ASDM Workshop Week 4 : Clustering with R

- Dr. Mo Saraee, Charith Silva

Clustering analysis is widely used in many fields. Traditionally clustering is regarded as unsupervised learning for its lack of a class label or a quantitative response variable, which in contrast is present in supervised learning such as classification and regression.

Why Clustering analysis is unsupervised? Because, the target variable is not present. The model is trained based on given input variables which attempt to discover intrinsic groups (or clusters).

Clustering is a division of data into groups of similar objects. Representing the data by fewer clusters necessarily loses certain fine details, but achieves simplification. It models data by its clusters. Data modeling puts clustering in a historical perspective rooted in mathematics, statistics, and numerical analysis. From a machine learning perspective clusters correspond to hidden patterns, the search for clusters is unsupervised learning, and the resulting system represents a data concept. From a practical perspective clustering plays an outstanding role in data mining applications such as scientific data exploration, information retrieval and text mining, spatial database applications, Web analysis, CRM, marketing, medical diagnostics, computational biology (Berkhin,2006).

Some other applications of clustering are:

- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs.
- Land use: Identification of areas of similar land use in an earth observation database.
- Insurance: Identifying groups of motor insurance policy holders with a high average claim cost.
- City-planning: Identifying groups of houses according to their house type, value, and geographical location.

Part 1: Exercise

We will be using **costpercompany.csv** dataset in this workshop which can be downloaded from Blackboard. The dataset contains several different financial indicators about the given companies.

Data Explanation

Variable	Explanation
Company	Name of the company
surcharges	Surcharge cost per day
RoR	Return on Risk
dailycost	Daily miscellaneous cost
costwithload	Cost related to loading
costofDemand	Demand cost
Sales	Sales of company
WearandTear	Natural wear and tear cost
Fcost	Fuel cost

Implementation:

In this workshop, we will be using "cluster" packages which use K-means clustering Algorithm. K-means (MacQueen, 1967) is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters). We will also be using hierarchical clustering for basic understanding.

1. Download **costpercompany.csv** dataset from Blackboard and save it to a folder on your F: drive (eg: F:\ASDM\Week4). Open it using excel to get a rough idea about the dataset, e.g, attributes and their values.

2. Start RStudio.

3. Change the working directory
   File → More → Go To Working Directory…

In the Go To Working Directory dialogue, navigate and select the folder where you saved your data file eg: F:\ASDM\Week4. Click OK.

Click Set as Working Directory option

or

using R commands as follow:

```
mypath = "F:\\ASDM\\Week4"    # you need to change the string to your directory

setwd(mypath)                 # set working directory
getwd()                       # check if the working directory has changed correctly
```

4. Open a new R script window:
   File → New File → R script

5. Read the data file

```
cost_data <- read.csv("costpercompany.csv", header= TRUE)
```

The data is read into the data frame named "cost_data"

6. Inspect the dataset in R

Once the file has been imported to R, we often want to do few things to explore the dataset:

```
names(cost_data)
head(cost_data)
tail(cost_data)
summary(cost_data)
str(cost_data)
```

```
> tail(cost_data)
       Company surcharges   RoR dailycost costwithload costofDemand Sales WearandTear Fcost
24 Commonwealth       1.02 11.20       168         56.0          0.3  6423        34.3 0.700
25     Central        1.43 15.40       113         53.0          3.4  9212         0.0 1.058
26       CA Gas       1.95  1.86        49        -39.4          6.4 15280        12.3 1.140
27           BP       3.90 21.16       370         72.0         16.4 40008        53.4 2.610
28      Boston        0.89 10.30       202         57.9          2.2  5088        25.3 1.555
29     Arizona        1.06  9.20       151         54.4          1.6  9077         0.0 0.628
> summary(cost_data)
         Company    surcharges         RoR          dailycost      costwithload     costofDemand        Sales        WearandTear        Fcost       
 Arizona     : 1   Min.   :0.750   Min.   : 1.86   Min.   : 49.0   Min.   :-49.80   Min.   :-2.200   Min.   : 3300   Min.   : 0.00   Min.   :-0.012  
 Boston      : 1   1st Qu.:1.050   1st Qu.: 9.20   1st Qu.:148.0   1st Qu.: 51.50   1st Qu.: 2.200   1st Qu.: 6650   1st Qu.: 0.00   1st Qu.: 0.636  
 BP          : 1   Median :1.150   Median :10.58   Median :173.0   Median : 56.00   Median : 3.500   Median : 9673   Median : 8.30   Median : 1.108  
 CA Gas      : 1   Mean   :1.403   Mean   :10.49   Mean   :172.4   Mean   : 46.86   Mean   : 4.907   Mean   :13025   Mean   :16.11   Mean   : 1.174  
 Central     : 1   3rd Qu.:1.430   3rd Qu.:12.20   3rd Qu.:199.0   3rd Qu.: 60.00   3rd Qu.: 7.200   3rd Qu.:15651   3rd Qu.:26.70   3rd Qu.: 1.652  
 Commonwealth: 1   Max.   :3.900   Max.   :21.16   Max.   :370.0   Max.   : 72.00   Max.   :16.400   Max.   :40008   Max.   :53.40   Max.   : 2.610  
 (Other)     :23                                                                                                                                       
> str(cost_data)
'data.frame':	29 obs. of  9 variables:
 $ Company     : Factor w/ 29 levels "Arizona ","Boston ",..: 29 28 27 26 25 24 23 22 21 20 ...
 $ surcharges  : num  2.7 1.2 1.07 1.04 1.16 1.05 1.95 0.76 1.16 0.96 ...
 $ RoR         : num  9.36 11.8 9.3 8.6 11.7 ...
 $ dailycost   : int  222 148 174 204 104 150 185 136 252 164 ...
 $ costwithload: num  12.1 59.9 54.3 61 54 56.7 36 61.9 56 62.2 ...
 $ costofDemand: num  12.9 3.5 5.9 3.5 -2.1 2.7 8.2 9 9.2 -0.1 ...
 $ Sales       : int  32721 7287 10093 6650 13507 10140 20004 5714 15991 6468 ...
 $ WearandTear : num  12.3 41.1 26.6 0 0 0 26.7 8.3 0 0.9 ...
 $ Fcost       : num  1.908 0.702 1.306 2.116 0.636 ...
```

7. Check the dimension and number of points of the "cost_data" dataset

```
nrow(cost_data)
ncol(cost_data)
dim(cost_data)
```

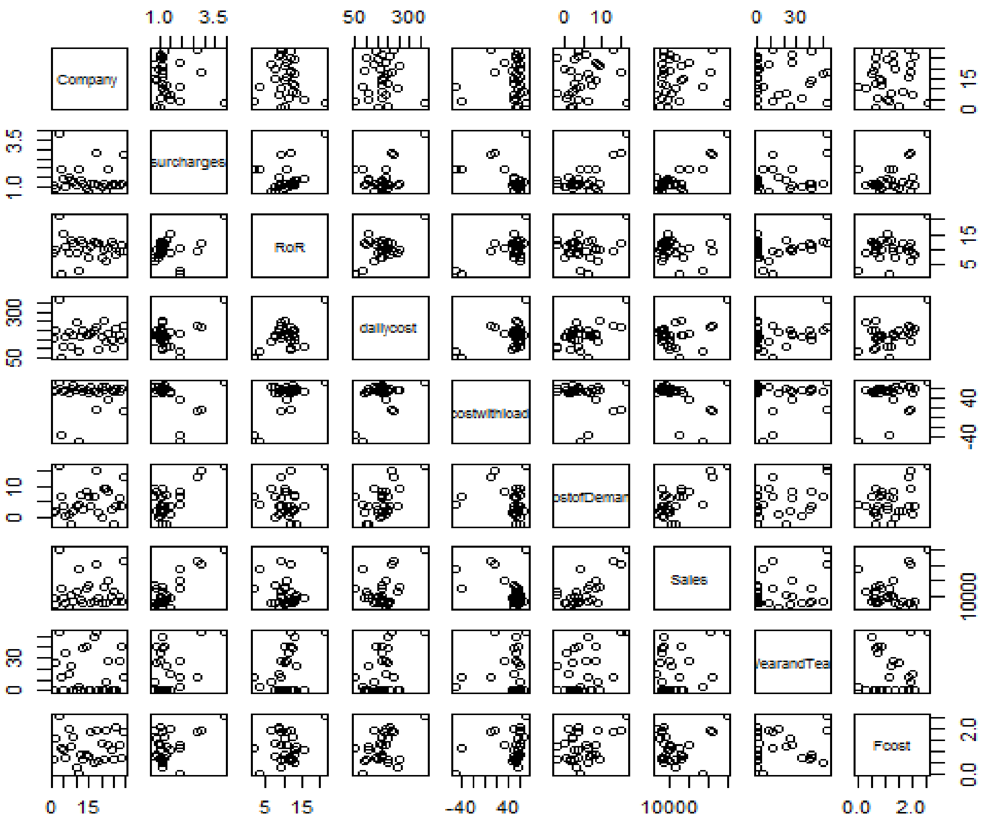
```
> nrow(cost_data)
[1] 29
> ncol(cost_data)
[1] 9
> dim(cost_data)
[1] 29  9
```

8. Install and activate "cluster" package.

```
#cluster package is a powerful tool for cluster analysis

install.packages("cluster")        # install "cluster" package
library(cluster)                   # activate "cluster" package
```

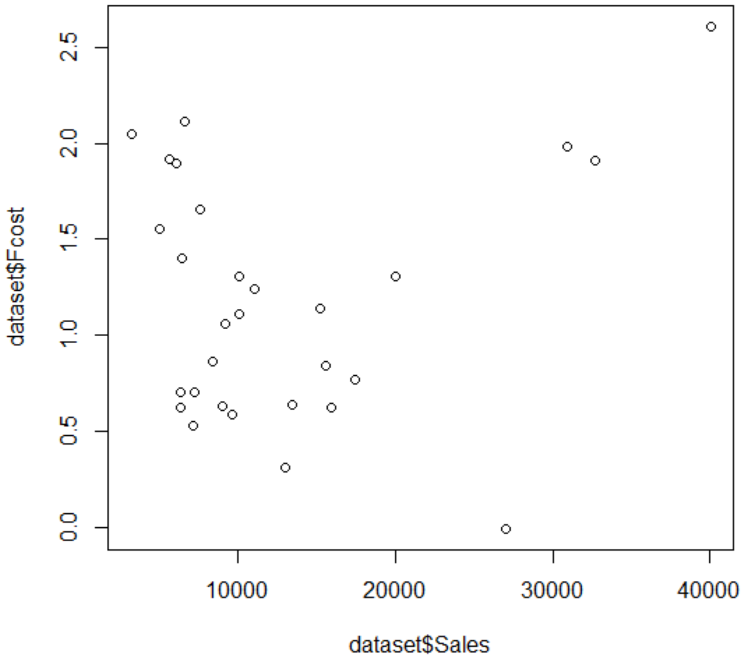
9. Following command can be used to create scatterplot matrix to compare the variables

```
pairs(cost_data)
```



