Lab-09

Ehshan Veerabangsa

7 December 2017

K-Means Clustering

Load Data

First we need to load the dataset. For these experiments we will use a subset of the data, using only the features Learning & Privilege, to avoid the need for PCA.

```
library(datasets)
dat = attitude[,c(3,4)]
```

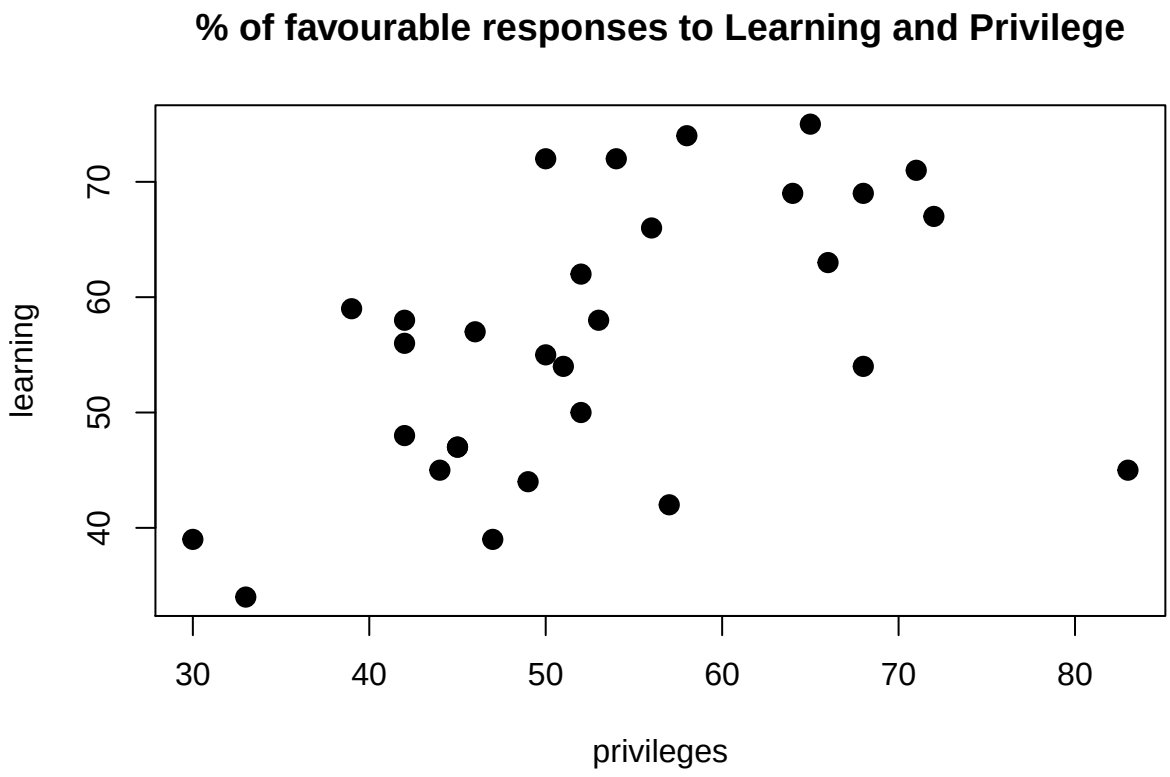
We can explore the data and its features by using the "?" command.

```
#?attitude
```

Part 1: Plot data

We can create a 2D plot of the dataset using the two extracted features.

```
plot(dat, main = "% of favourable responses to Learning and Privilege", pch = 20, cex = 2)
```



Part 2: Perform k-means with 2 clusters, and a single starting point

For our first experiment, we will use K-Means, with 2 cluster centers, and 1 start position for our cluster centers.

```
set.seed(1)

dat_k2_1 <- kmeans(dat,2, nstart = 1)

dat_k2_1$cluster
```

```
##  [1] 1 1 2 1 2 1 1 1 2 1 1 1 1 2 2 2 2 2 1 2 1 2 1 1 1 2 2 1 2 1
```

Part 3: Calcualte the total within clusters sum of squares (SOS for all clusters)

To measure the success of our K-Means algorithm, we can check the total sum of squares for all our clusters. The lower the figure the better the clustering.

