Clustering

Karen Mazidi

Load the wine data

Apply k-means to the wine data set, which contains 13 chemical measurements on 178 Italian red and white wines.

The column 'type' is a binary factor indicating red or white wine. This variable will not be included in the clustering.

```
wine <- read.csv('wine_all.csv')
str(wine)
```

```
## 'data.frame':    6497 obs. of  13 variables:
##  $ fixed_acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
##  $ volatile_acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
##  $ citric_acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
##  $ residual_sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
##  $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
##  $ free_sulfur_dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
##  $ total_sulfur_dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
##  $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
##  $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
##  $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
##  $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
##  $ type                : Factor w/ 2 levels "red","white": 1 1 1 1 1 1 1 1 1 1 ...
```

scale the data

Use R's built-in scale() function to scall all columns except the factor 'type'.

Since we don't have a train/test split in unsupervised learning, just scale the whole data frame.

```
df <- scale(wine[-13])
head(df)
```

```
##      fixed_acidity volatile_acidity citric_acid residual_sugar chlorides
## [1,]     0.1424623        2.1886645   -2.192664     -0.7447208 0.5699140
## [2,]     0.4510010        3.2819823   -2.192664     -0.5975941 1.1978825
## [3,]     0.4510010        2.5531038   -1.917405     -0.6606484 1.0266184
## [4,]     3.0735801       -0.3624106    1.660957     -0.7447208 0.5413699
## [5,]     0.1424623        2.1886645   -2.192664     -0.7447208 0.5699140
## [6,]     0.1424623        1.9457049   -2.192664     -0.7657389 0.5413699
##      free_sulfur_dioxide total_sulfur_dioxide   density         pH sulphates
## [1,]          -1.1000552           -1.4462472 1.0349132  1.8129500 0.1930819
## [2,]          -0.3112961           -0.8624022 0.7014323 -0.1150642 0.9995017
## [3,]          -0.8746955           -1.0924018 0.7681285  0.2580999 0.7978967
## [4,]          -0.7620156           -0.9862481 1.1016093 -0.3638402 0.3274852
## [5,]          -1.1000552           -1.4462472 1.0349132  1.8129500 0.1930819
## [6,]          -0.9873753           -1.3400936 1.0349132  1.8129500 0.1930819
##         alcohol    quality
## [1,] -0.9153937 -0.9371575
## [2,] -0.5800235 -0.9371575
## [3,] -0.5800235 -0.9371575
## [4,] -0.5800235  0.2079830
```

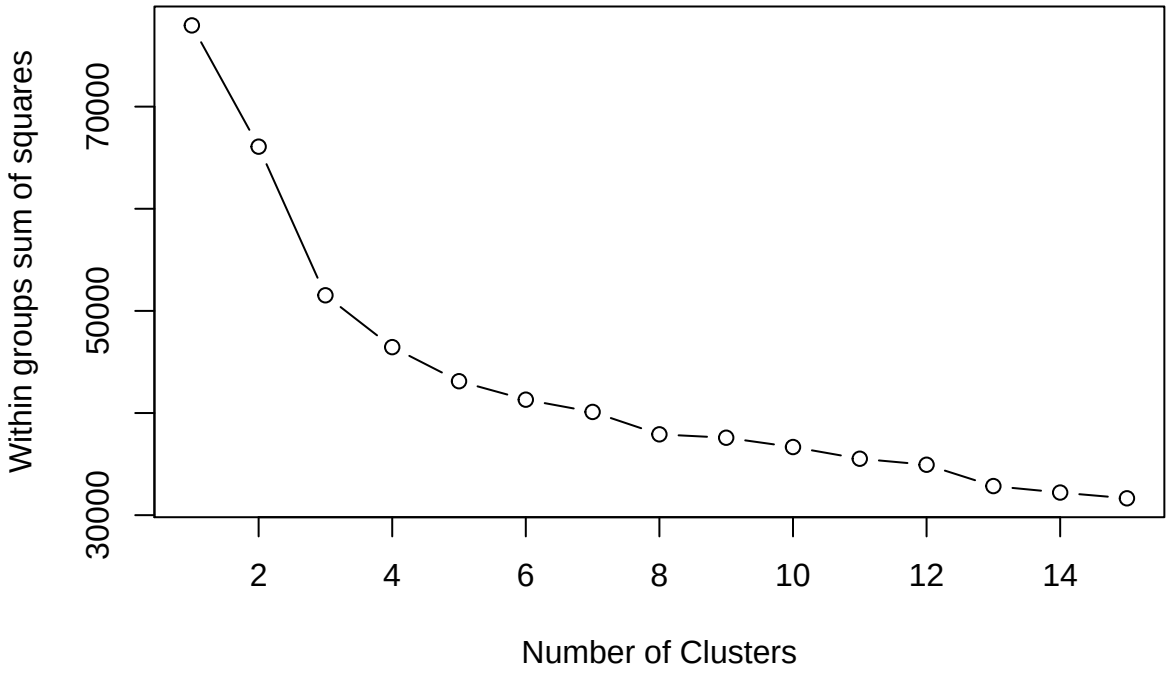
```
## [5,] -0.9153937 -0.9371575
## [6,] -0.9153937 -0.9371575
```