10. Use the following line of code to plot and understand the relationship between the Sales and Fuel cost of the company:

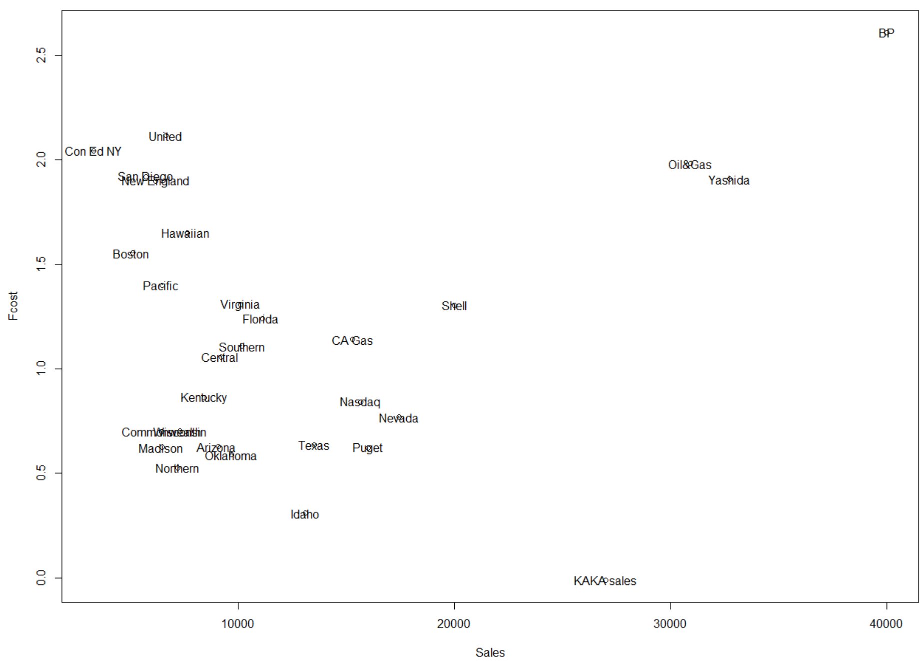
```
plot(Fcost~ Sales, data = cost_data)
```



11. The result does not tell you anything about the companies related to each cluster. Hence, using the following line of code to see the names of the companies:

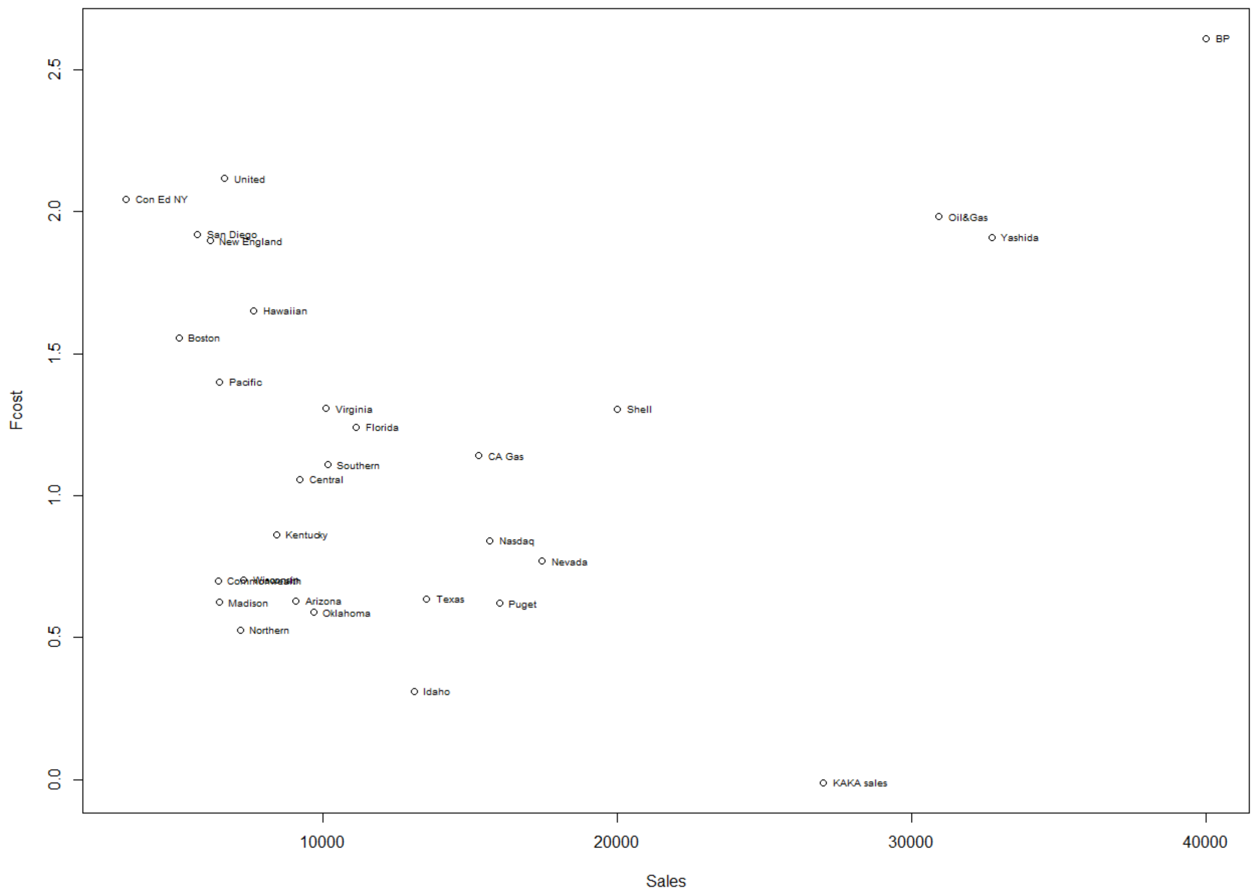
```
with(cost_data,text(Fcost ~ Sales, labels= Company))
```

Result:



12. Remove the overlap of names of the companies:

```
plot(Fcost~ Sales, data = cost_data)
with(cost_data,text(Fcost ~ Sales, labels= Company,pos=4,cex=.6))
```



13. Normalization of the dataset is among the preprocessing processes in data exploration, in which the attribute data are scaled to fall in a small specified range.

    In this dataset some variables are in decimal points (Fuel-cost) and some are with very high values like thousands (Sales), so these values will be dominating in the analysis.

    Normalization before clustering is specifically needed for distance matrix, like the Euclidian distance that are sensitive to variations within the magnitude or scales from the attributes.

```
#normalise function

normalise <- function(df)
{
  return(((df- min(df)) /(max(df)-min(df))*(1-0))+0)
}
```

```
head(cost_data)
```

```
> head(cost_data)
    Company surcharges   RoR dailycost costwithload costofDemand Sales WearandTear Fcost
1   Yashida       2.70  9.36       222         12.1         12.9 32721        12.3 1.908
2 Wisconsin       1.20 11.80       148         59.9          3.5  7287        41.1 0.702
3  Virginia       1.07  9.30       174         54.3          5.9 10093        26.6 1.306
4    United       1.04  8.60       204         61.0          3.5  6650         0.0 2.116
5     Texas       1.16 11.70       104         54.0         -2.1 13507         0.0 0.636
6  Southern       1.05 12.60       150         56.7          2.7 10140         0.0 1.108
```

```
#Normalise the dataset using above normalise function

company<-cost_data[,1]
cost_data_n<-cost_data[,2:9] #remove company column before normalise
cost_data_n<-as.data.frame(lapply(cost_data_n,normalise))
cost_data_n$Company<-company #add company column after normalise

#rearrange the columns in the dataset after normalising
cost_data_n<-cost_data_n[,c(9,1,2,3,4,5,6,7,8)]

head(cost_data_n)
```

```
> head(cost_data_n)
    Company surcharges       RoR dailycost costwithload costofDemand      Sales WearandTear     Fcost
1   Yashida 0.61904762 0.3886010 0.5389408    0.5082102  0.811827957 0.80148741   0.2303371 0.7322654
2 Wisconsin 0.14285714 0.5150259 0.3084112    0.9006568  0.306451613 0.10861393   0.7696629 0.2723112
3  Virginia 0.10158730 0.3854922 0.3894081    0.8546798  0.435483871 0.18505503   0.4981273 0.5026697
4    United 0.09206349 0.3492228 0.4828660    0.9096880  0.306451613 0.09126076   0.0000000 0.8115942
5     Texas 0.13015873 0.5098446 0.1713396    0.8522167  0.005376344 0.27805928   0.0000000 0.2471396
6  Southern 0.09523810 0.5564767 0.3146417    0.8743842  0.263440860 0.18633540   0.0000000 0.4271548
```

14. Since we have normalized our data, now we can calculat the distance matrix using Euclidean Distance. The choice of distance measures is very important, as it has a strong influence on the clustering results. For most common clustering the default distance measure is the Euclidean distance.

```
#Dist() function computes and returns the distance matrix computed by using the specified distance measure to compute the distances between the rows of a data matrix. Dist() function accepts only numeric data as an input.
```

#Note that, allowed values for the option **method** include one of: "euclidean", "maximum", "manhattan", "canberra", "binary" and "minkowski".

#choose distance method and create distance matrix

```
distance <- dist(cost_data_n,method = "euclidean",)
```

In this matrix, the value represents the distance between companies.

```
print(distance)
```

```
> distance
           1         2         3         4         5         6         7         8         9        10        11        12        13        14
2  1.3779637
3  1.0900706 0.4455350
4  1.1918947 1.0305857 0.6557062
5  1.3910132 0.9085196 0.8014188 0.8020101
6  1.2150186 0.8411797 0.5987688 0.5099920 0.3851189
7  0.6889819 0.6889819 0.4676341 0.8755935 0.9515602 0.7856404
8  1.1775998 0.9378339 0.5454479 0.4545513 0.9312438 0.6405588 0.8114437
9  1.0173962 0.9832388 0.6994714 0.7606013 0.8183244 0.5847719 0.7254177 0.7724816
10 1.3409603 0.9071903 0.6371691 0.3843243 0.4999180 0.3631547 0.9142693 0.5941891 0.7515437
11 1.3470049 0.8522907 0.7207823 0.7590054 0.2352620 0.2943494 0.8920285 0.8087363 0.7168157 0.4966276
12 0.8406961 1.3059485 1.2161691 1.6068993 1.7801013 1.6184633 0.8552425 1.5004362 1.4742499 1.7338989 1.7306672
13 1.4156067 0.3147726 0.6078059 1.2212377 1.1688037 1.0627062 0.7594840 1.1043016 1.0680394 1.1501769 1.0955256 1.2163865
14 1.1919754 0.9562812 0.6146599 0.1803758 0.6938388 0.3721955 0.8324323 0.4633696 0.7152954 0.3550891 0.6296577 1.5944125 1.1600842
15 1.0890280 0.9345940 0.6283006 0.6740632 0.6173063 0.4549315 0.7450020 0.6374166 0.3676625 0.5678501 0.5286573 1.5516763 1.0950898 0.6327377
```

#Round the distance figures to 3 decimals.