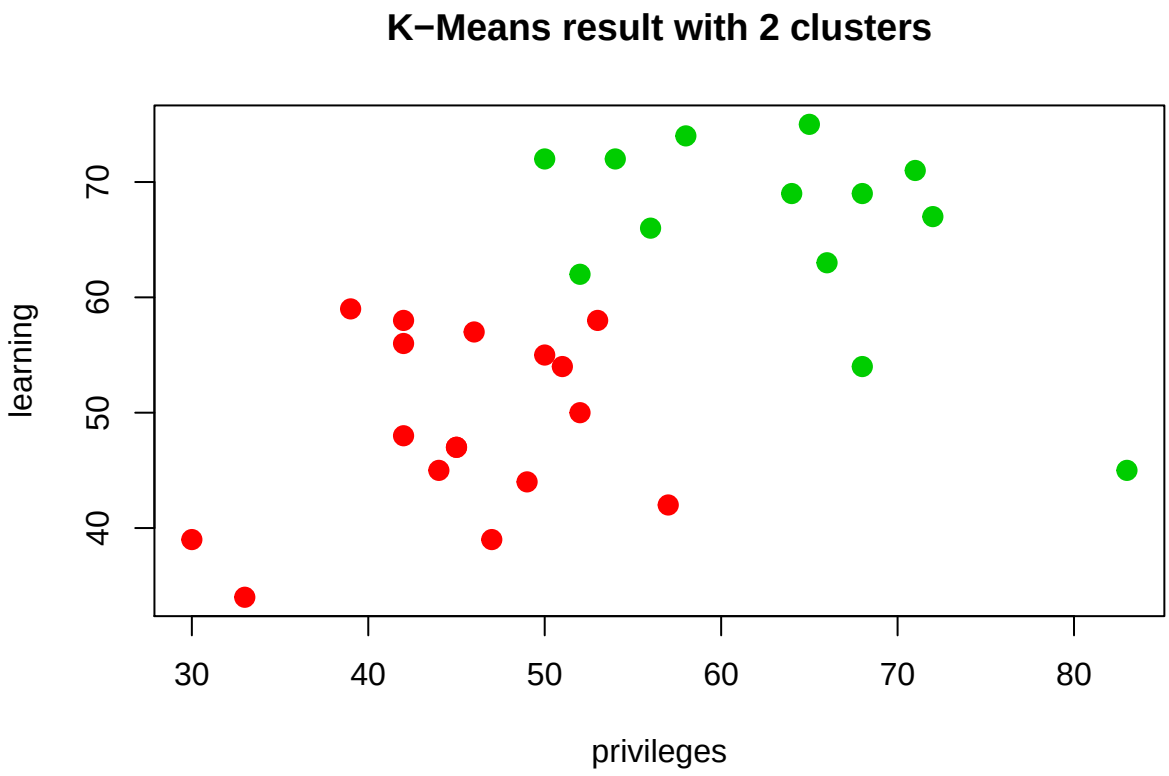
```
dat_k2_1$tot.withinss
```

```
## [1] 3652.706
```

Part 4: Plot the Clustering results

We can visualise our K-Means result by plotting the outcome.

```
plot(dat, col = (dat_k2_1$cluster +1) , main = "K-Means result with 2 clusters", pch = 20, cex = 2)
```



Part 5: Perform k-means with 2 clusters, and 100 starting points

Next will will repeat the experiment, but use 100 different starting positions for our cluster centers, and take the best result. We can then compare the total sum of squares with our original value.

```
set.seed(1)

dat_k2_100 <- kmeans(dat,2, nstart = 100)

dat_k2_100$cluster
```

```
##  [1] 2 2 1 2 1 2 2 2 1 2 2 2 2 1 1 1 1 1 2 1 2 1 2 2 2 1 1 2 1 2
```

