QuantUniversity, LLC
www.quantuniversity.com

Data science (Unsupervised learning methods)

Presented By:
Sri Krishnamurthy, CFA, CAP
www.QuantUniversity.com
sri@quantuniversity.com

Unsupervised learning Introduction

Unsupervised learning methods

- Are known as methods deal with finding patterns or classes of unlabeled data objects. In other word in such datasets there is no continuous or discrete responds associated with observations.
- Methods include of clustering (partitioning and hierarchical), P.C.A, association rules mining and Kernel density clustering.
- The AdultUCI, iris and inquisition.basket are the sample datasets used in this chapter.
- In python mainly scikit-learn package deals with data mining and machine learning algorithms while there are plenty of packages and functions in R.

Clustering

- ✓ K-Means clustering
- ✓ Linkage methods
- ✓ Distance functions
- ✓ Hierarchical clustering
- ✓ Elbow chart and bend graph
- ✓ Kernel density estimation
- ✓ Kernel density clustering

K-Means clustering

- K-Means is defined as the main partitioning clustering method which divides data into different clusters.
- K-Means randomly assigns initial clusters to observations and tries to modify that at each iteration such a way within clusters distance of objects reaches minimum.
- Final answers in K-Mean depends on initial assignments and are defined as local minima. To reach global minima this process should be repeated over and over for reasonable number of times.
- K-Means calculates the distance of each data object from the center of the cluster while function form can be Euclidean, Manhattan or correlation, etc.

Linkage methods

- In clustering methods to answer how close the data objects and clusters are will be answered by different types of linkage as:
- Complete: The largest distance of inter-clusters by comparing pairwise
- Single: The smallest distance of inter-clusters by comparing pairwise
- Average: The average distance of all inter-clusters pairwise data objects
- Centroid: The distance between centroids of the clusters

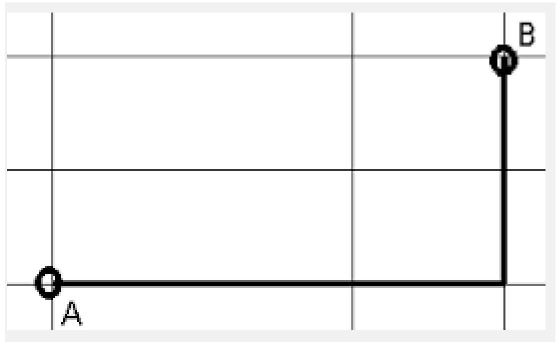
Distance functions

- Three main commonly used distance functions in clustering methods are Euclidean, Manhattan and correlation (mostly used in time series analysis) while there are couple of other distance types such as Minkowski, Canberra, etc.
- Euclidean distance: In a simple 2 dimensional space for points A,B is defined as below and it can be easily extended to n dimensional space:

$$D = \sqrt{(X_A - X_B)^2 + (Y_A - Y_B)^2}$$

Distance functions

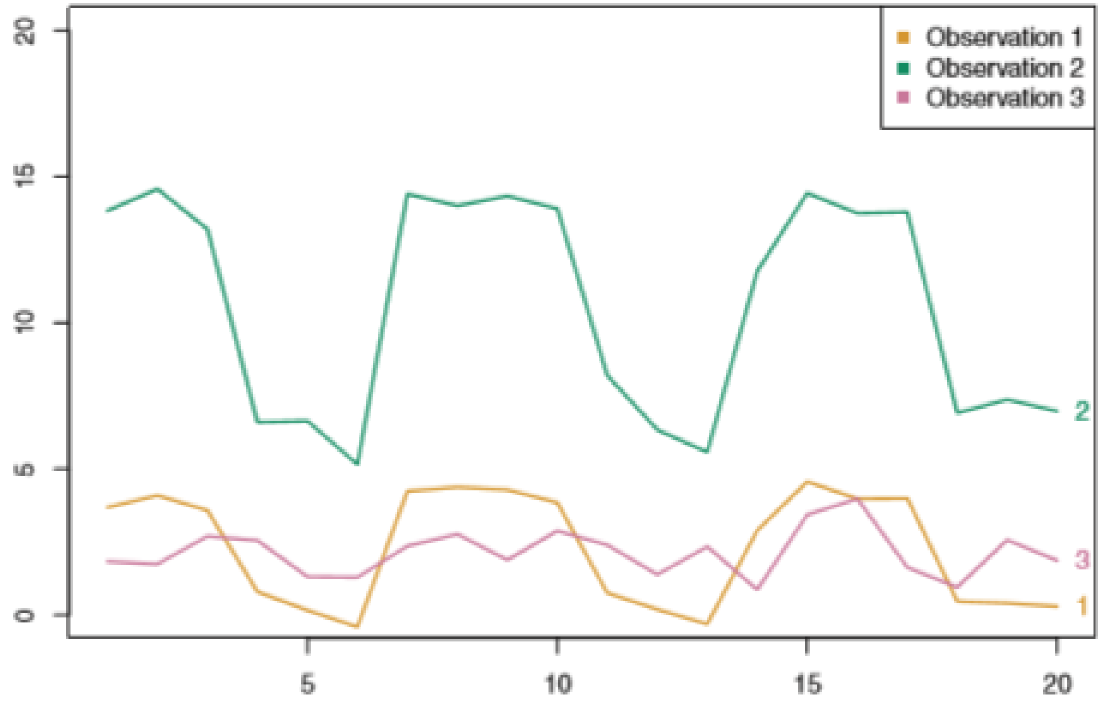
- Manhattan distance: In a simple 2 dimensional space for points A,B is defined as below and it can be easily extended to n dimensional space:



$$D = |X_A - X_B| + |Y_A - Y_B|$$

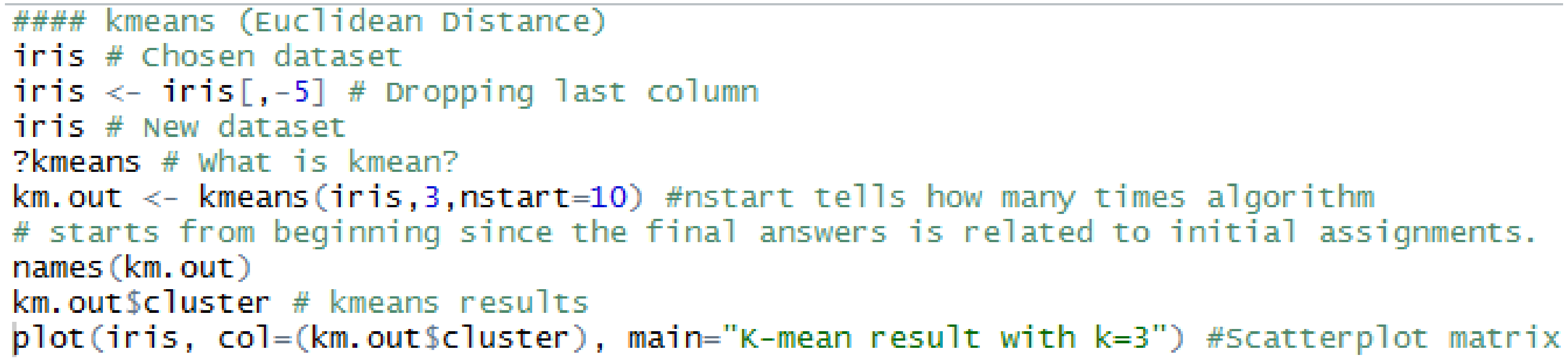
Distance functions

- Correlation distance: Considers two observations are similar if their features are highly correlated such as observations 1 and 2 in below graph:

