K-means clustering

Karen Mazidi

Load the data

We are using the built-in iris data set. One nice thing about this data is that the features are already on the same scale, so scaling is not necessary.

```
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
```

Run the k-means algorithm

Now we try clustering with kmeans().

We are using just Petal.Length and Petal.Width for clustering. The number of clusters is set to 3 and the number of starts is 20.

The k=3 choice is somewhat like cheating. We know iris has 3 species. Normally in k-means clustering we don't have any ground truth knowledge.

We need several starts because of the random initialization. The k-means algorithm will select the best clustering of the 20 iterations.

```
set.seed(1234)
irisCluster <- kmeans(iris[, 3:4], 3, nstart=20)
irisCluster
```

```
## K-means clustering with 3 clusters of sizes 52, 48, 50
##
## Cluster means:
##   Petal.Length Petal.Width
## 1     4.269231    1.342308
## 2     5.595833    2.037500
## 3     1.462000    0.246000
##
## 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 3 3 3 3 3 3 3 3 3 3 3 3
##  [38] 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 1 1 1 1 1 1
##  [75] 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 2 2 2 2
## [112] 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2
## [149] 2 2
##
## Within cluster sum of squares by cluster:
## [1] 13.05769 16.29167  2.02200
##  (between_SS / total_SS =  94.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Examine the clustering

The clusters were of size 52, 48, 50 and we know the species are 50-50-50, so the algorithm did find similarities in the features.

Compare the clusters with the species. This is not usally something we can do in clustering because we normally don't have labels. We are usually clustering blind, not knowing the true grouping in the data.

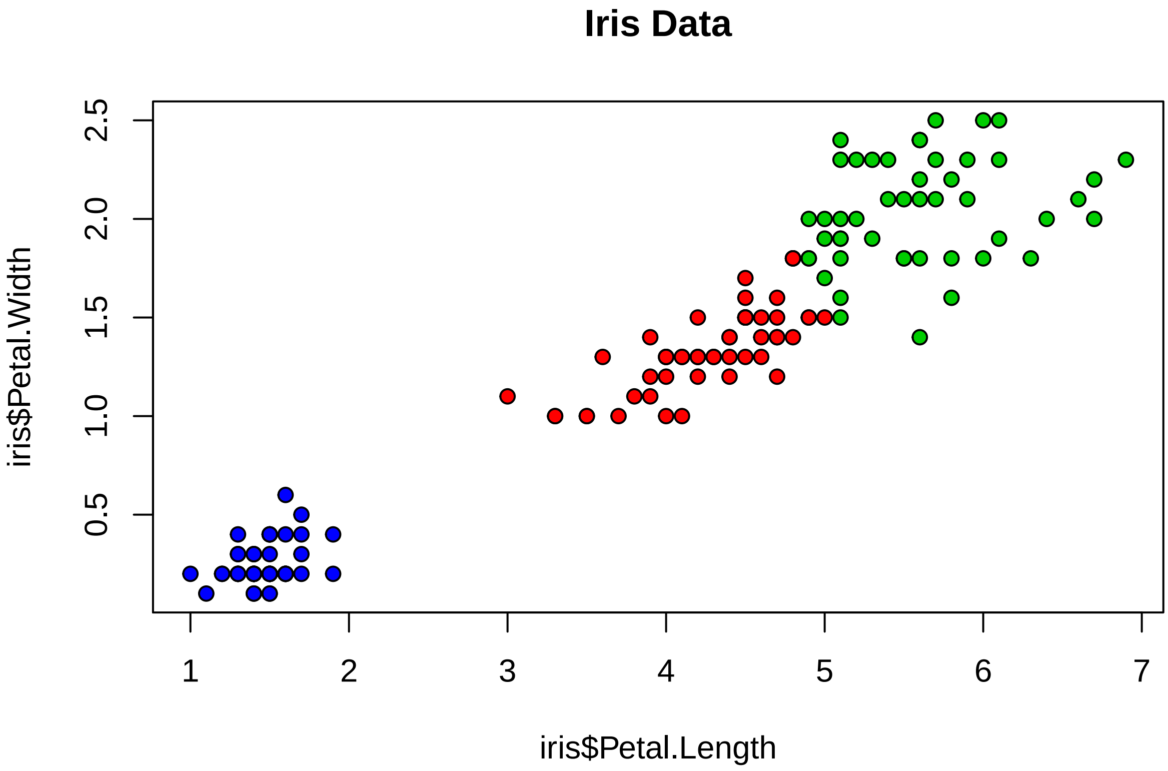
```
table(irisCluster$cluster, iris$Species)
```