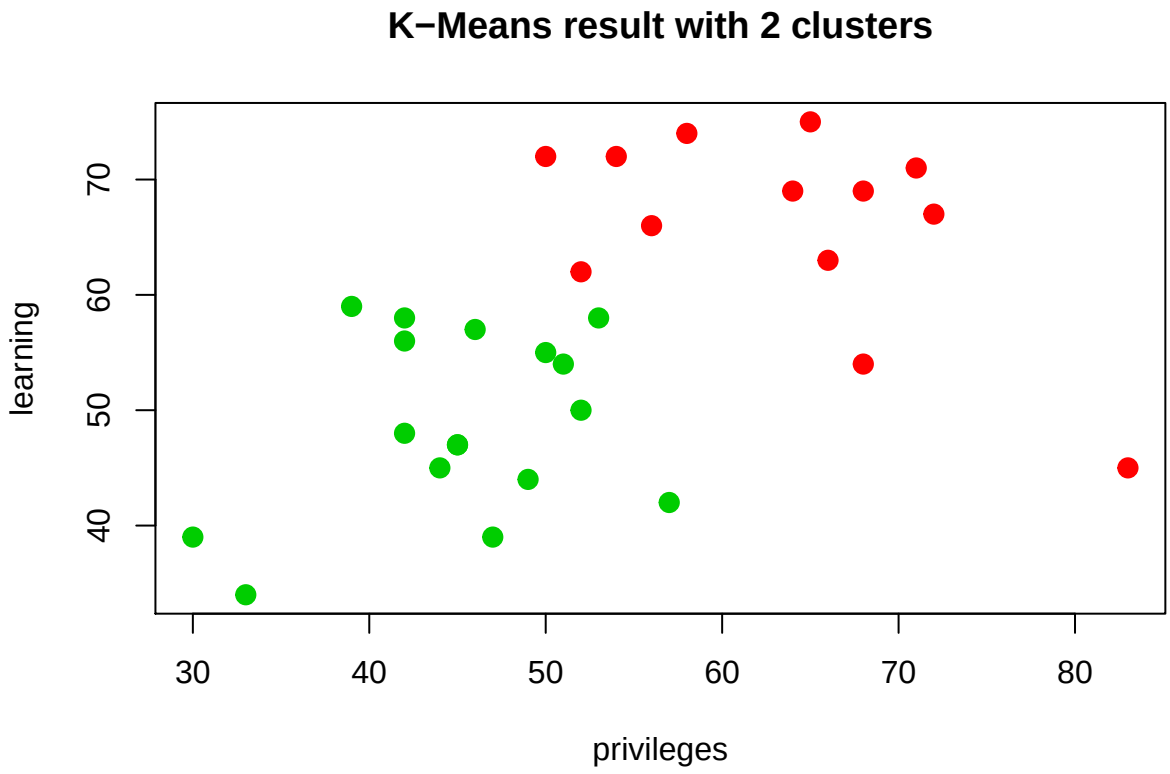
```
dat_k2_100$tot.withinss
```

```
## [1] 3652.706
```

As we can see, the total sum of squares gives the same value.

Now we can plot the results from our new model.

```r
plot(dat, col = (dat_k2_100$cluster +1) , main = "K-Means result with 2 clusters", pch = 20, cex = 2)
```



Again, we see the same resulting clusters.

Part 6: Find best K value

To obtain the best value for K, we can use a loop, to create several different model, and return each total sum of squares value in a vector.

```r
# initialise vector to hold results
totWithinSS <- rep(0,15)

# capture total sum of squares for all k values 1-15
for(i in 1:15){
  set.seed(1)
  totWithinSS[i] = kmeans(dat,i,nstart = 100)$tot.withinss
}

totWithinSS
```