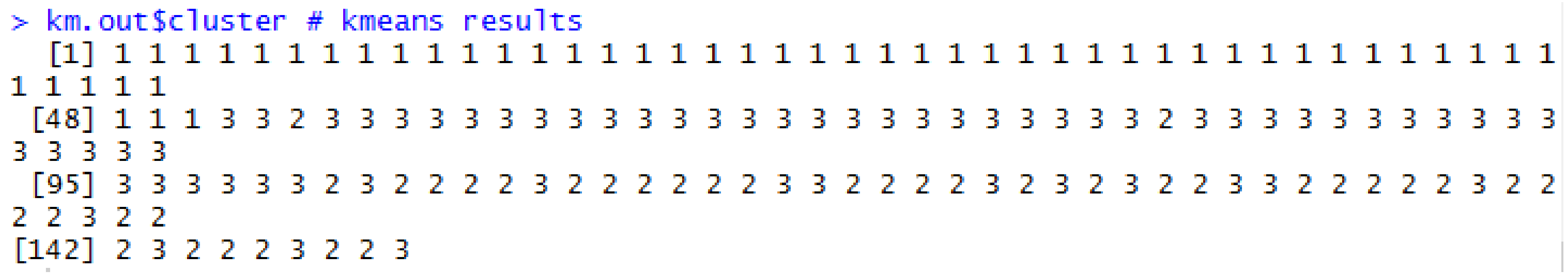


QuantUniversity, LLC
www.quantuniversity.com

K-Means clustering (R)



```
#### kmeans (Euclidean Distance)
iris # Chosen dataset
iris <- iris[,-5] # Dropping last column
iris # New dataset
?kmeans # What is kmean?
km.out <- kmeans(iris,3,nstart=10) #nstart tells how many times algorithm
# starts from beginning since the final answers is related to initial assignments.
names(km.out)
km.out$cluster # kmeans results
plot(iris, col=(km.out$cluster), main="K-mean result with k=3") #Scatterplot matrix
```

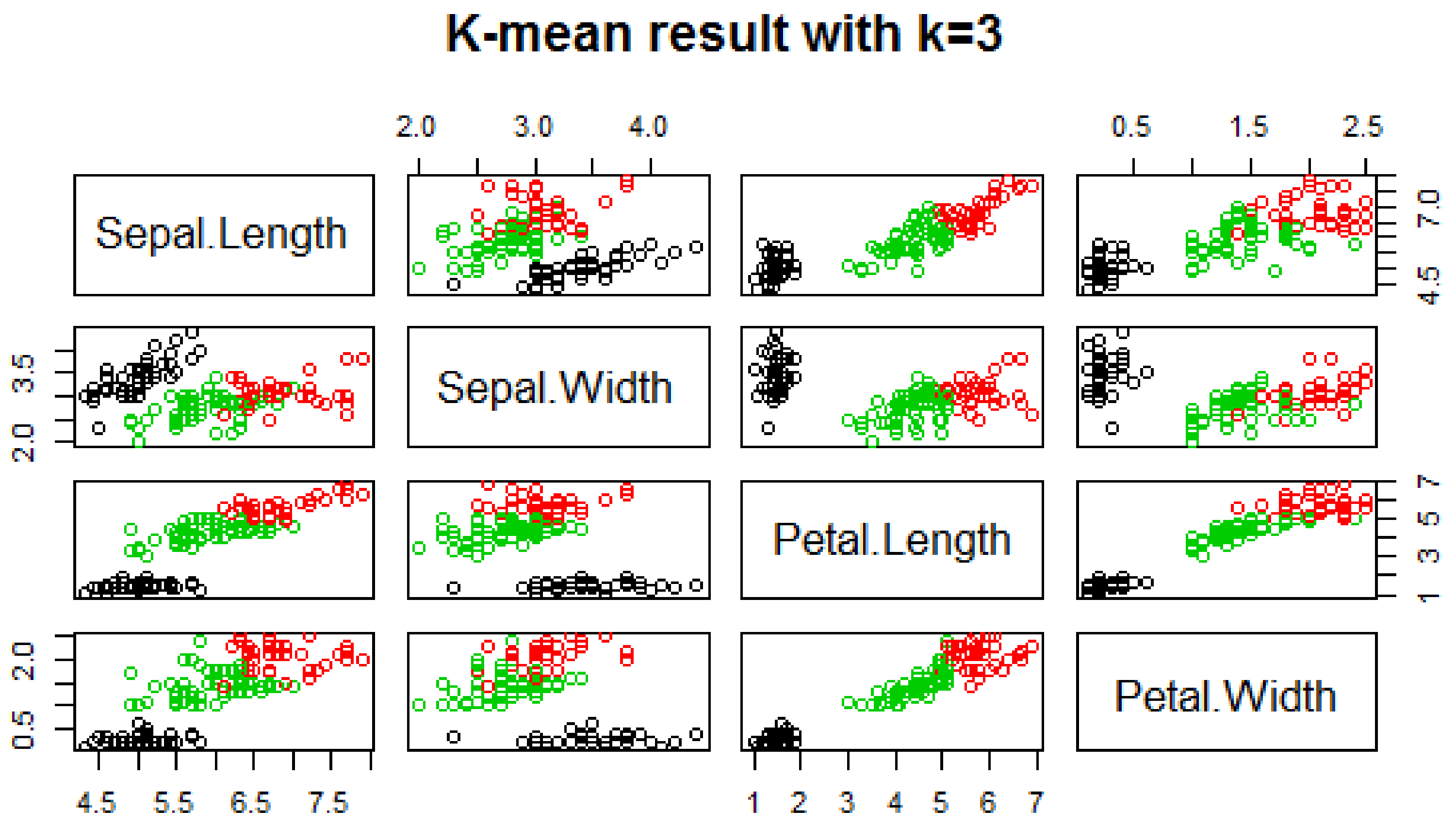


```
> km.out$cluster # kmeans results
   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1
  [48] 1 1 1 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3
  [95] 3 3 3 3 3 3 2 3 2 2 2 2 3 2 2 2 2 2 2 3 3 2 2 2 2 3 2 3 2 3 2 2 3 3 2 2 2 2 2 3 2 2
2 2 3 2 2
[142] 2 3 2 2 2 3 2 2 3
```

