Kmeans & Hierarchical clustering

```r
#install.packages('rattle')
data(wine, package='rattle.data') #inbuilt data in rattle package
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
```

```r
wine.stand <- scale(wine[-1])  # To standarize the variables
View(wine.stand)
# K-Means #flat clustering where we decide number of clustering
k.means.fit <- kmeans(wine.stand, 3) # k = 3
attributes(k.means.fit)
```

```
## $names
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
##
## $class
## [1] "kmeans"
```

```r
#centroid
k.means.fit$centers
```

```
##      Alcohol      Malic        Ash Alcalinity   Magnesium     Phenols
## 1 -0.9234669 -0.3929331 -0.4931257  0.1701220 -0.49032869 -0.07576891
## 2  0.8328826 -0.3029551  0.3636801 -0.6084749  0.57596208  0.88274724
## 3  0.1644436  0.8690954  0.1863726  0.5228924 -0.07526047 -0.97657548
##    Flavanoids Nonflavanoids Proanthocyanins      Color        Hue
## 1  0.02075402   -0.03343924      0.05810161 -0.8993770  0.4605046
## 2  0.97506900   -0.56050853      0.57865427  0.1705823  0.4726504
## 3 -1.21182921    0.72402116     -0.77751312  0.9388902 -1.1615122
##     Dilution    Proline
## 1  0.2700025 -0.7517257
## 2  0.7770551  1.1220202
## 3 -1.2887761 -0.4059428
```

```r
# Clusters:
k.means.fit$cluster
```

```
##   [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
##  [36] 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 1 1 3 1 1 1 1 1 1 1 1
##  [71] 1 1 1 2 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
## [106] 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 2 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3
## [141] 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 3 3 3 3 3 3 3 3 3
## [176] 3 3 3
```

```
# Cluster size:
k.means.fit$size
```

```
## [1] 65 62 51
```

```
sum(apply(wine.stand,2,var)) #calculating variance column wise &
```

```
## [1] 13
```

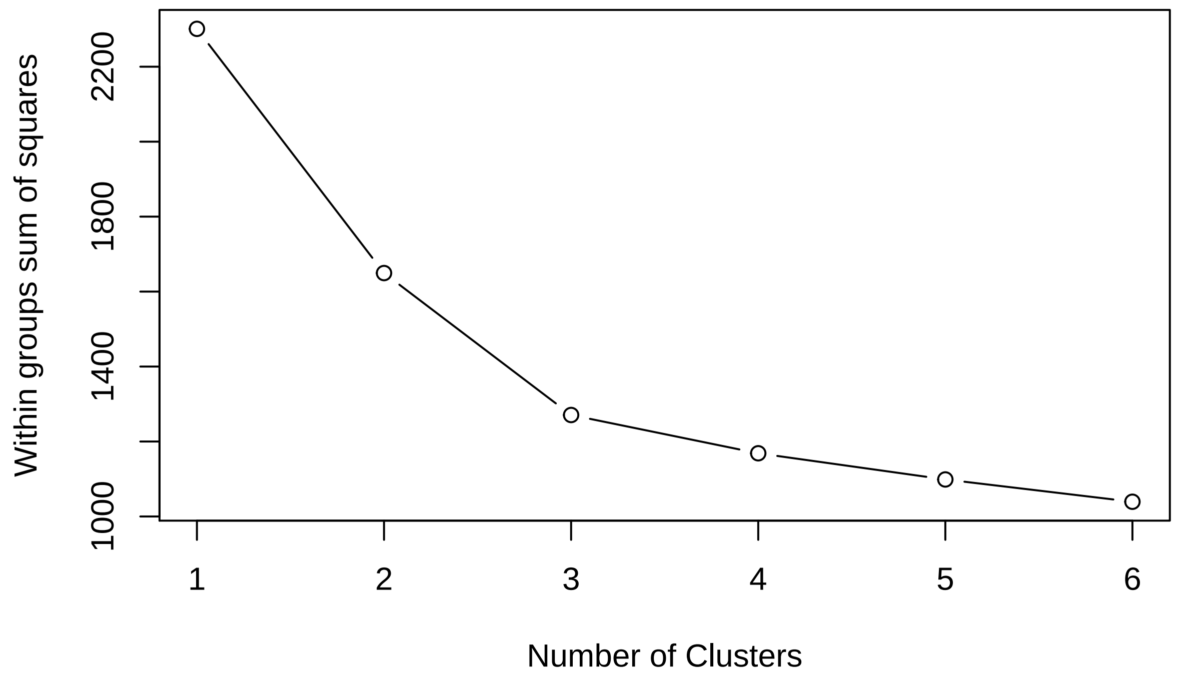
```
#then summing it up.
(nrow(wine.stand)-1)#total number of observation-1
```

```
## [1] 177
```

```
(nrow(wine.stand)-1)*sum(apply(wine.stand,2,var)) #within sum of
```