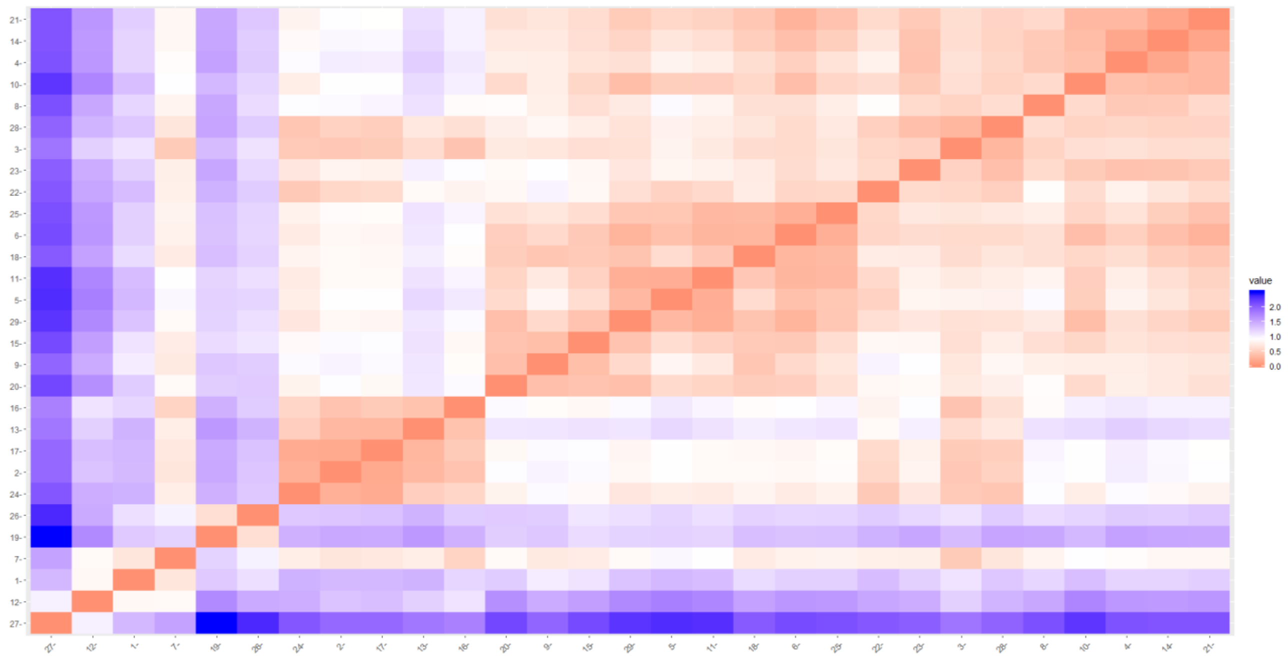
```
print(distance,digits=3)
```

```
> print(distance,digits=3)
      1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18    19    20    21    22    23    24    25    26    27
2  1.378
3  1.090 0.446
4  1.192 1.031 0.656
5  1.391 0.909 0.801 0.802
6  1.215 0.841 0.599 0.510 0.385
7  0.689 0.689 0.468 0.876 0.952 0.786
8  1.178 0.938 0.545 0.455 0.931 0.641 0.811
9  1.017 0.983 0.699 0.761 0.818 0.585 0.725 0.772
10 1.341 0.907 0.637 0.384 0.500 0.363 0.914 0.594 0.752
11 1.347 0.852 0.721 0.759 0.235 0.294 0.892 0.809 0.717 0.497
12 0.841 1.306 1.216 1.607 1.780 1.618 0.855 1.500 1.474 1.734 1.731
13 1.416 0.315 0.608 1.221 1.169 1.063 0.759 1.104 1.068 1.150 1.096 1.216
14 1.192 0.956 0.615 0.180 0.694 0.372 0.832 0.463 0.715 0.355 0.630 1.594 1.160
15 1.089 0.935 0.628 0.674 0.617 0.455 0.745 0.637 0.368 0.568 0.529 1.552 1.095 0.633
16 1.174 0.406 0.413 1.049 1.049 0.914 0.545 0.869 0.871 1.008 0.972 1.089 0.400 1.003 0.841
17 1.384 0.192 0.467 1.011 0.905 0.826 0.708 0.975 0.934 0.891 0.856 1.340 0.308 0.944 0.923 0.465
18 1.139 0.853 0.610 0.622 0.607 0.305 0.733 0.639 0.426 0.574 0.445 1.539 1.017 0.489 0.456 0.874 0.839
19 1.266 1.485 1.361 1.538 1.210 1.315 1.195 1.499 1.276 1.388 1.191 1.699 1.596 1.507 1.135 1.434 1.481 1.340
20 1.244 0.919 0.726 0.768 0.591 0.497 0.850 0.878 0.372 0.594 0.552 1.664 1.060 0.716 0.406 0.934 0.847 0.483 1.235
21 1.224 0.913 0.624 0.311 0.573 0.272 0.831 0.581 0.695 0.323 0.534 1.619 1.132 0.166 0.616 0.993 0.897 0.438 1.493 0.644
22 1.349 0.572 0.572 0.789 0.529 0.556 0.774 0.884 0.983 0.610 0.580 1.506 0.856 0.691 0.842 0.800 0.601 0.744 1.423 0.838 0.596
23 1.218 0.805 0.528 0.398 0.813 0.612 0.769 0.598 0.915 0.468 0.792 1.467 1.008 0.411 0.836 0.923 0.789 0.735 1.505 0.858 0.456 0.597
24 1.425 0.250 0.457 0.928 0.757 0.729 0.757 0.921 0.940 0.751 0.735 1.454 0.496 0.861 0.862 0.558 0.211 0.806 1.437 0.796 0.802 0.452 0.703
25 1.218 0.867 0.691 0.664 0.440 0.242 0.802 0.754 0.700 0.556 0.318 1.600 1.088 0.497 0.618 0.972 0.872 0.331 1.329 0.651 0.398 0.571 0.709 0.793
26 1.118 1.287 1.095 1.250 1.182 1.174 0.989 1.135 1.239 1.184 1.113 1.475 1.419 1.237 1.068 1.246 1.319 1.200 0.637 1.256 1.278 1.242 1.170 1.271 1.182
27 1.388 1.932 1.856 2.085 2.320 2.130 1.534 2.103 1.957 2.271 2.302 0.999 1.832 2.071 2.137 1.768 1.933 2.033 2.486 2.159 2.069 2.045 2.003 2.057 2.098 2.341
28 1.278 0.526 0.304 0.569 0.794 0.597 0.694 0.619 0.831 0.545 0.752 1.407 0.711 0.543 0.761 0.641 0.492 0.693 1.518 0.770 0.546 0.505 0.368 0.427 0.715 1.247 1.968
29 1.306 0.841 0.649 0.643 0.324 0.279 0.855 0.727 0.585 0.358 0.243 1.710 1.066 0.554 0.406 0.933 0.819 0.412 1.197 0.370 0.473 0.629 0.703 0.701 0.434 1.115 2.274
```

15. Visualise the distance matrices using a function called **fviz_dist()** in **factoextra** package

```
install.packages("factoextra")        # install "factoextra" package
library(factoextra)                   # activate "factoextra" package
```

```
fviz_dist(distance)
```



The colour level is proportional to the value of the dissimilarity between observations:

Red: high similarity (ie: low dissimilarity)
Blue: low similarity

16. Add company name to the distance matrix

```
#inspect the first few observations
head(cost_data_n)
```

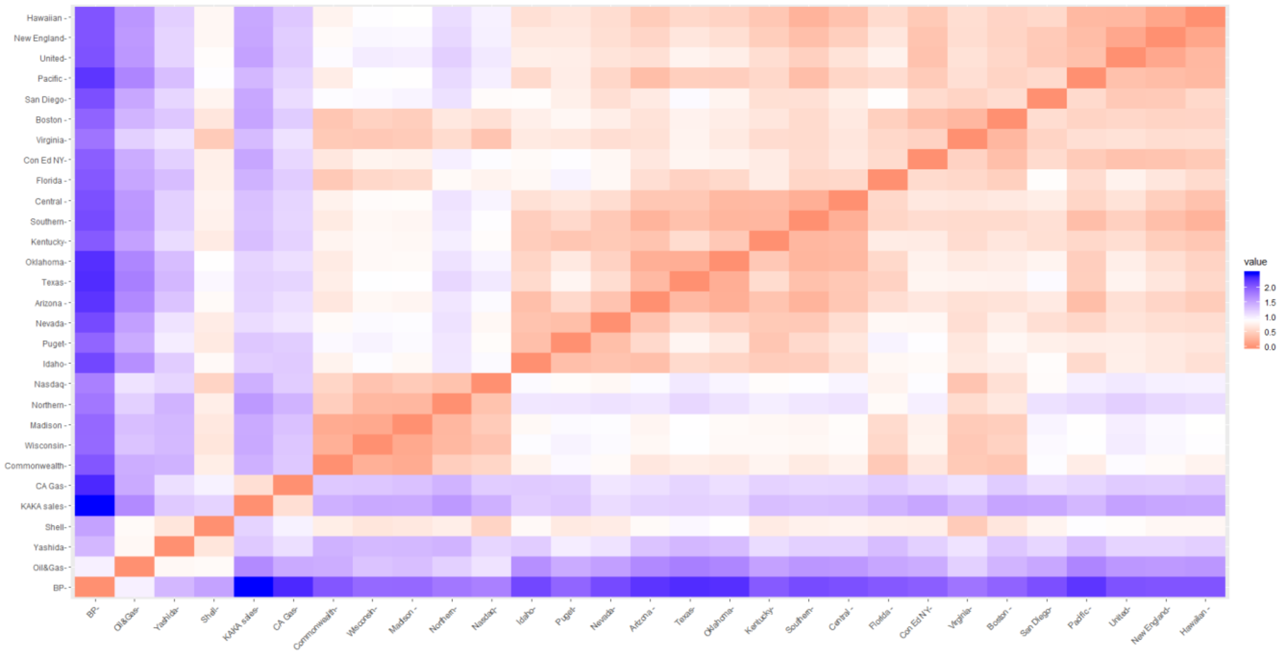
```
> head(cost_data_n)
    Company surcharges       RoR dailycost costwithload costofDemand      Sales WearandTear     Fcost
1   Yashida 0.61904762 0.3886010 0.5389408    0.5082102  0.811827957 0.80148741   0.2303371 0.7322654
2 Wisconsin 0.14285714 0.5150259 0.3084112    0.9006568  0.306451613 0.10861393   0.7696629 0.2723112
3  Virginia 0.10158730 0.3854922 0.3894081    0.8546798  0.435483871 0.18505503   0.4981273 0.5026697
4    United 0.09206349 0.3492228 0.4828660    0.9096880  0.306451613 0.09126076   0.0000000 0.8115942
5     Texas 0.13015873 0.5098446 0.1713396    0.8522167  0.005376344 0.27805928   0.0000000 0.2471396
6  Southern 0.09523810 0.5564767 0.3146417    0.8743842  0.263440860 0.18633540   0.0000000 0.4271548
```

```
#rownames() function retrieve the row names of a matrix-like object.
#Set company names as row names
rownames(cost_data_n)<-cost_data_n$Company

#remove Company column from the dataset
cost_data_n$Company<-NULL
```

```
#inspect the top observations
head(cost_data_n)
```