See ***Clustering_iris.R***

QuantUniversity, LLC
www.quantuniversity.com

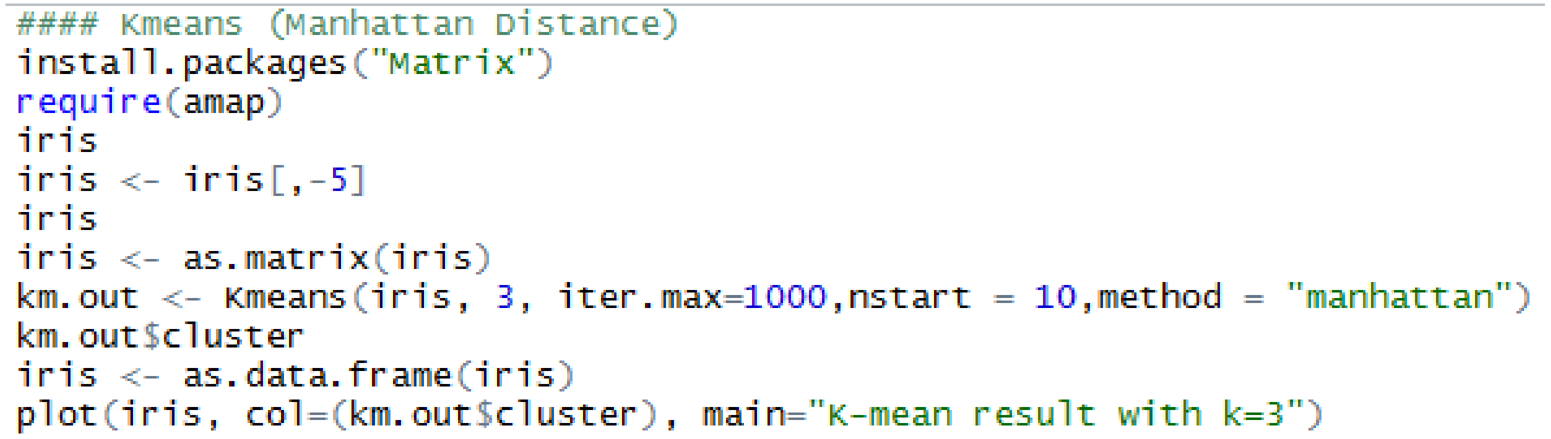
K-Means clustering (R)



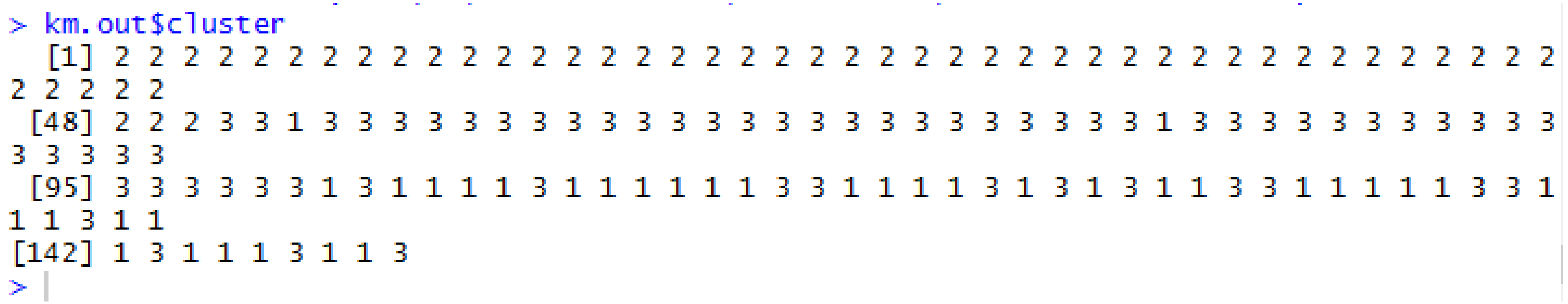
See ***Clustering_iris.R***

QuantUniversity, LLC
www.quantuniversity.com

K-Means clustering (R)



```
#### Kmeans (Manhattan Distance)
install.packages("Matrix")
require(amap)
iris
iris <- iris[,-5]
iris
iris <- as.matrix(iris)
km.out <- Kmeans(iris, 3, iter.max=1000,nstart = 10,method = "manhattan")
km.out$cluster
iris <- as.data.frame(iris)
plot(iris, col=(km.out$cluster), main="K-mean result with k=3")
```

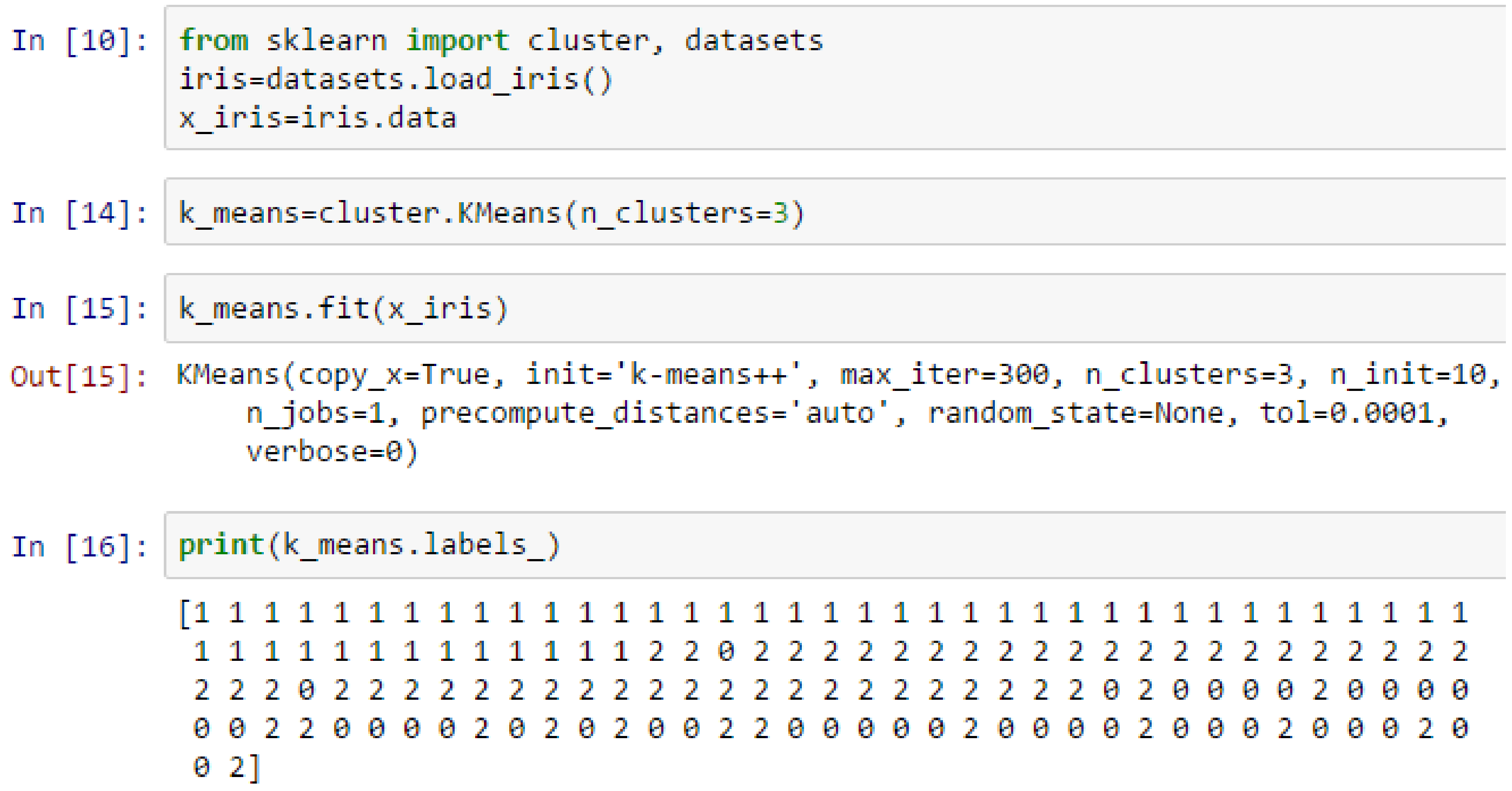


```
> km.out$cluster
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2
 [48] 2 2 2 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3
 [95] 3 3 3 3 3 3 1 3 1 1 1 1 3 1 1 1 1 1 1 3 3 1 1 1 1 3 1 3 1 3 1 1 3 3 1 1 1 1 1 3 3 1
1 1 3 1 1
[142] 1 3 1 1 1 3 1 1 3
> 
```