```
##
##     setosa versicolor virginica
##   1      0         48         4
##   2      0          2        46
##   3     50          0         0
```

Plot the data

Plot the iris data, color coding the true labels.

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



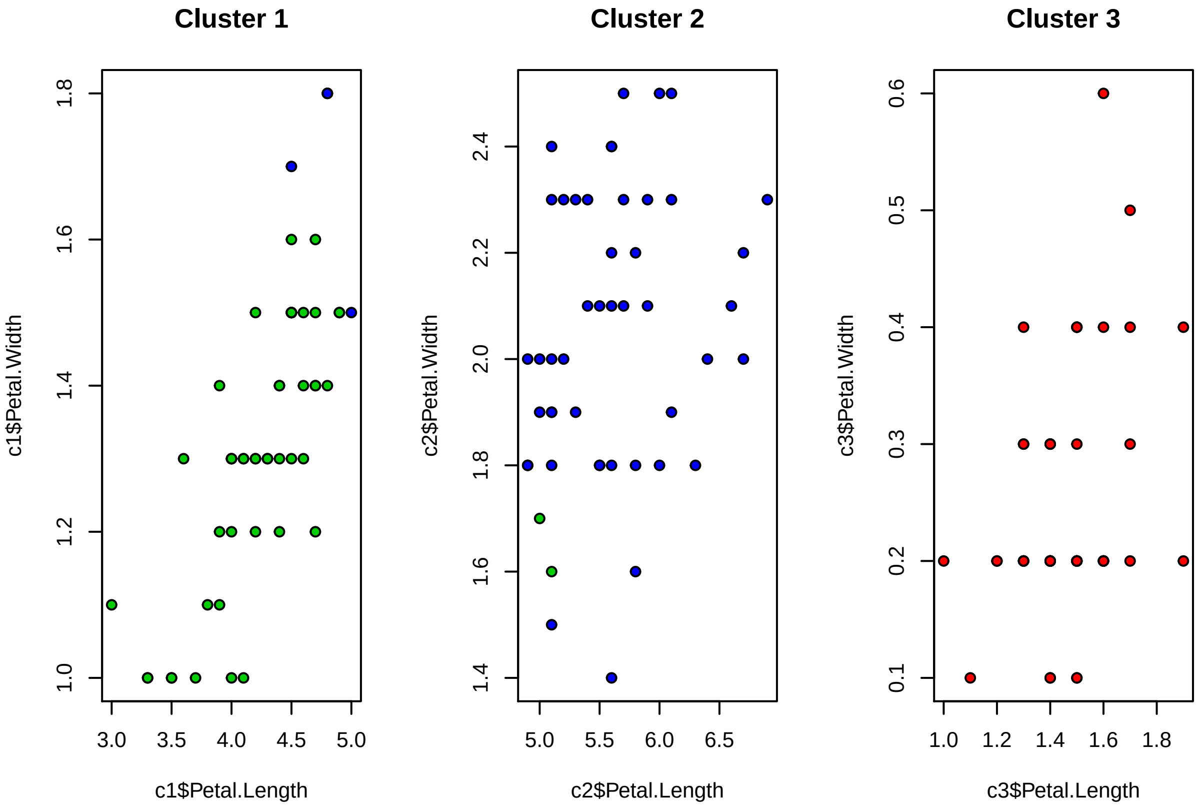
Plot the clusters.

Notice that each cluster plots on its own x axis and y axis. Compare the cluster means in the output with each graph.

```
c1 = iris[irisCluster$cluster==1,]
c2 = iris[irisCluster$cluster==2,]
```

```
c3 = iris[irisCluster$cluster==3,]

par(mfrow=c(1,3))
plot(c1$Petal.Length, c1$Petal.Width, pch=21, bg=c("red","green3","blue")
[unclass(c1$Species)], main="Cluster 1")
plot(c2$Petal.Length, c2$Petal.Width, pch=21, bg=c("red","green3","blue")
[unclass(c2$Species)], main="Cluster 2")
plot(c3$Petal.Length, c3$Petal.Width, pch=21, bg=c("red","green3","blue")
[unclass(c3$Species)], main="Cluster 3")
```



k-means metrics

What is a cluster?

A clustering of data creates sets of observations for which each observation belongs to one and only one cluster.

Take another look at the output of the clustering. One metric that is output is the within cluster sum of squares for each cluster:

```
13.05769 16.29167  2.02200
```

Each centroid has its own mean, shown in the output. The squared difference between each observation and the mean is summed to create the "sum of squares" metric. Cluster 3 has the lowest value. Look back at the cluster 3 diagram above. The scale of both the x axis and the y axis is the lowest range of the 3 clusters, indicating that these observations are very close.

Another metric is the between_SS divided by the total SS. This is a metric of how well separated the clusters are.

```
 (between_SS / total_SS =  94.3 %)
```

The 94% is a good result. A large value indicates that the clusters are well separated.

```r
irisCluster
```

```
## K-means clustering with 3 clusters of sizes 52, 48, 50
##
## Cluster means:
##   Petal.Length Petal.Width
## 1     4.269231    1.342308
## 2     5.595833    2.037500
## 3     1.462000    0.246000
##
## 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 3 3 3 3 3 3 3 3 3 3 3 3
##  [38] 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 1 1 1 1 1 1
##  [75] 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 2 2 2 2
## [112] 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2
## [149] 2 2
##
## Within cluster sum of squares by cluster:
## [1] 13.05769 16.29167  2.02200
##  (between_SS / total_SS =  94.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Notice the algorithm lists the variables you can examine. Items we haven't look at are number of iterations. Convergence happened after only two iterations. Convergence is when the total sum of squared errors is below a certain threshhold.

The documentation says that ifault is an indication of a possible algorithm problem - for experts.

```r
print(paste("number of iterations", irisCluster$iter))
```

```
## [1] "number of iterations 2"
```

```r
print(paste("ifault", irisCluster$ifault))
```

```
## [1] "ifault 0"
```