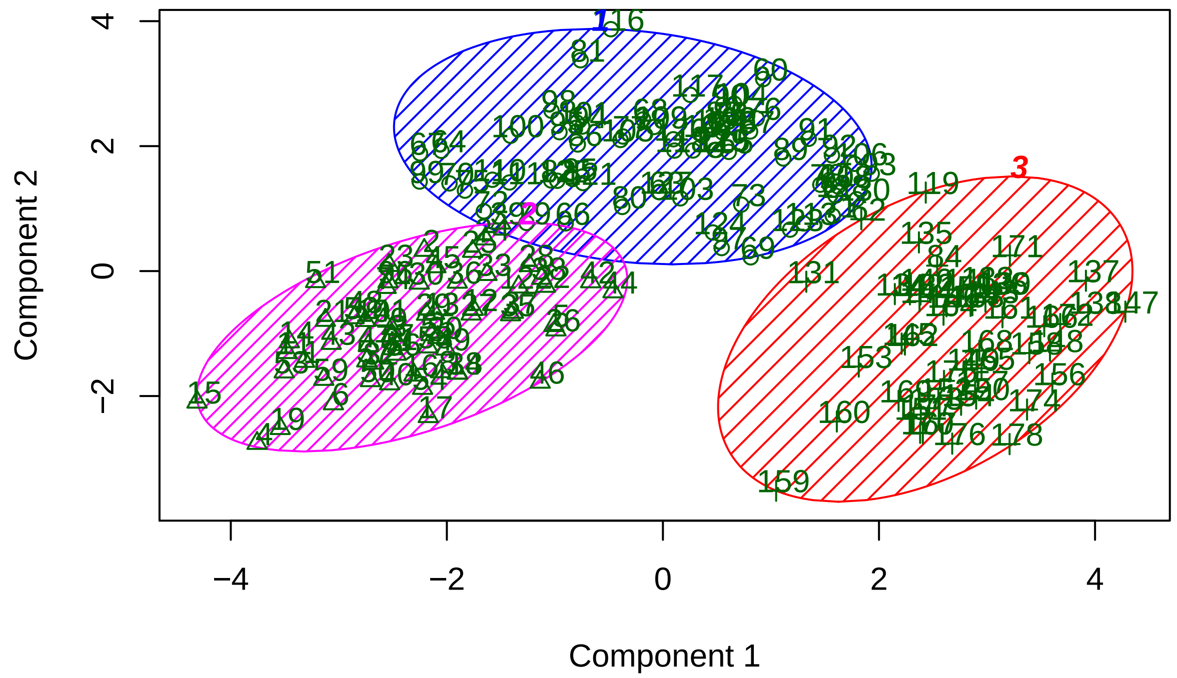
```
## [1] 2301
```

```
#variance in the data.
#now we will turn the code in loop to find wss for different
#clusters created using Kmeans.

wssplot <- function(data, nc=15, seed=1234){
  wss <- (nrow(data)-1)*sum(apply(data,2,var))
  for (i in 2:nc){
    set.seed(seed)
    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
  plot(1:nc, wss, type="b", xlab="Number of Clusters",
       ylab="Within groups sum of squares")}

wssplot(wine.stand, nc=6)
```



```r
#Our whole purpose here to reduce wss, and as we see after 3 or 4
# clusters we are not able to reduce wss much, so we can go ahead
# with 3 or 4.

library(cluster)
clusplot(wine.stand, k.means.fit$cluster, main='2D representation of the Cluster solution',
         color=TRUE, shade=TRUE,
         labels=2, lines=0)
```

2D representation of the Cluster solution



These two components explain 55.41 % of the point variability.