See ***Kmeans_DifferentDistances_iris.R***

QuantUniversity, LLC
www.quantuniversity.com

K-Means clustering (Python)



```
In [10]: from sklearn import cluster, datasets
         iris=datasets.load_iris()
         x_iris=iris.data

In [14]: k_means=cluster.KMeans(n_clusters=3)

In [15]: k_means.fit(x_iris)

Out[15]: KMeans(copy_x=True, init='k-means++', max_iter=300, n_clusters=3, n_init=10,
             n_jobs=1, precompute_distances='auto', random_state=None, tol=0.0001,
             verbose=0)

In [16]: print(k_means.labels_)

         [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
          1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
          2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 0 0 0 0 2 0 0 0 0
          0 0 2 2 0 0 0 0 2 0 2 0 2 0 0 2 2 0 0 0 0 0 2 0 0 0 0 2 0 0 0 2 0 0 0 2 0
          0 2]
```

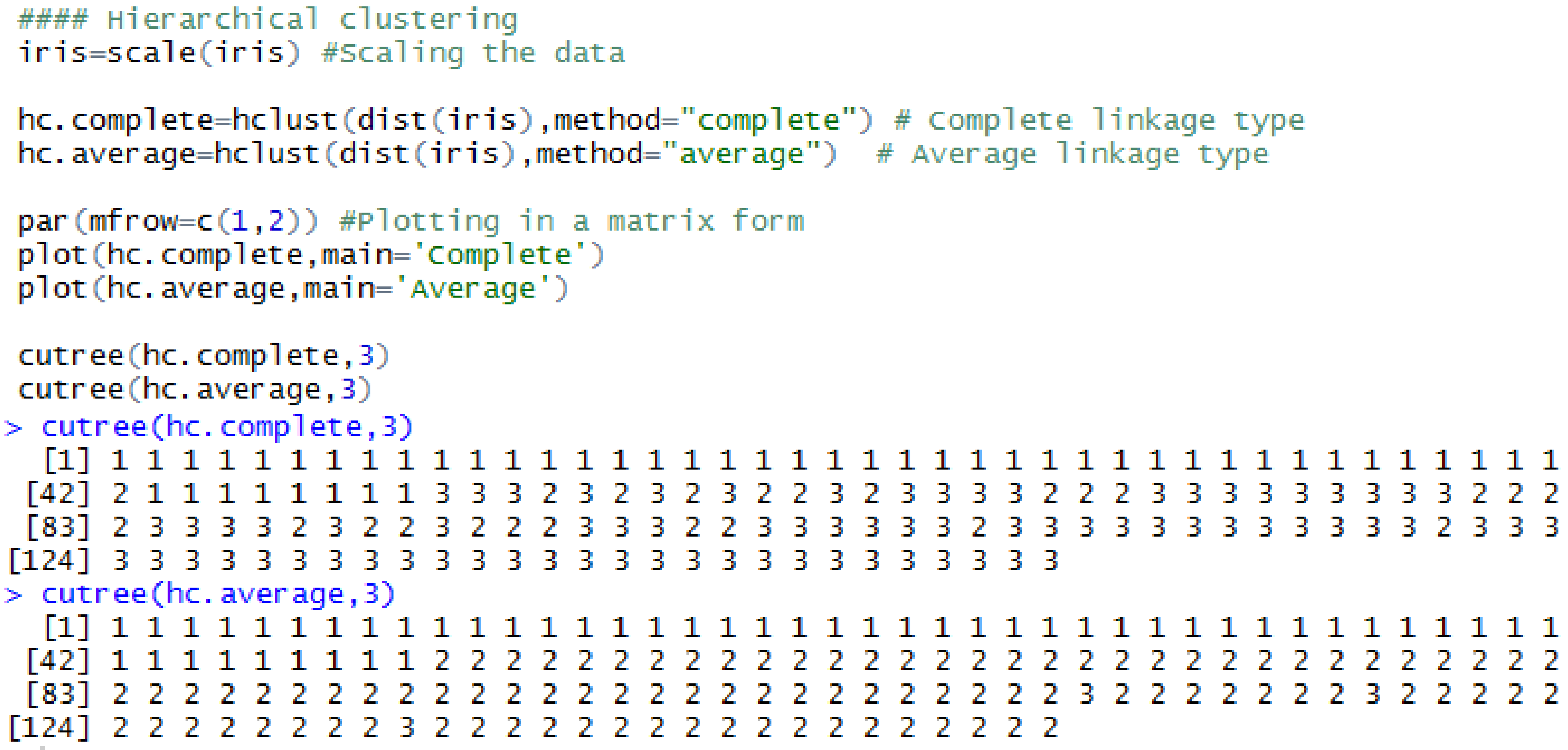
See ***Kmeans_Clustering_iris.ipynb***

QuantUniversity, LLC
www.quantuniversity.com

Hierarchical clustering

- Despite K-Means in which initial number of clusters are assigned, in hierarchical we don`t need to know about the number of clusters.
- Hierarchical clustering applies bottom-up or agglomerative methods to create the tree-based visualization of clusters which is called Dendrogram.
- By cutting Dendrogram at any specific point we will have different number of clusters .
- Linkage methods which can be used are complete, single, average and centroid while the distance function can be Euclidean, correlation, etc.