function to plot results for various values of k

Write a function to plot the within-groups sums of squares vs. the number of clusters. This function is modified from Kabacoff, "R in Action", 2nd ed

The plot shows an 'elbow' starting at around k=3. Higher numbers of clusters than 3 give diminishing reductions in sum of squared errors.

```
wsplot <- 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")
}
wsplot(df)
```



KMeans

Fit the model using the kmeans() function. We set a seed first so we get reproducible results.

The total within ss is displayed along with comparison of clusters and wine type.

```
set.seed(1234)
fit.km <- kmeans(df, 3, nstart=25)
print(paste('total withinss = ', fit.km$tot.withinss))
```

```
## [1] "total withinss =  51528.7734674255"
```

```
km_table <- table(wine$type, fit.km$cluster)
km_table
```

```
##
##            1    2    3
##   red     59 1536    4
##   white 2849   83 1966
```

Compare to k=4.

```
set.seed(1234)
fit.km <- kmeans(df, 4, nstart=25)
print(paste('total withinss = ', fit.km$tot.withinss))
```

```
## [1] "total withinss =  46452.0720312822"
```

```
km4_table <- table(wine$type, fit.km$cluster)
km4_table
```

```
##
##            1    2    3    4
##   red     60  927    3  609
##   white 2775  139 1932   52
```

The total within ss for k=4 clusters was lower than for k=3. However, when you compare the tables, the k=3 clustering had 146 observations out of their type compared to 202 for k=4. As you can see in the diagram above, the higher the k the lower the within ss.

Domain knowledge and application awareness are key to determining a 'best' clustering of the data.

We can quantify the agreement between the type and the cluster using a metric called Rand index. The Rand index provides a measure of the agreement between two partitions. The range of the index is from 0 (no agreement) to +1 (perfect agreement).

The agreement for both clusterings is good but not great. The correct= argument was set to FALSE to not correct for chance. The parameter 'correct=FALSE' prevents adjusting the metric by chance (class distribution). When set to TRUE, the metric is called 'Adjusted Rand Index' and it will generally be lower. Adjusted Rand Index values range from -1 to +1.

```
library(flexclust)
```

```
## Loading required package: grid
## Loading required package: lattice
## Loading required package: modeltools
## Loading required package: stats4
```

```
print(paste('rand index for k=3', randIndex(km_table, correct=FALSE)))
```

```
## [1] "rand index for k=3 0.696656793472698"
```

```
print(paste('rand index for k=4', randIndex(km4_table, correct=FALSE)))
```

```
## [1] "rand index for k=4 0.655872196792608"
```