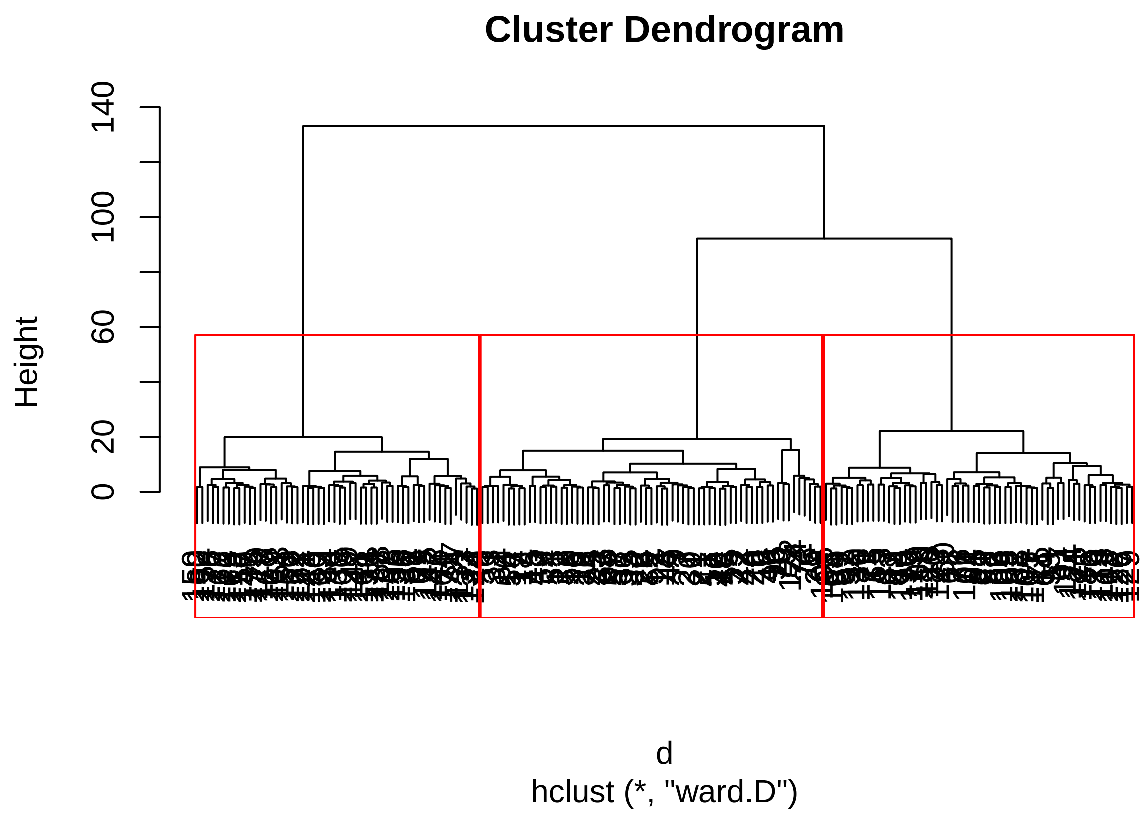
```
#good job with 3 clusters, as the we are able to make a good
#seperation between different clusters.

table(wine[,1],k.means.fit$cluster)
```

```
##
##      1  2  3
##   1  0 59  0
##   2 65  3  3
##   3  0  0 48
```

```
#Hierarchical clustering #soft clustering, as the clusters get
#decided automatically.
d <- dist(wine.stand, method = "euclidean") # Euclidean distance matrix.

H.fit <- hclust(d, method="ward")
```

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```
plot(H.fit) # display dendogram
groups <- cutree(H.fit, k=3) # cut tree into 3 clusters
# draw dendogram with red borders around the 3 clusters
rect.hclust(H.fit, k=3, border="red")
```

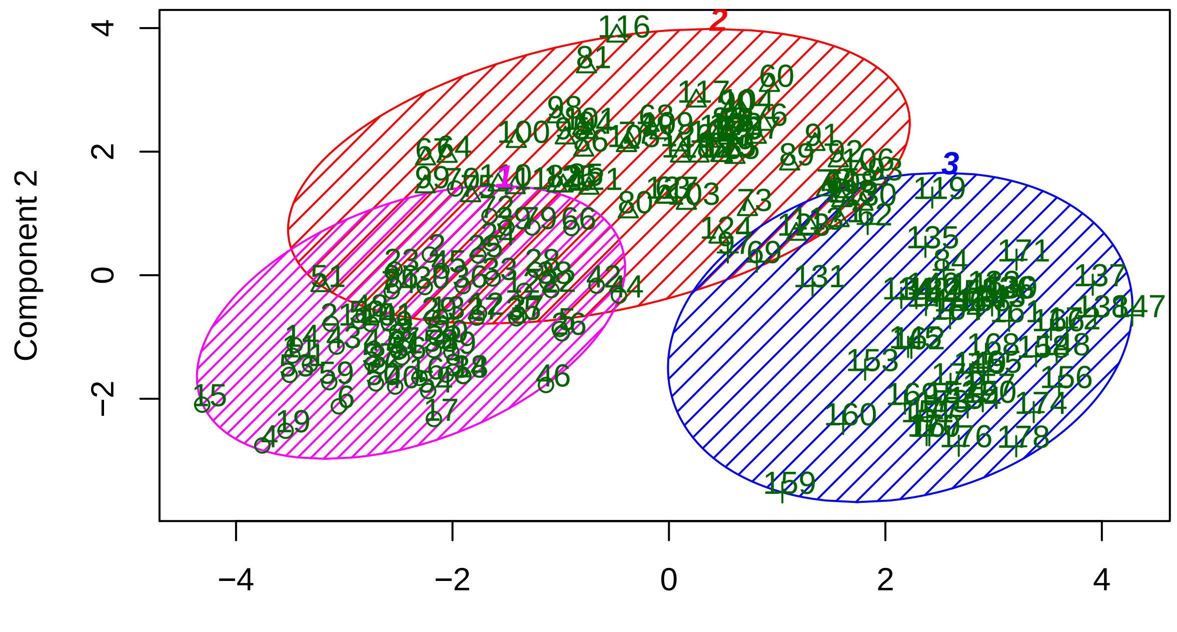


```
table(wine[,1],groups)
```

```
##     groups
##       1  2  3
##    1 58  1  0
##    2  7 58  6
##    3  0  0 48
```

```
#here we can see that hierarchical is doing good job in creating
# a good clusters.

clusplot(wine.stand, groups, main='2D representation of the Cluster solution',
         color=TRUE, shade=TRUE,
         labels=2, lines=0)
```

2D representation of the Cluster solution



These two components explain 55.41 % of the point variability.

```
#though this plot does have more overlapping than the kmeans
#clustering but this method is able to distiguish the class well.
```