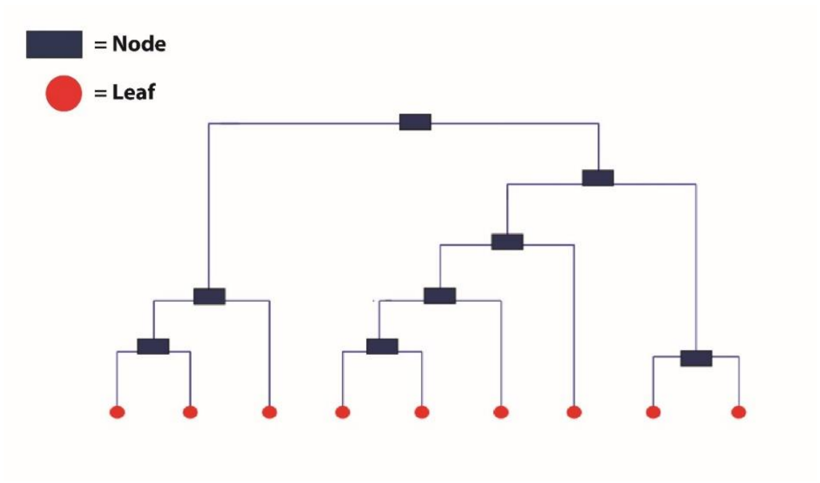
```
> head(cost_data_n)
          surcharges       RoR dailycost costwithload costofDemand      Sales WearandTear     Fcost
Yashida   0.61904762 0.3886010 0.5389408    0.5082102  0.811827957 0.80148741   0.2303371 0.7322654
Wisconsin 0.14285714 0.5150259 0.3084112    0.9006568  0.306451613 0.10861393   0.7696629 0.2723112
Virginia  0.10158730 0.3854922 0.3894081    0.8546798  0.435483871 0.18505503   0.4981273 0.5026697
United    0.09206349 0.3492228 0.4828660    0.9096880  0.306451613 0.09126076   0.0000000 0.8115942
Texas     0.13015873 0.5098446 0.1713396    0.8522167  0.005376344 0.27805928   0.0000000 0.2471396
Southern  0.09523810 0.5564767 0.3146417    0.8743842  0.263440860 0.18633540   0.0000000 0.4271548
```

```
distance <- dist(cost_data_n,method = "euclidean")
fviz_dist(distance)
```



17. Hierarchical clustering

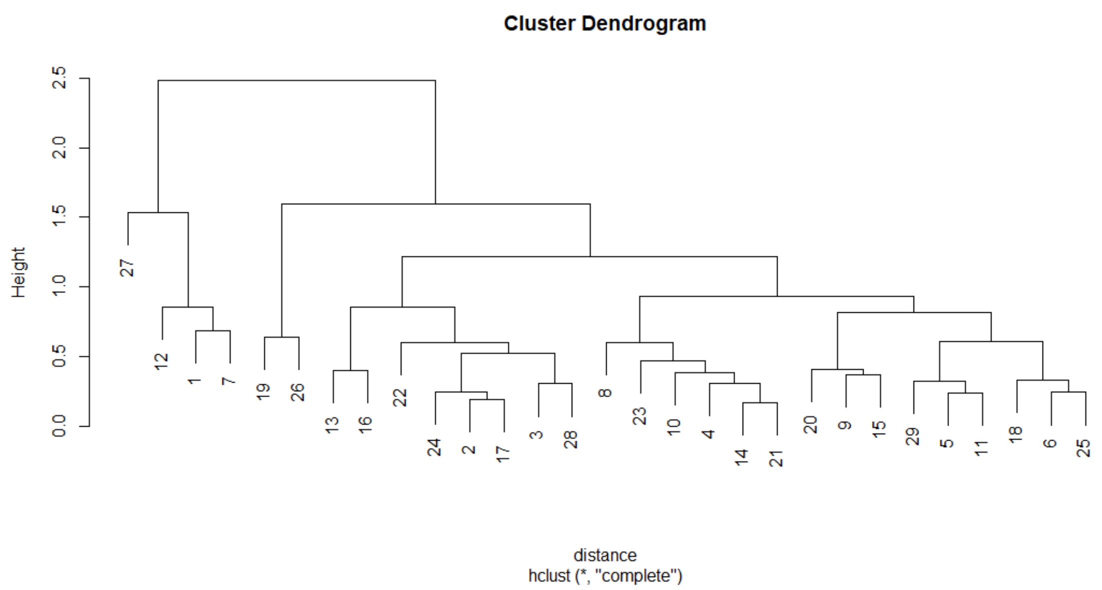
In simple words, hierarchical clustering tries to create a sequence of nested clusters to explore deeper insights from the data. Most commonly used hierarchical clustering method is **Agglomerative Clustering.** It starts with treating every observation as a cluster. Then, it merges the most similar observations into a new cluster. This process continues until all the observations are merged into one cluster. It uses a bottoms-up approach. This hierarchy of clusters is graphically presented using a Dendogram (shown below).



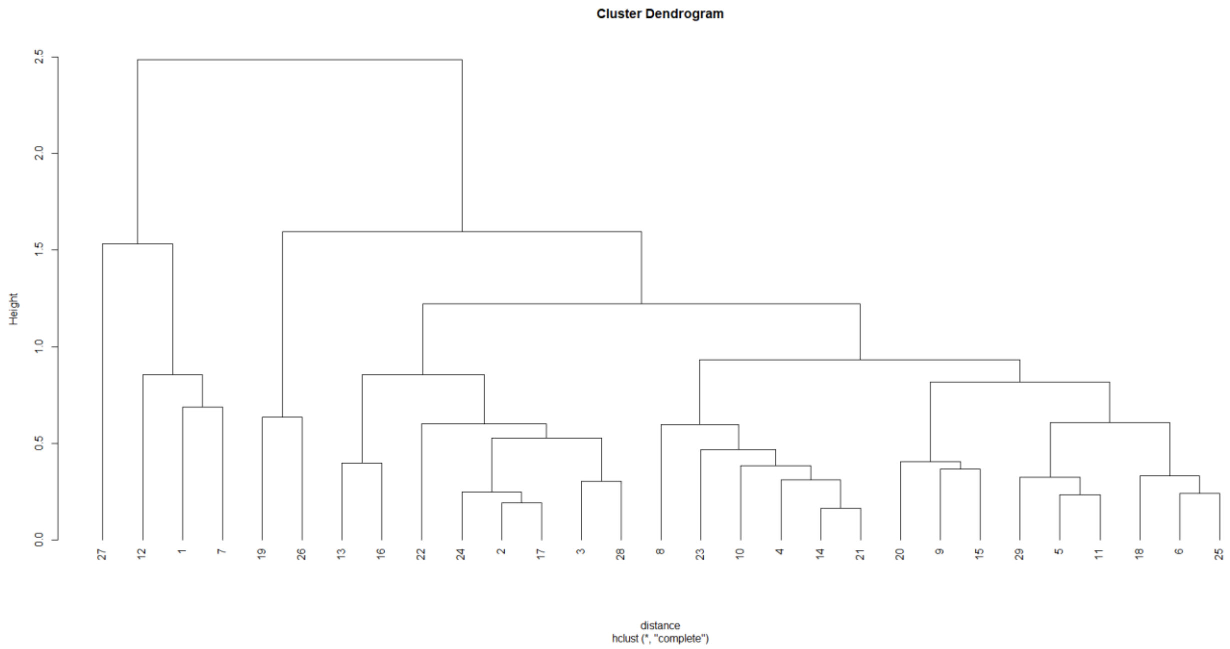

```
#choose previously created intercluster distance matrix and
perform Hierarchical clustering using hclust() function

cost_data.hclust <- hclust(distance)
cost_data.hclust
```

```
> cost_data.hclust <- hclust(distance)
> cost_data.hclust

Call:
hclust(d = distance)

Cluster method   : complete
Distance         : euclidean
Number of objects: 29
```

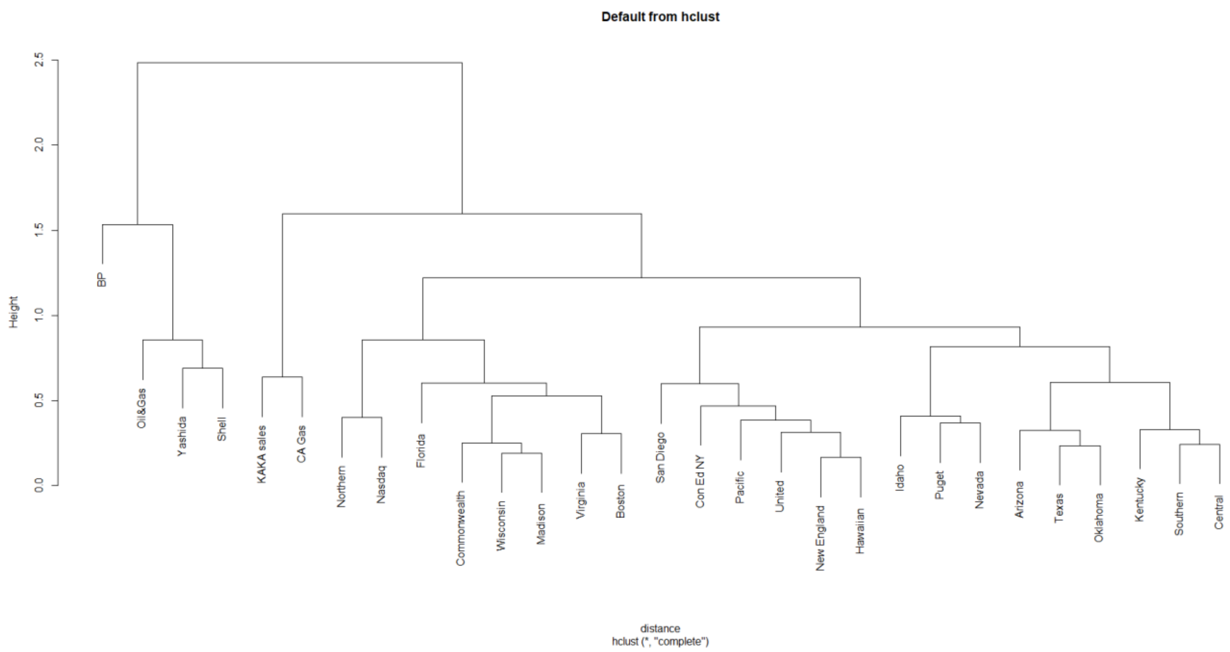
```
plot(cost_data.hclust) # plot the results
```



```
plot(cost_data.hclust,hang=-1)
```




```
plot(cost_data.hclust,labels=cost_data$Company)
```