Hierarchical clustering (R)



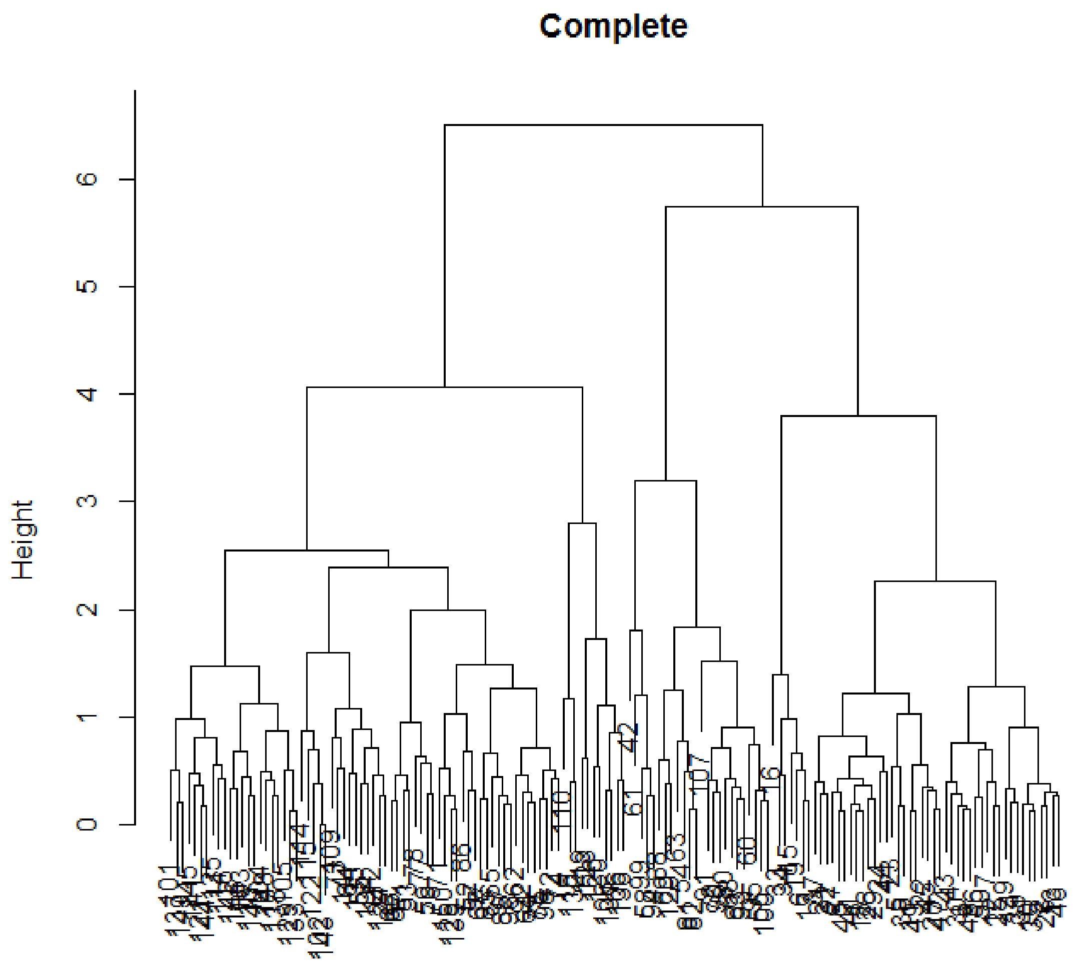
```
#### Hierarchical clustering
iris=scale(iris) #Scaling the data

hc.complete=hclust(dist(iris),method="complete") # Complete linkage type
hc.average=hclust(dist(iris),method="average")  # Average linkage type

par(mfrow=c(1,2)) #Plotting in a matrix form
plot(hc.complete,main='Complete')
plot(hc.average,main='Average')

cutree(hc.complete,3)
cutree(hc.average,3)
```

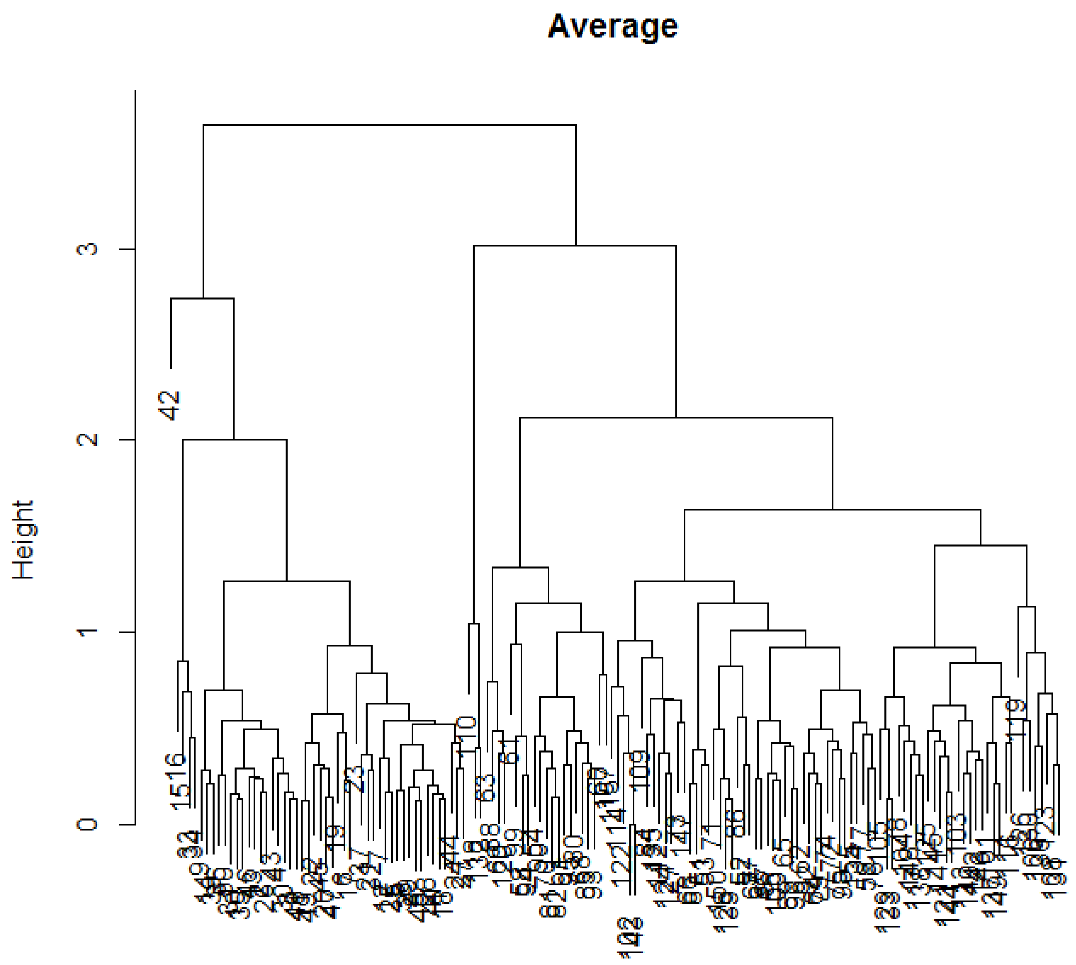
```
> cutree(hc.complete,3)
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [42] 2 1 1 1 1 1 1 1 1 3 3 3 2 3 2 3 2 3 2 2 3 2 3 3 3 3 2 2 2 3 3 3 3 3 3 3 3 3 2 2 2
 [83] 2 3 3 3 3 2 3 2 2 3 2 2 2 3 3 3 2 2 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3
[124] 3 3 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
> cutree(hc.average,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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [42] 1 1 1 1 1 1 1 1 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
 [83] 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 3 2 2 2 2 2 2 2 3 2 2 2 2 2
[124] 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

See ***Clustering_iris.R***

QuantUniversity, LLC
www.quantuniversity.com

Hierarchical clustering (R)

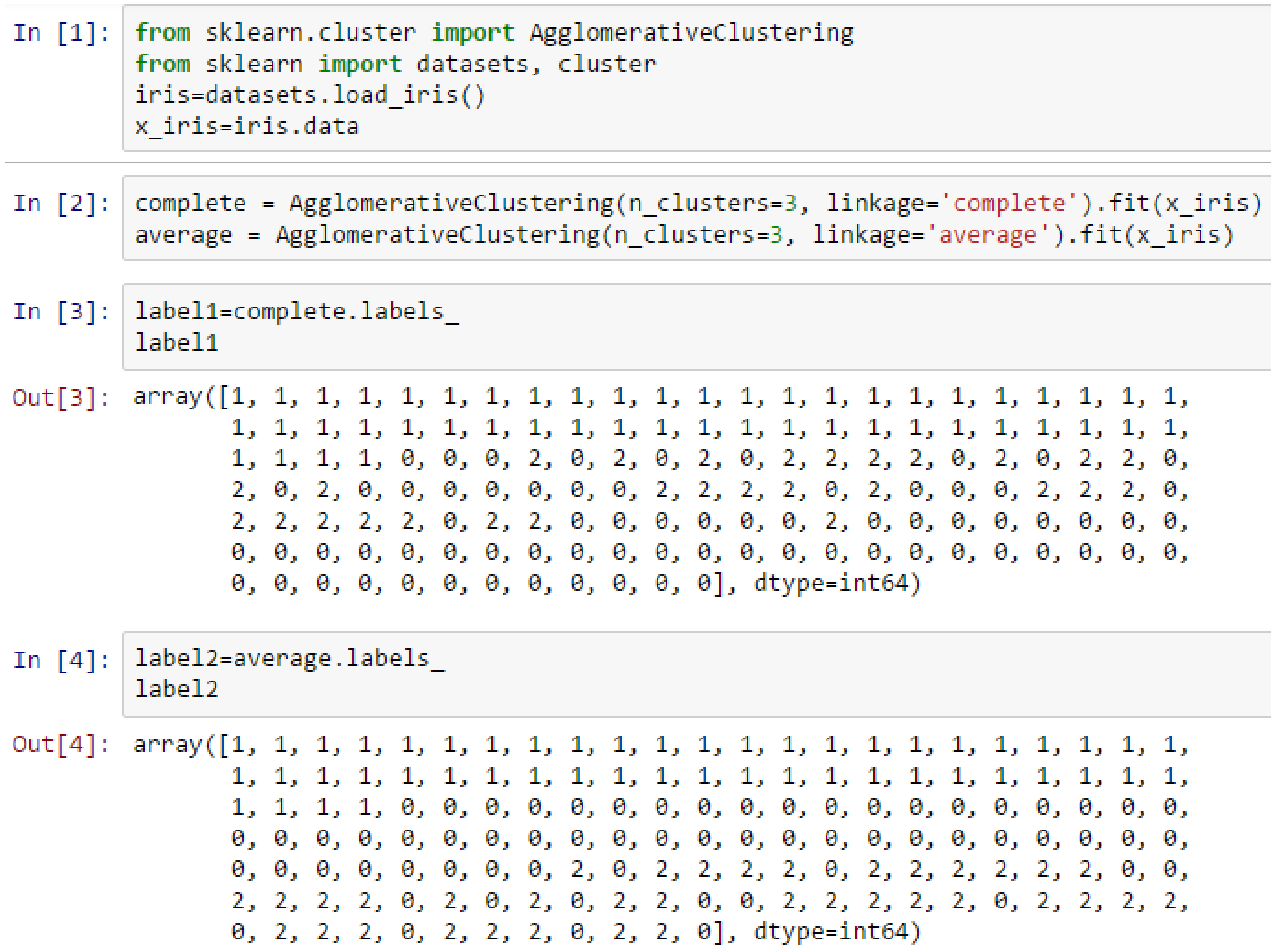




See ***Clustering_iris.R***

QuantUniversity, LLC
www.quantuniversity.com

Hierarchical clustering (Python)