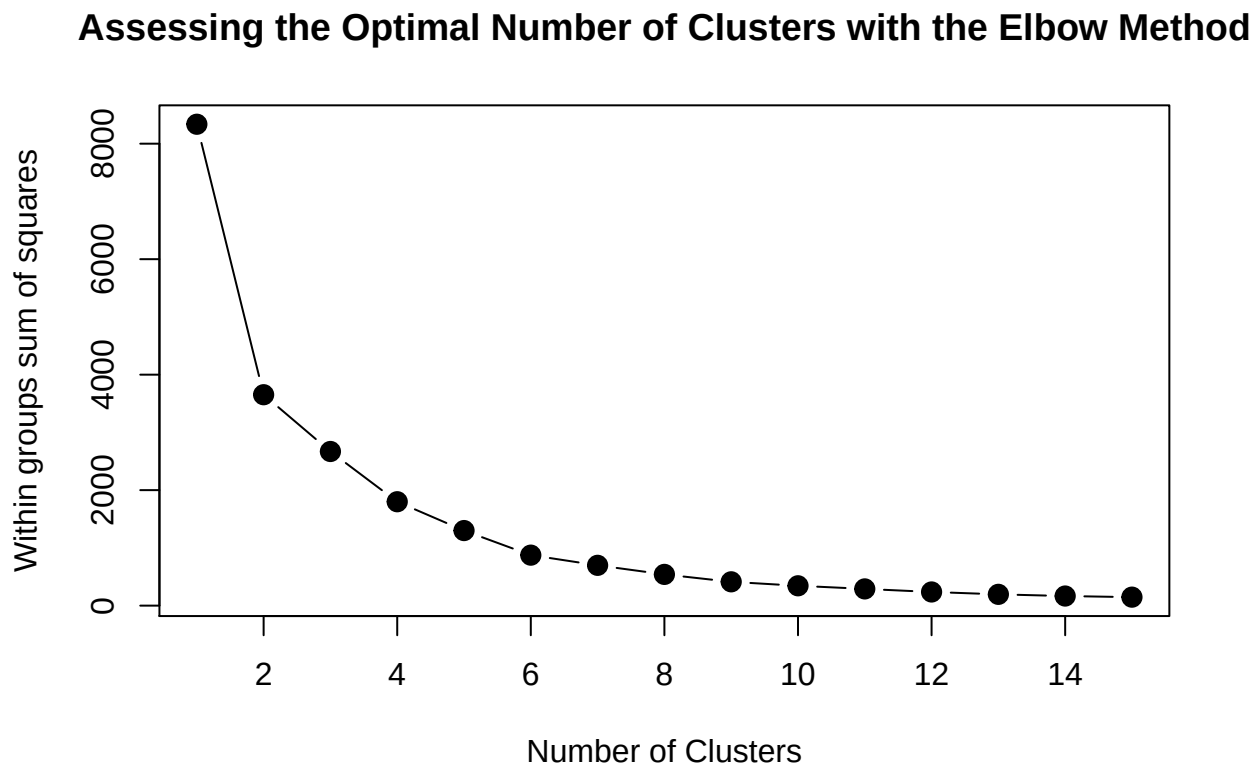
```
##  [1] 8336.4333 3652.7059 2669.3422 1799.2222 1300.1250  874.4583  697.7976
##  [8]  540.7643  412.4667  343.8833  289.7167  236.6667  195.4167  166.3333
## [15]  147.0833
```

As we can see, the value decreases in line with the number of clusters.

Part 7: Plot results and find optimal value

To find the optimal value, we can plot our resulting vector, to find the optimal result. We will use the Elbow method to find the point were the improvement in the sum of squares starts to decrease.

```
plot(1:15, totWithinSS, type = "b", xlab="Number of Clusters",
     ylab = "Within groups sum of squares",
     main = "Assessing the Optimal Number of Clusters with the Elbow Method",
     pch = 20, cex = 2)
```