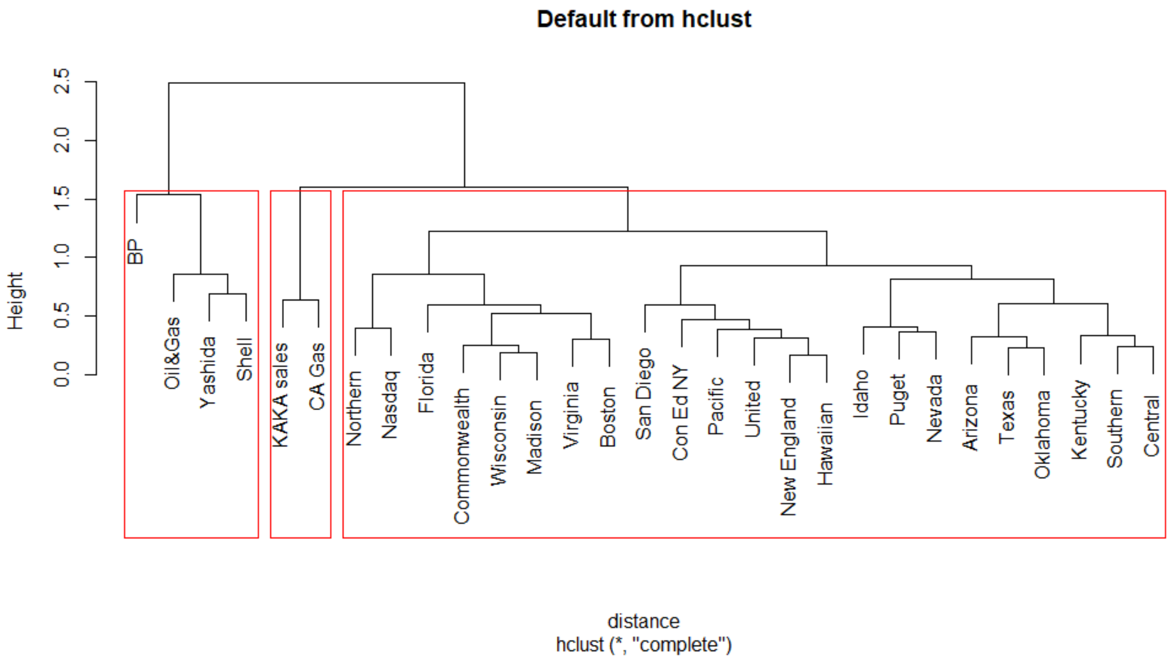


18. plot clusters for chosen number of clusters

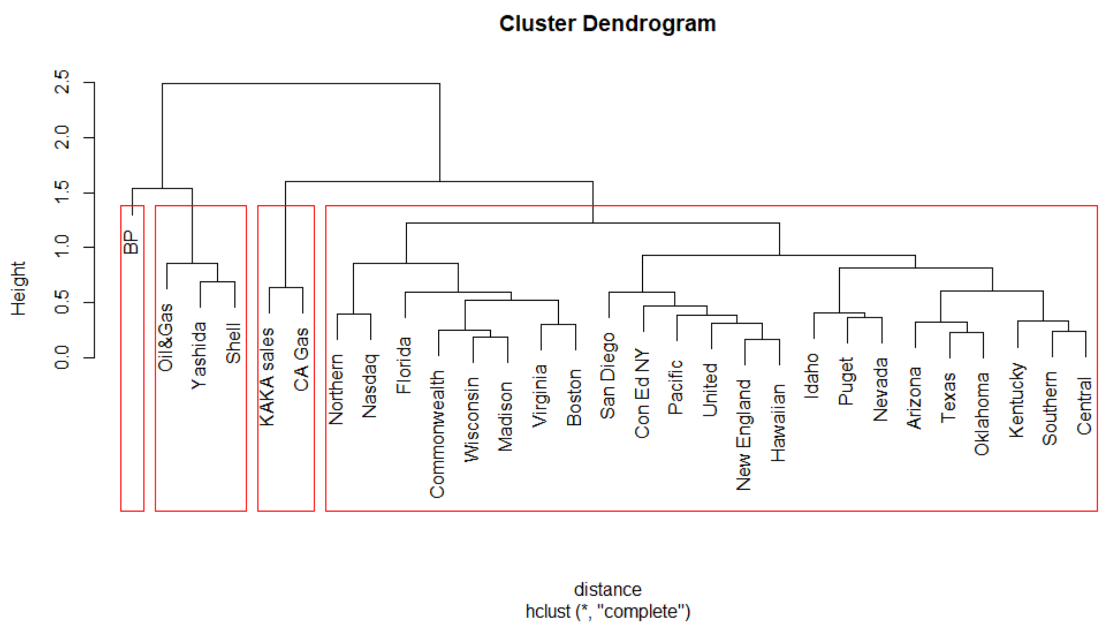
```
# rect.hclust() function draws rectangles around the branches of a
dendrogram highlighting the corresponding clusters. First the
```

dendrogram is cut at a certain level, then a rectangle is drawn around selected branches.

```
# draw 3 clusters
rect.hclust(cost_data.hclust, 3)
```



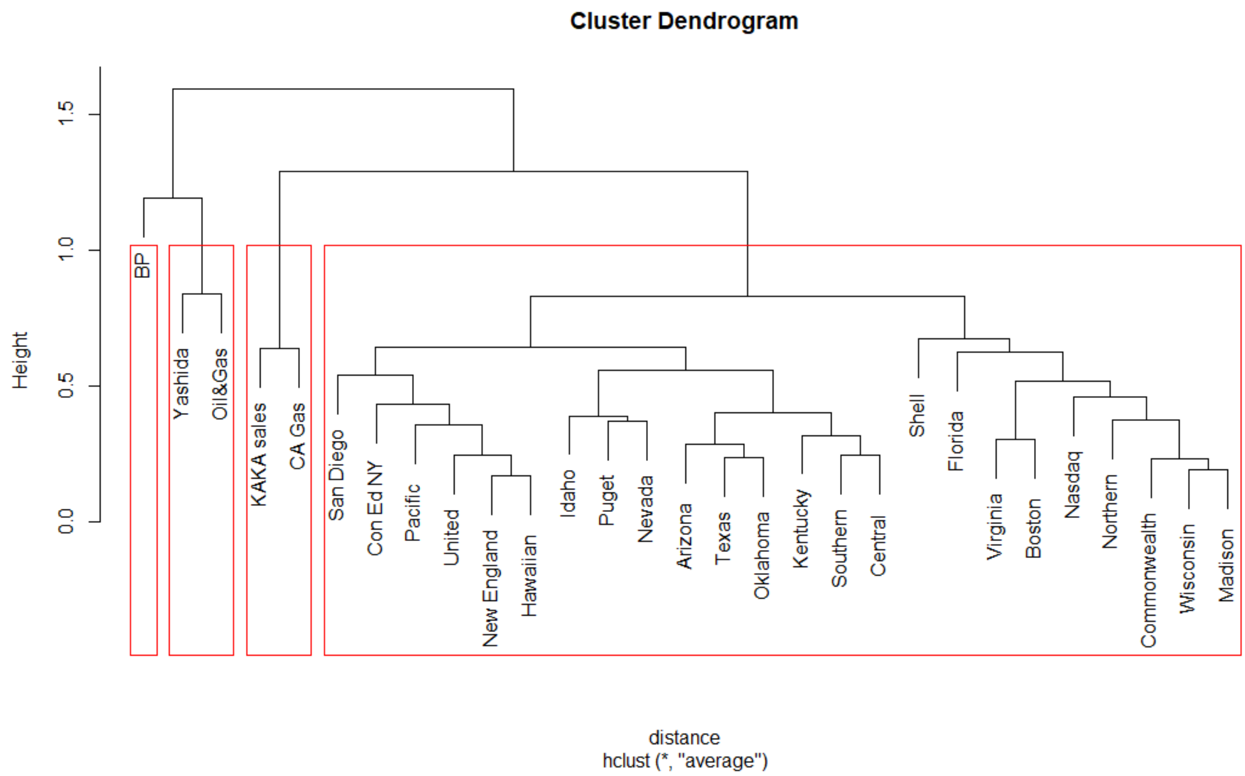

```
# draw 4 clusters
  plot(cost_data.hclust,labels=cost_data$Company)
  rect.hclust(cost_data.hclust, 4)
```