```
In [1]: from sklearn.cluster import AgglomerativeClustering
        from sklearn import datasets, cluster
        iris=datasets.load_iris()
        x_iris=iris.data

In [2]: complete = AgglomerativeClustering(n_clusters=3, linkage='complete').fit(x_iris)
        average = AgglomerativeClustering(n_clusters=3, linkage='average').fit(x_iris)

In [3]: label1=complete.labels_
        label1

Out[3]: array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1, 1, 1, 0, 0, 0, 2, 0, 2, 0, 2, 0, 2, 2, 2, 2, 0, 2, 0, 2, 2, 0,
               2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 0, 2, 0, 0, 0, 2, 2, 2, 0,
               2, 2, 2, 2, 2, 0, 2, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)

In [4]: label2=average.labels_
        label2

Out[4]: array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 0, 0,
               2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2,
               0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0], dtype=int64)
```

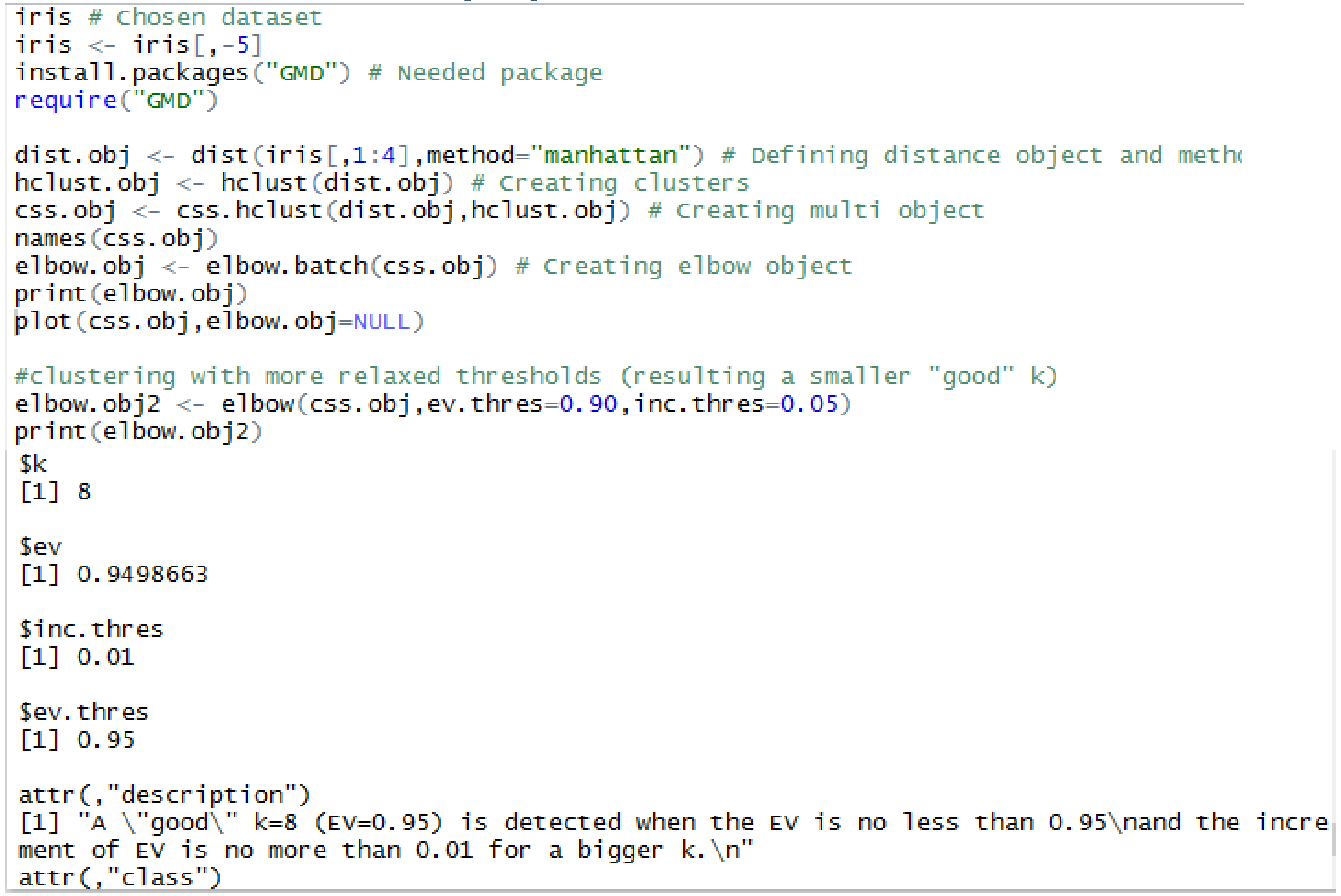
See ***Hierarchical_Clustering_iris.ipynb***

QuantUniversity, LLC
www.quantuniversity.com

Elbow chart and bend graph

- As mentioned in hierarchical clustering, by cutting the Dendrogram at any specific point we may have different number of clusters, but what is the optimum number of clusters?
- Elbow function answers to this question by considering a threshold for explained variance of the dataset.
- Although in K-Means we may initially assign different number of clusters, to answer the same question, we may plot within sum of squared error for different number of clusters and choose the best number of clusters.

Elbow chart (R)