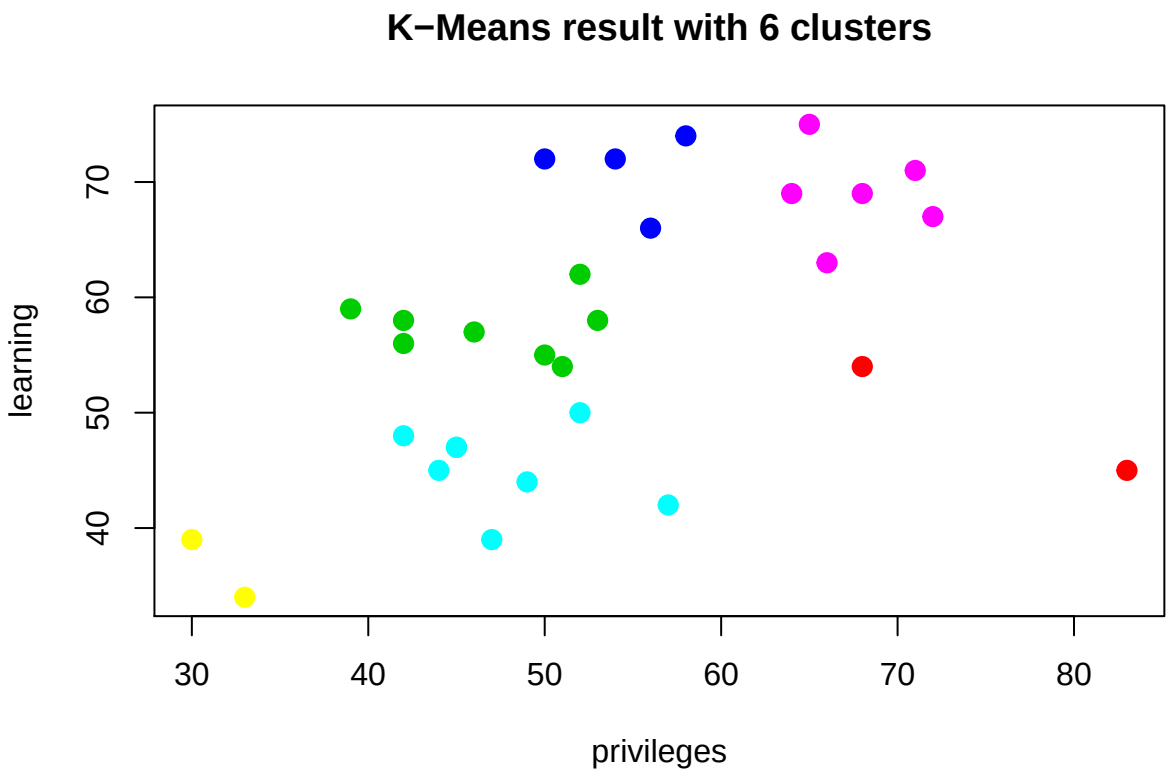


As we can see, we start to get diminishing returns from k = 6 onwards.

Part 8: Plot the clusters for the optimal value fo k (K = 6)

Finally, we can plot the results from our optimal K-Means model.

```
set.seed(1)

dat_k6_100 <- kmeans(dat,6,nstart = 100)

plot(dat, col = (dat_k6_100$cluster +1) , main = "K-Means result with 6 clusters", pch = 20, cex = 2)
```



Hierarchical Clustering

Load dataset

First we will load the dataset from our local csv file.

```
DF <- read.csv("..\\..\\week10\\Ch10Ex11.csv", header = FALSE)
```

Part 1: Transposed dataset

For the next stage, we will transpose the resulting matrix, for a more intuitive interpretation of the data.

```
DF <- t(DF)
```

Part 2: Calculate the dissimilarity metric

Now we will calculate the dissimilarity metric.

```
# cor computes the correlation of *columns* so we need to take the transpose of DF
D = as.dist( 1 - cor(t(DF)))
#D
```

```
DFcor =  1 - cor(t(DF))
#DFcor
```

Part 3: Calculate the different models by linkage

We will now build 4 models, each using a differnt form of cluster linkage.