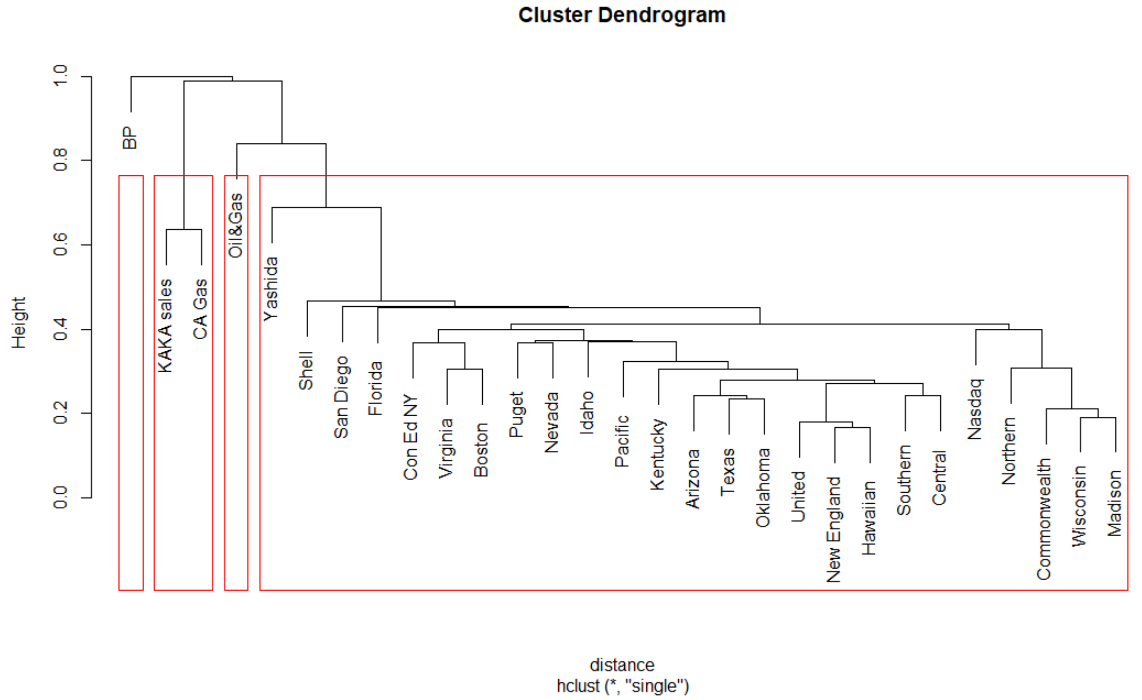
19. There are a few ways to determine how close two clusters are:

- **Complete linkage clustering**: Find the maximum possible distance between points belonging to two different clusters.
- **Single linkage clustering**: Find the minimum possible distance between points belonging to two different clusters.
- **Average linkage clustering:** Find all possible pairwise distances for points belonging to two different clusters and then calculate the average.
- **Centroid linkage clustering:** Find the centroid of each cluster and calculate the distance between centroids of two clusters.

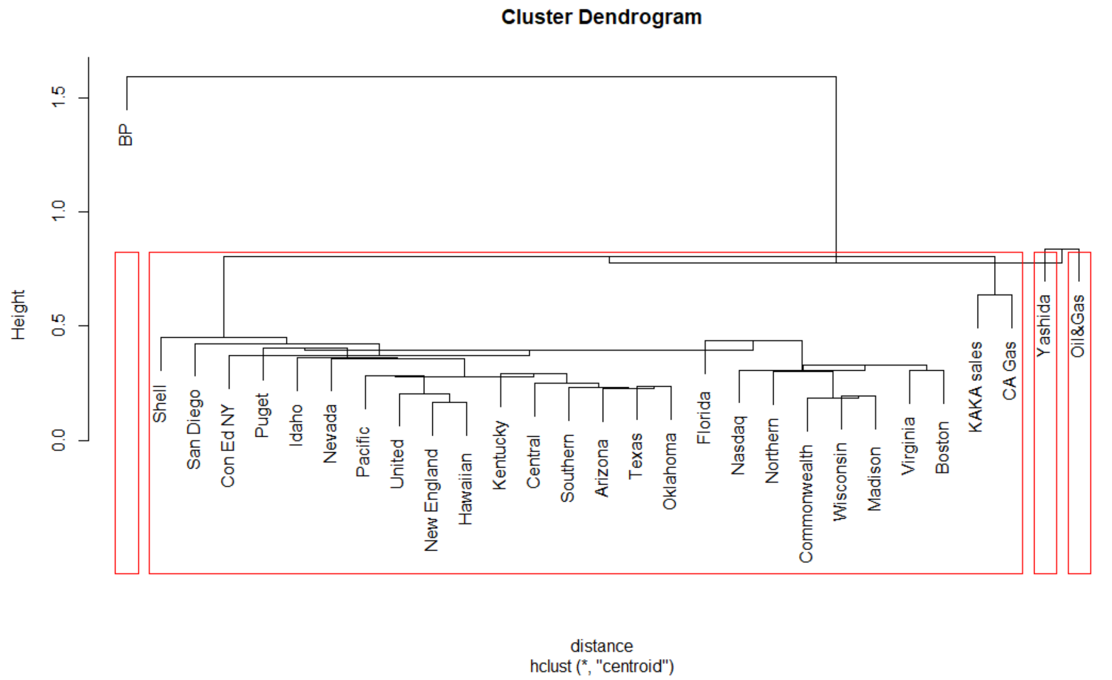
```
# Cluster using average linkage
hclust.average <- hclust(distance, method = "average")
plot(hclust.average,labels=cost_data$Company)
rect.hclust(hclust.average, 4)
```



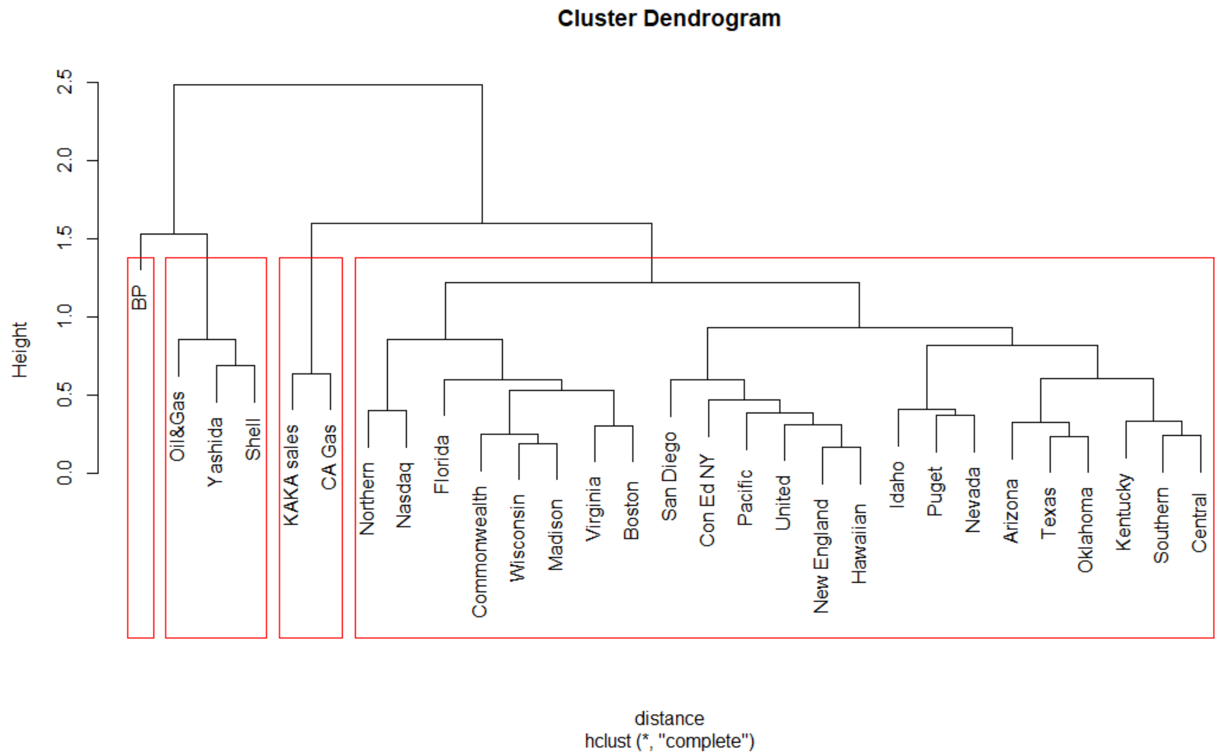
```
# Cluster using single linkage
hclust.single <- hclust(distance, method = "single")
plot(hclust.single,labels=cost_data$Company)
rect.hclust(hclust.single, 4)
```




```
# Cluster using centroid linkage
hclust.centroid<- hclust(distance, method = "centroid")
plot(hclust.centroid,labels=cost_data$Company)
rect.hclust(hclust.centroid, 4)
```



```
# Cluster using complete linkage
hclust.complete <- hclust(distance, method = "complete")
plot(hclust.complete,labels=cost_data$Company)
rect.hclust(hclust.complete, 4)
```



20. Compare the cluster membership

```
#cutree function() cuts a dendrogram tree into several groups by
specifying the desired number of clusters.

member.centroid <- cutree(hclust.centroid,4)
member.centroid
member.complete <- cutree(hclust.complete,4)
member.complete

table(member.centroid,member.complete)
```

```
> table(member.centroid,member.complete)
               member.complete
member.centroid  1  2  3  4
              1  1  0  0  0
              2  1 23  2  0
              3  1  0  0  0
              4  0  0  0  1
>
```

21. K-Means clustering

K-means clustering is a type of unsupervised learning technique, which is used when you have unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of *K* groups based on the features that are provided. Data points are clustered based on feature similarity.

K-Means clustering algorithm can be used for any type of grouping. Some examples of use cases are: (source : Oracle data science)

- Behavioural segmentation:
    - Segment by purchase history
    - Segment by activities on application, website, or platform
    - Define personas based on interests
    - Create profiles based on activity monitoring

- Inventory categorization:
    - Group inventory by sales activity
    - Group inventory by manufacturing metrics

- Sorting sensor measurements:
    - Detect activity types in motion sensors
    - Group images
    - Separate audio
    - Identify groups in health monitoring

- Detecting bots or anomalies:
    - Separate valid activity groups from bots
    - Group valid activity to clean up outlier detection