```
iris # Chosen dataset
iris <- iris[,-5]
install.packages("GMD") # Needed package
require("GMD")

dist.obj <- dist(iris[,1:4],method="manhattan") # Defining distance object and meth
hclust.obj <- hclust(dist.obj) # Creating clusters
css.obj <- css.hclust(dist.obj,hclust.obj) # Creating multi object
names(css.obj)
elbow.obj <- elbow.batch(css.obj) # Creating elbow object
print(elbow.obj)
plot(css.obj,elbow.obj=NULL)

#clustering with more relaxed thresholds (resulting a smaller "good" k)
elbow.obj2 <- elbow(css.obj,ev.thres=0.90,inc.thres=0.05)
print(elbow.obj2)
```

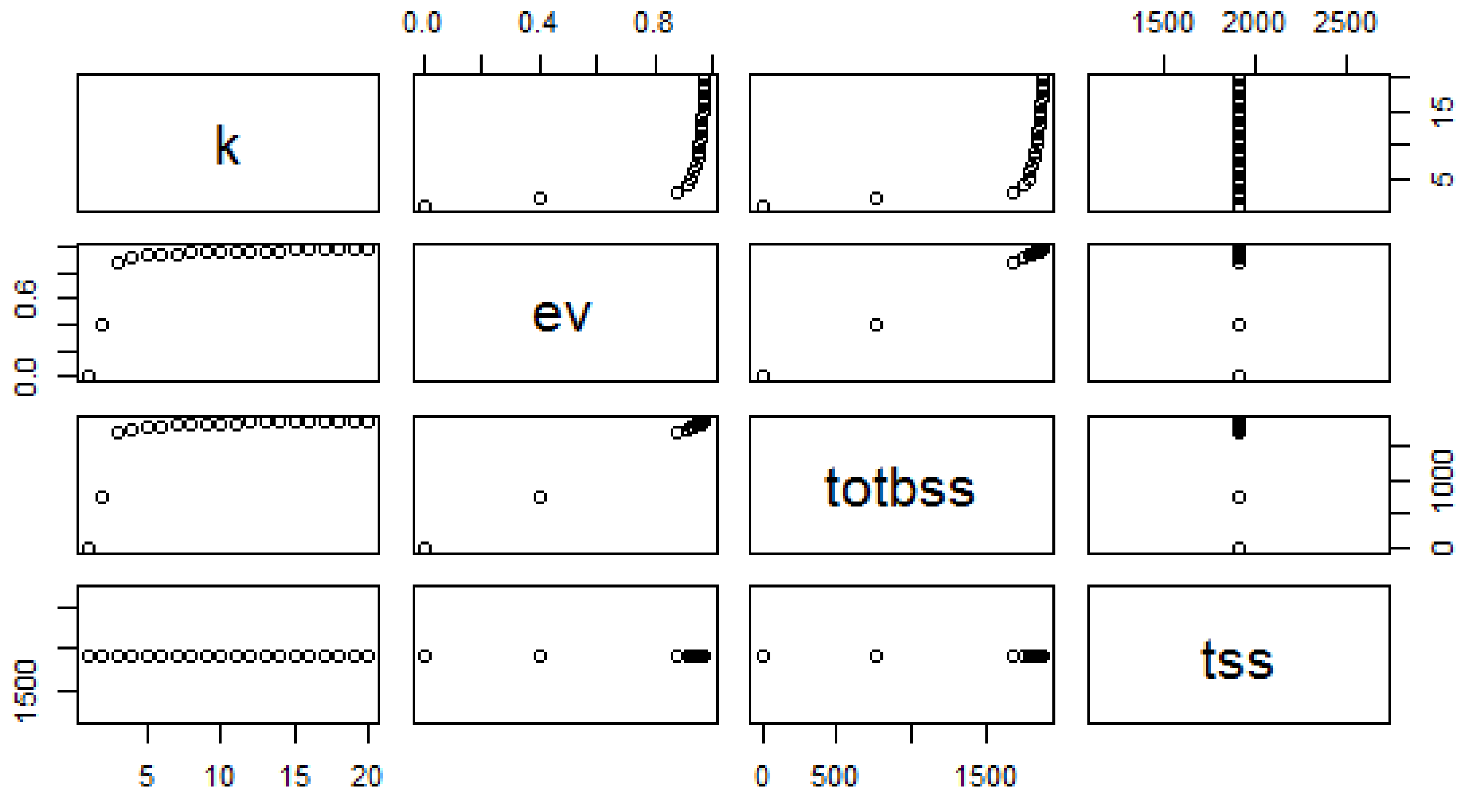
```
$k
[1] 8