a: Complete

```
complete_fit <-hclust(D, method = "complete")
```

b: Average

```
average_fit <- hclust(D, method = "average")
```

c: Single

```
single_fit <- hclust(D, method = "single")
```

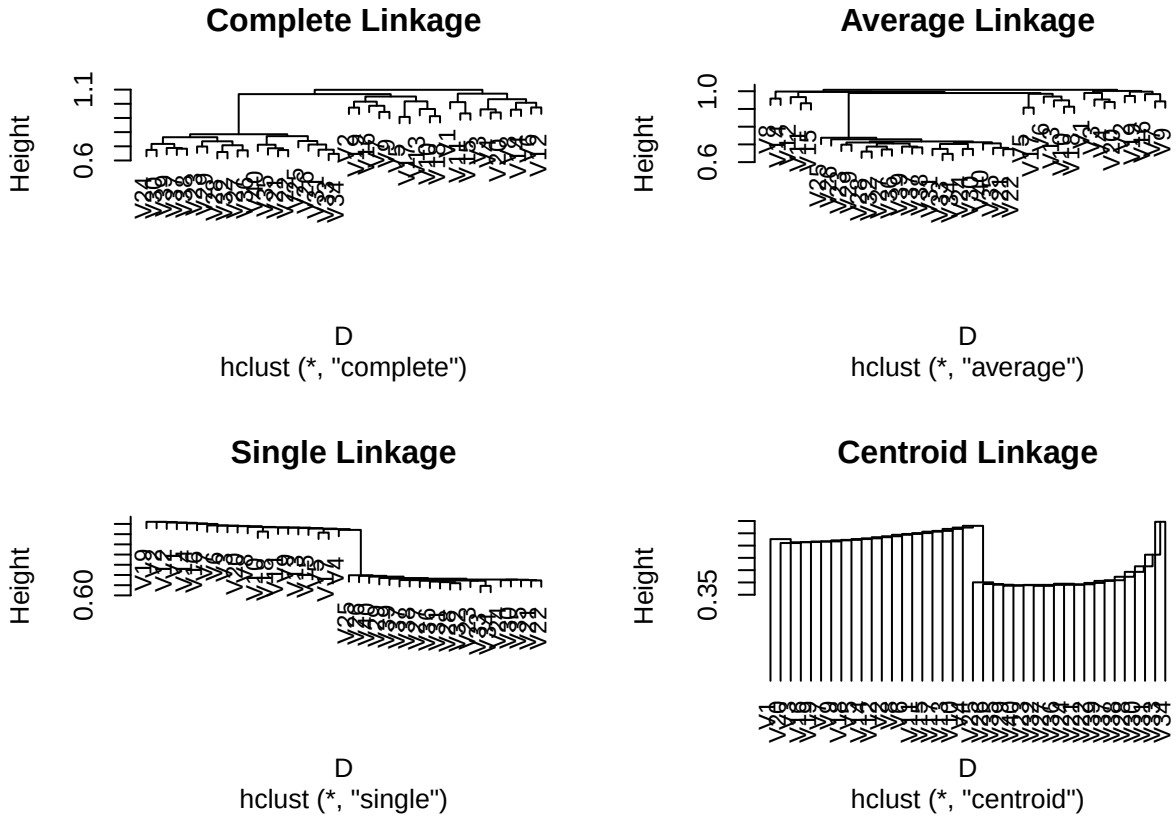
d: Centroid

```
centroid_fit <- hclust(D, method = "centroid")
```

Part 4: Plot models

We can plot our 4 model to visualise the differences.

```
# split the plot space
par(mfrow=c(2,2))

# Plot models

plot(complete_fit, main = "Complete Linkage",cex = 0.9)

plot(average_fit, main = "Average Linkage",cex = 0.9)

plot(single_fit, main = "Single Linkage",cex = 0.9)

plot(centroid_fit, main = "Centroid Linkage",cex = 0.9)
```



Part 5: Confusion Matrix

To see which model performs best, we will compute the confusion matrix for each.

First we need to create a label vector representing the actual values in the dataset. As there are 20 healthy and 20 unhealthy samples, we will assign the first 20 values as healthy, and the second 20 as unhealthy.

```
actual_labels <- c(rep(0,20), rep(1,20))
```

```
print(table(predicted = cutree(complete_fit, k = 2 ), actual = actual_labels))
```

```
##           actual
## predicted  0  1
##         1 10  0
##         2 10 20
```

```
print(table(predicted = cutree(average_fit, k = 2 ), actual = actual_labels))
```

```
##           actual
## predicted  0  1
##         1  9  0
##         2 11 20
```

```
print(table(predicted = cutree(single_fit, k = 2 ), actual = actual_labels))
```

```
##           actual
## predicted  0  1
##         1 19 20
```

```
##         2  1  0
```

```
print(table(predicted = cutree(centroid_fit, k = 2 ), actual = actual_labels))
```

```
##          actual
## predicted  0  1
##         1  1  0
##         2 19 20
```

Part 6: Analysis

Part 5 shows that results will depend on the type of linkage used.