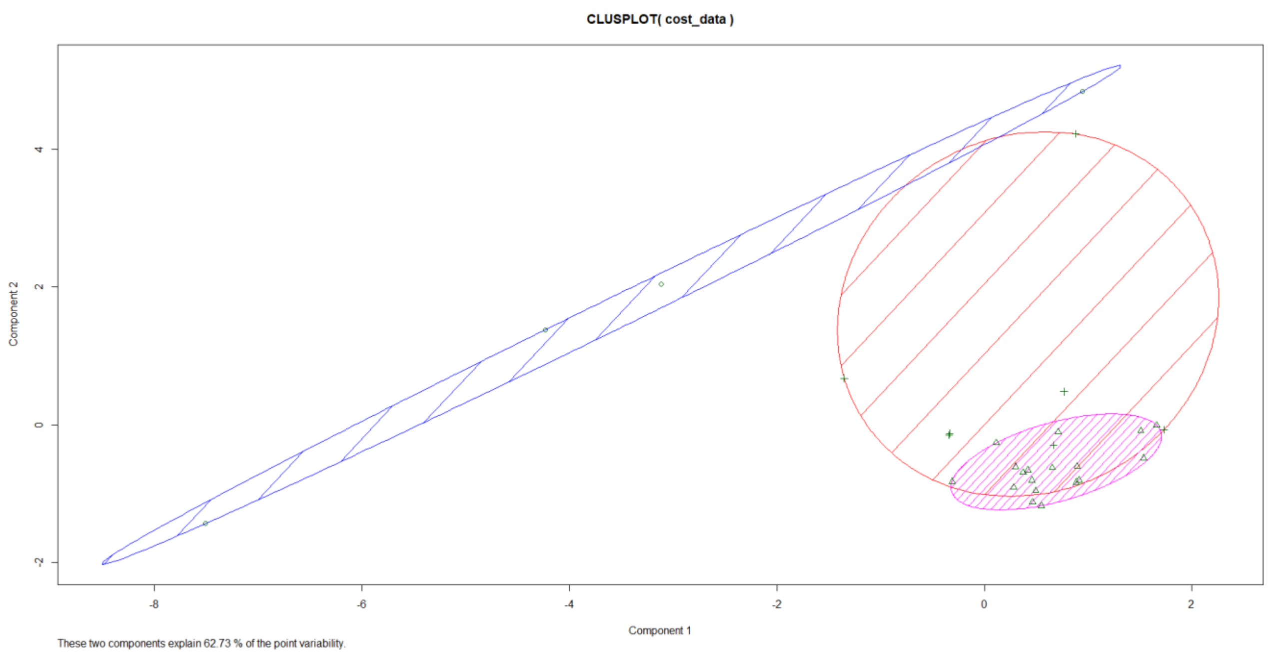
```
Use the following line of code to create k means clustering:

#k-means clustering is an algorithm used to find homogeneous
subgroups in a population
```

```
#Defining the optimal number of cluster as k-means clustering
requires to specify the number of clusters to generate.

#kmeans function perform k-means clustering on a data matrix.
kc<-kmeans(cost_data[,-1],3) #k=3
kc
```

```
> kc<-kmeans(cost_data[,-1],3)
> kc
K-means clustering with 3 clusters of sizes 4, 7, 18

Cluster means:
  surcharges       RoR dailycost costwithload costofDemand     Sales WearandTear     Fcost
1      2.845 11.360000  224.2500     12.97500    12.000000 32666.750    30.77500 1.6220000
2      1.280  8.691429  168.2857     38.87143     5.700000 15850.857    11.44286 0.8028571
3      1.130 10.994444  162.5556     57.50000     3.022222  7560.444    14.66667 1.2181667

Clustering vector:
 [1] 1 3 3 3 2 3 2 3 2 3 3 1 3 3 2 2 3 3 1 2 3 3 3 3 3 2 1 3 3

Within cluster sum of squares by cluster:
[1] 88992681 33363677 70434368
 (between_SS / total_SS =  91.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
```

22. The result can be plotted using the clusplot() function

```
clusplot(cost_data, kc$cluster, color=TRUE, shade=TRUE,  lines=0)
```



Part 2: Exercise

In this exercise we will practice how to implement K-Means clustering in real-world dataset.

The dataset, **crimes-2017-18.csv** can be downloaded from Blackboard. Dataset contains quarterly statistics on various crime figures in England and Wales.

Source : https://www.gov.uk/government/statistics/police-recorded-crime-open-data-tables

Dataset Explanation

FINANCIAL YEAR	
Possible values	Various
Combined with the **Financial Quarter** column, this identifies the period during which offences took place. Each financial year runs from April to March.	
FINANCIAL QUARTER	
Possible values	Various

Combined with **Financial Year** column, this identifies the period during which offences took place. Quarter 1 runs from April-June, Quarter 2 from July-September, Quarter 3 from October-December, and Quarter 4 from January-March.	
FORCE NAME	
Possible values	Various
This column identifies the police force area in which offences took place. The reference table 'PRC Geog reference table.csv' shows how these areas map up to regions within England and Wales.	
CSP NAME (CSP tables only)	
Possible values	Various
This column identifies the Community Safety Partnership in which offences took place. This is a geographic area within a Police Force, and generally corresponds to Local Authority boundaries. The reference table 'PRC Geog reference table.csv' shows how these areas map up to Police Forces and regions within England and Wales.	
OFFENCE DESCRIPTION	
Possible values	Various
This column provides a description of the offence covered by each **Offence Code** value.	
OFFENCE GROUP	
Possible values	Various
This column identifies the offence group within which the **Offence Code** falls. Each groups also consists of **Offence Sub-groups**, which in turn consist of **Offence Codes**.	

OFFENCE SUBGROUP	
Possible values	Various
This column identifies the offence sub-group within which the **Offence Code** falls. These sub-groups contain **Offence Codes.**	
OFFENCE CODE	
Possible values	Various
This column identifies the specific offence code used by the police and the Home Office to classify offences. The reference table 'Ref-Offence.csv' shows descriptions of these codes, as well as the offence groups that they map up to.	
NUMBER OF OFFENCES	
Possible values	Various
This column contains the total number of police recorded crimes for the specified **Offence Code**, **CSP Name/Force Name** and time period (**Financial Year** and **Financial Quarter**).	

1. Read the data file

```
crimes <- read.csv("crimes-2017-18.csv", header= TRUE)
```

2. Inspect the dataset

```
names(crimes)
head(crimes)
tail(crimes)
summary(crimes)
str(crimes)

nrow(crimes)
ncol(crimes)
dim(crimes)
```

```
> str(crimes)
'data.frame':   191260 obs. of  9 variables:
 $ Financial.Year     : Factor w/ 1 level "2017/18": 1 1 1 1 1 1 1 1 1 1 ...
 $ Financial.Quarter  : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Force.Name         : Factor w/ 43 levels "Avon & Somerset",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ CSP.Name           : Factor w/ 317 levels "Adur","Allerdale",..: 14 14 14 14 14 14 14 14 14 14 ...
 $ Offence.Description: Factor w/ 131 levels "Absconding from lawful custody",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ Offence.Group      : Factor w/ 9 levels "Criminal damage and arson",..: 3 7 7 8 8 8 8 8 3 1 ...
 $ Offence.Subgroup   : Factor w/ 25 levels "Arson","Bicycle theft",..: 7 9 9 8 8 5 5 21 7 1 ...
 $ Offence.Code       : Factor w/ 131 levels "1      ","104     ",..: 105 100 102 48 47 41 42 56 103 82 ...
 $ Number.of.offences : int  0 0 0 0 0 0 5 2 0 4 ...
> nrow(crimes)
[1] 191260
> ncol(crimes)
[1] 9
> dim(crimes)
[1] 191260      9
```

3. Select 3 columns for the cluster analysis

```
crimes<-crimes[,c(3,7,9)]
head(crimes)
```

```
> head(crimes)
         Force.Name       Offence.Subgroup Number.of.offences
1 Avon & Somerset  Miscellaneous crimes                  0
2 Avon & Somerset Other sexual offences                  0
3 Avon & Somerset Other sexual offences                  0
4 Avon & Somerset Non-domestic burglary                  0
5 Avon & Somerset Non-domestic burglary                  0
6 Avon & Somerset     Domestic burglary                  0
>
```

4. Transform data

reshape2 is an R package written by Hadley Wickham that makes it easy to transform data between wide and long formats.

```
install.packages("reshape2")  # install "reshape2" package
library(reshape2)             # activate "reshape2" package
```

dcast() function can be used to convert from a long format to a wide format.

Useage: The dcast() function takes three arguments:

- data: A molten data frame.
- formula: A formula that specifies how you want to cast the data. This formula takes the form x_variable ~ y_variable.
- fun.aggregate: A function to use if the casting formula results in data aggregation (for example, length(), sum(), or mean()).

```
crimes_pivot <- dcast(crimes, Force.Name ~ Offence.Subgroup, sum,
                      value.var ="Number.of.offences")
head(crimes_pivot))
```

```
       Force.Name Arson Bicycle theft Criminal damage Death or serious injury - unlawful driving Domestic burglary Homicide Miscellaneous crimes
1 Avon & Somerset   717          3443           15492                                          3              8029       14                 1906
2     Bedfordshire   215          1023            5674                                          8              4512        4                  524
3  Cambridgeshire   355          4295            7818                                         25              4452        7                  998
4        Cheshire   491          1397            9877                                          8              3489        5                 1731
5       Cleveland   368           963            9187                                         11              4329        3                  882
6         Cumbria   193           269            4948                                         16              1148        3                  604
```

5. Set values in **Force.Name** column as rownames

```
rownames(crimes_pivot) <- crimes_pivot[,1]
```

```
crimes_pivot[,1] <- NULL
head(crimes_pivot)
```

```
> head(crimes_pivot)
                 Arson Bicycle theft Criminal damage Death or serious injury - unlawful driving Domestic burglary Homicide
Avon & Somerset   717          3443           15492                                          3              8029       14
Bedfordshire      215          1023            5674                                          8              4512        4
Cambridgeshire    355          4295            7818                                         25              4452        7
Cheshire          491          1397            9877                                          8              3489        5
Cleveland         368           963            9187                                         11              4329        3
Cumbria           193           269            4948                                         16              1148        3
```

6. Normalising the dataset

```
normalise <- function(df)
{
return(((df- min(df)) /(max(df)-min(df))*(1-0))+0)
}
#Normalise the crimes_pivot data set using above normalise function

Force.Name<-rownames(crimes_pivot)
crimes_pivot_n<-as.data.frame(lapply(crimes_pivot,normalise))
rownames(crimes_pivot_n)<-Force.Name

head(crimes_pivot_n)
```

```
> head(crimes_pivot_n)
                     Arson Bicycle.theft Criminal.damage Death.or.serious.injury...unlawful.driving Domestic.burglary    Homicide Miscellaneous.crimes
Avon & Somerset 0.27318008   0.162085731      0.26572171                                 0.03846154        0.14046618 0.077419355           0.18325955
Bedfordshire    0.08084291   0.042022227      0.09454818                                 0.13461538        0.07882930 0.012903226           0.04097601
Cambridgeshire  0.13448276   0.204356023      0.13192810                                 0.46153846        0.07777778 0.032258065           0.08977659
Cheshire        0.18659004   0.060577496      0.16782607                                 0.13461538        0.06090081 0.019354839           0.16524246
Cleveland       0.13946360   0.039045446      0.15579615                                 0.19230769        0.07562215 0.006451613           0.07783383
Cumbria         0.07241379   0.004614011      0.08189061                                 0.28846154        0.01987382 0.006451613           0.04921240
```

7. Assessing clustering tendency

get_clust_tendency() function in **factoextra** package helps to calculate **Clustering tendency.** It's determines whether a given dataset contains meaningful clusters (i.e., non-random structure). **Hopkins statistic** is used to assess the **clustering tendency** of a dataset by measuring the probability that a given dataset is generated by a uniform data distribution. In other words, it tests the **spatial randomness** of the data.

If the value of **Hopkins statistic** is close to zero, then we can reject the null hypothesis and conclude that the dataset is significantly a clusterable data.

