K-means Clustering Analysis of Red Wine Quality**

In Jun 2014, Business Insider published [an article](http://www.businessinsider.com/recognize-high-quality-wine-2014-6) to list three main explanation of high quality of red wine: complexity, intensity, and balance. In 2009, a dataset, created by Paulo Cortez (Univ. Minho), Antonio Cerdeira, Fernando Almeida, Telmo Matos and Jose Reis, provided 1599 types of red wine with 10 scientific attributes associated with the quality. The data is available to be downloaded on University of California Irvine machine learning [web page](http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/).   
Create a code in R Studio to conduct the k-means clustering analysis of the red wine quality based on the data provided on UCI website.

Hind - Step1: Scale the data - As the measurement of free sulfur dioxide is from 1 to 72 while citric acidity is scaled from 0 to 1. We need to scale the data in order to perform accuracy of distance of each clusters.