The null and the alternative hypotheses are defined as follow:

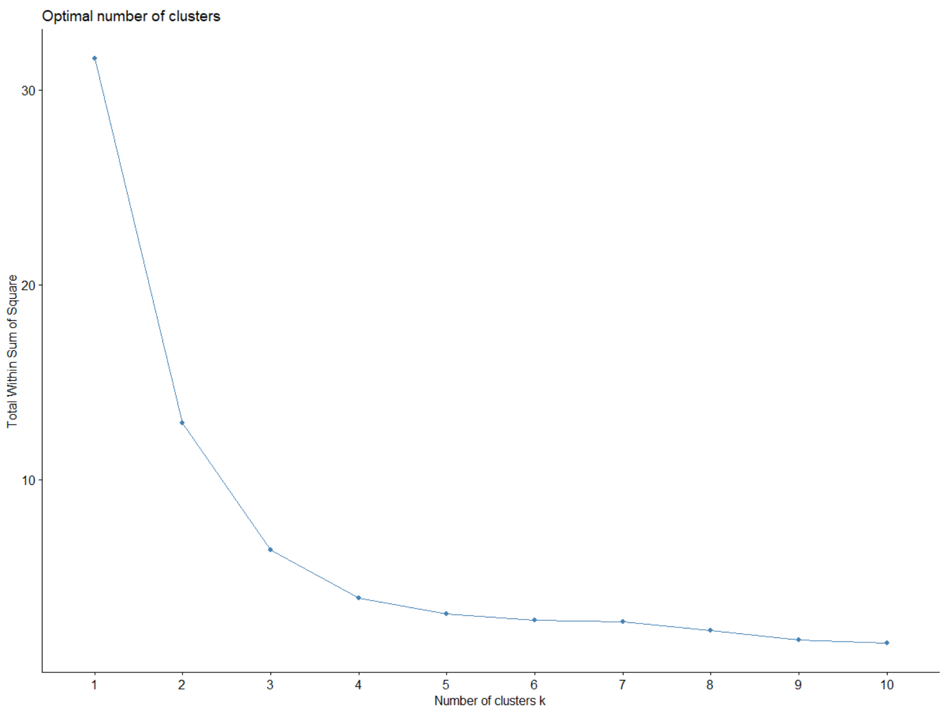
- **Null hypothesis**: the dataset is uniformly distributed (i.e., no meaningful clusters)
- **Alternative hypothesis**: the dataset is not uniformly distributed (i.e., contains meaningful clusters)

```
tendency <- get_clust_tendency(crimes_pivot_n, 30, graph = TRUE)
tendency$hopkins_stat
```

```
> tendency$hopkins_stat
[1] 0.1746332
```

8. If the data is clusterable, then how to choose the right number of expected clusters (k)?

```
#fviz_nbclust()function dertemines and visualize the optimal number of clusters using different methods: within cluster sums of squares, average silhouette and gap statistics.

# fviz_nbclust()function plot the Within Cluster Sum of Squares and the number of clusters to find the location of a bend or a knee in the plot which is considered as an indicator of the appropriate number of clusters.

fviz_nbclust(crimes_pivot_n, kmeans, method = "wss")
```



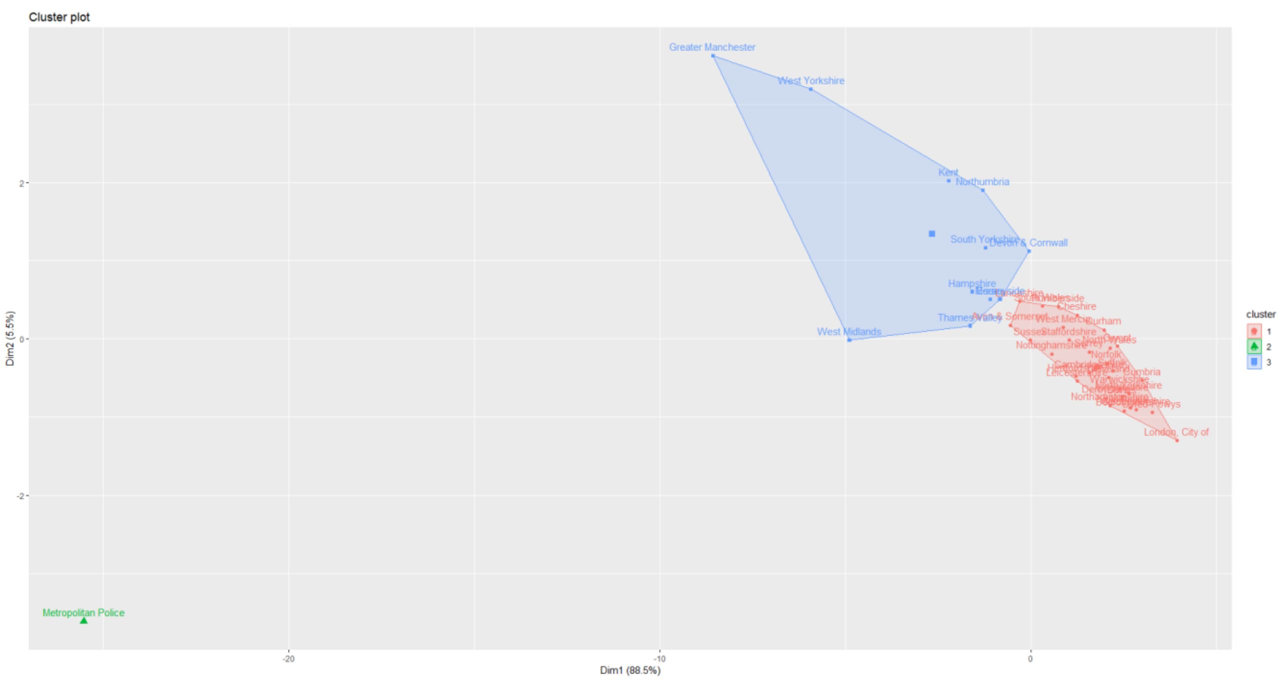
9. Perform k-means clustering on a **crimes_pivot_n** dataset with k=3

```
set.seed(123)
km.fit <- kmeans(crimes_pivot_n, 3, nstart = 30)
km.fit$cluster
km.fit$size
```

```
> km.fit$size
[1] 31  1 11
```

10. Visualize clusters using **fviz_cluster()** function in **factoextra** package

```
fviz_cluster(km.fit,crimes_pivot_n)
```



11. Since “Metropolitan Police” has a large number of crimes and behave like an outlier. So, perform k-means clustering on a **crimes_pivot_n** dataset with k=3, but without “Metropolitan Police” crime figures.

```
crimes_pivot_n2< - subset(crimes_pivot_n,
                         rownames(crimes_pivot_n)!="Metropolitan Police")
```

```
# Compute k-means
set.seed(123)
km.fit2 <- kmeans(crimes_pivot_n2, 3, nstart = 30)
km.fit2$cluster
km.fit2$size
```

```
> km.fit2$size
[1] 26  2 14
```

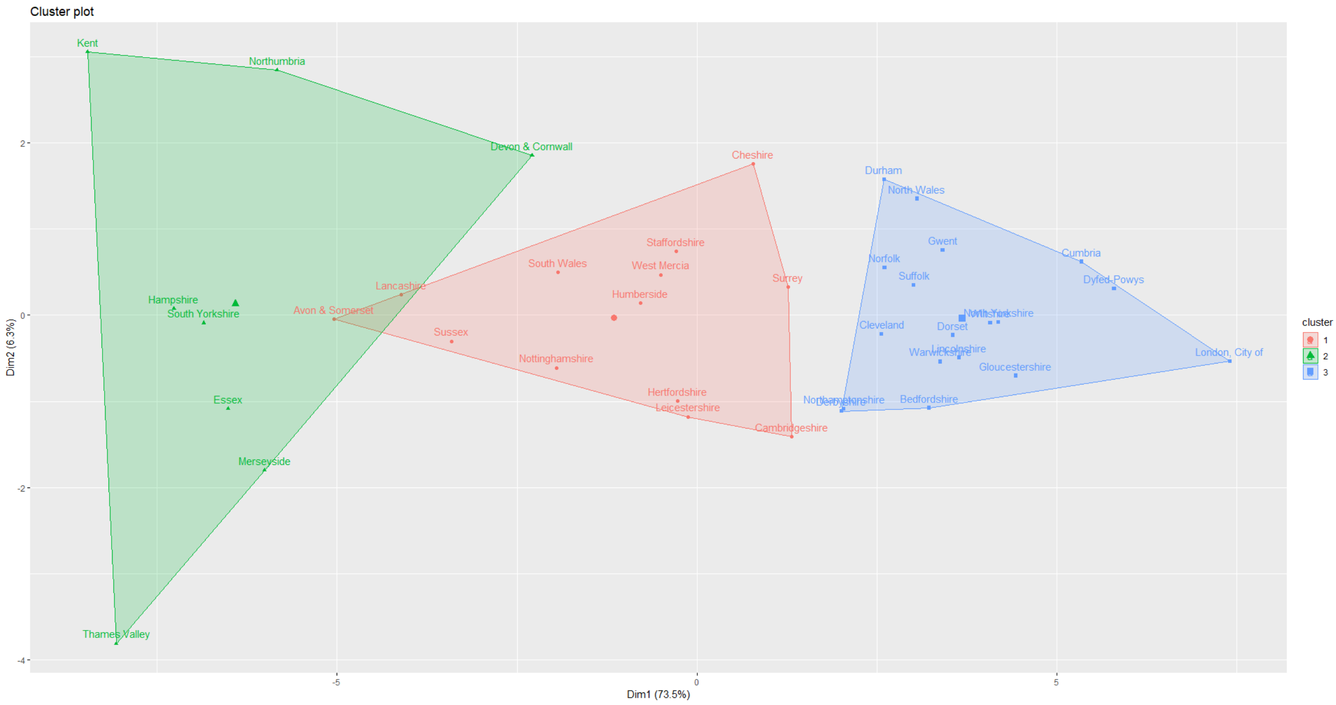
```
# Visualise the clusters

fviz_cluster(km.fit2,crimes_pivot_n2)
```

12. Perform k-means clustering on a **crimes_pivot_n** dataset with k=3, but without below mentioned areas.
    - Metropolitan Police
    - Greater Manchester
    - West Midlands
    - West Yorkshire

```
crimes_pivot_n3<-subset(crimes_pivot_n, !(rownames(crimes_pivot_n) %in%
                                c("Metropolitan Police","Greater Manchester",
                                "West Midlands", "West Yorkshire")))
```

```
# Compute k-means
set.seed(123)
km.fit3 <- kmeans(crimes_pivot_n3, 3, nstart = 30)
km.fit3$cluster
km.fit3$size
```

```
fviz_cluster(km.fit3,crimes_pivot_n3)
```




```
fviz_cluster(km.fit3,crimes_pivot_n3,ellipse.type = "norm")
```