$ev
[1] 0.9498663

$inc.thres
[1] 0.01

$ev.thres
[1] 0.95

attr(,"description")
[1] "A \"good\" k=8 (EV=0.95) is detected when the EV is no less than 0.95\nand the incre
ment of EV is no more than 0.01 for a bigger k.\n"
attr(,"class")
```

css.obj is a multi-object list computing Sum-of-Square for clustering evaluation

See ***Elbow&Bend_Charts_iris.R***

QuantUniversity, LLC
www.quantuniversity.com

Elbow chart (R)



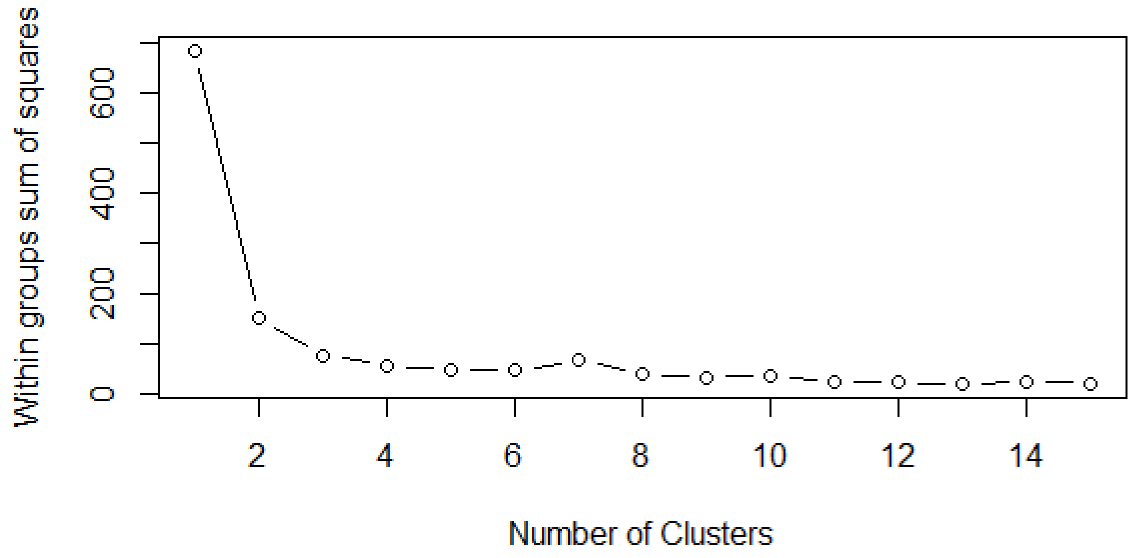
K: number of clusters
totbss: total between cluster sum-of-square
tss: total sum of square of the data
ev: explained variance given k

See ***Elbow&Bend_Charts_iris.R***

QuantUniversity, LLC
www.quantuniversity.com

Bend graph (R)

```
##Bend graph
iris # Chosen dataset
iris <- iris[,-5] # Dropping last column
iris
wss <- nrow((iris)-1)*sum(apply(iris,2,var))
for (i in 2:15)
{
  wss[i] <- sum(kmeans(iris,centers=i)$withinss) # Within Sum of Squared for
                                                  # different number of clusters
}

plot(1:15, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares")
```



See ***Elbow&Bend_Charts_iris.R***

QuantUniversity, LLC
www.quantuniversity.com

Bend graph (Python)

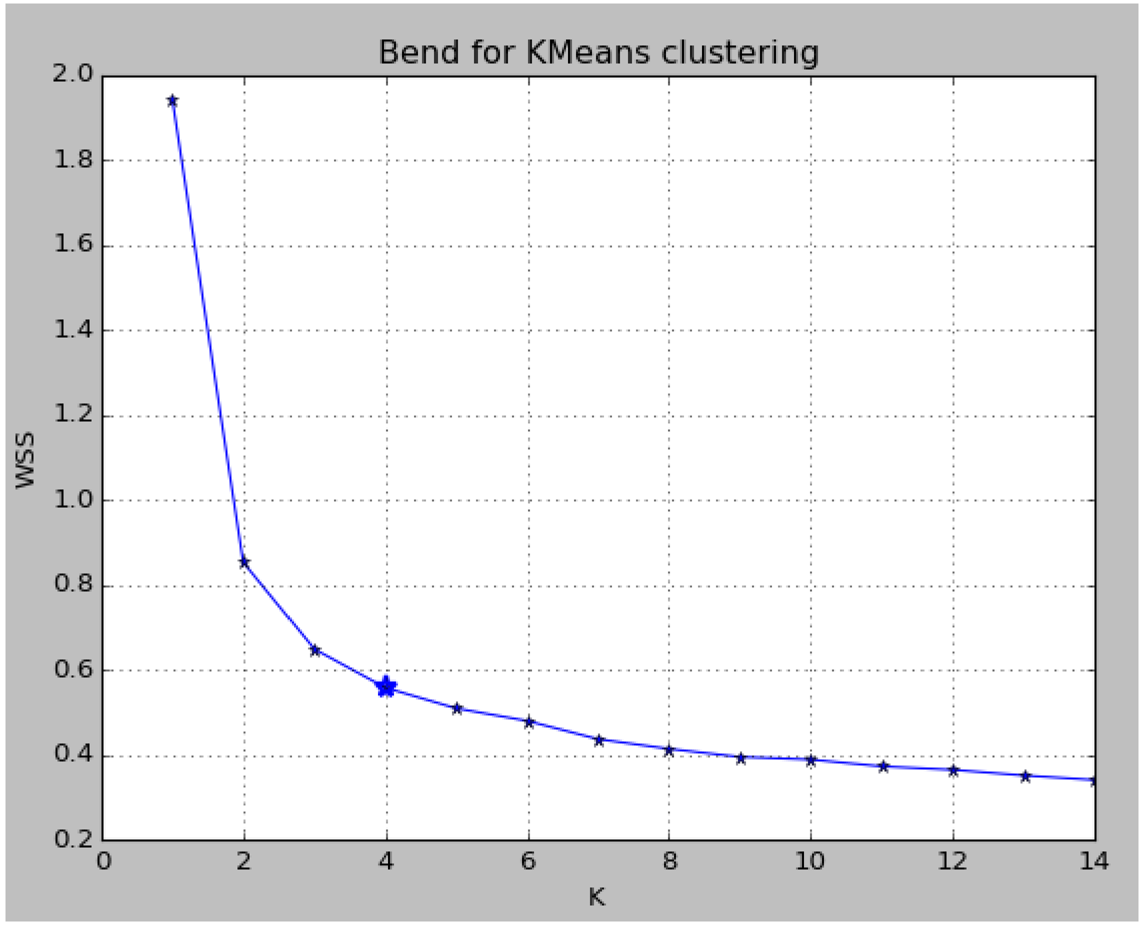
```
### Needed dataset
from sklearn import *
iris=datasets.load_iris()
print iris.data.shape
```

```
(150L, 4L)
```

```
### Needed modules and functions
import numpy as np
import scipy
from scipy.cluster.vq import kmeans,vq
from scipy.spatial.distance import cdist
import matplotlib.pyplot as plt
```

```
### Defining different number of clusters and calculate wss for each of them
K = range(1,15)
KM = [kmeans(iris.data,k) for k in K]
centroids = [cent for (cent,var) in KM]   # cluster centroids
avgWithinSS = [var for (cent,var) in KM] # mean within-cluster sum of squares
```

```
kIdx = 3
### Bend graph
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(K, avgWithinSS, 'b*-')
ax.plot(K[kIdx], avgWithinSS[kIdx], marker='*', markersize=10,
    markeredgewidth=2, markeredgecolor='b', markerfacecolor='None')
plt.grid(True)
plt.xlabel('K')
plt.ylabel('WSS')
plt.title('Bend for KMeans clustering')
plt.show()
```



See *Bend_Charts_iris.ipynb*

QuantUniversity, LLC
www.quantuniversity.com

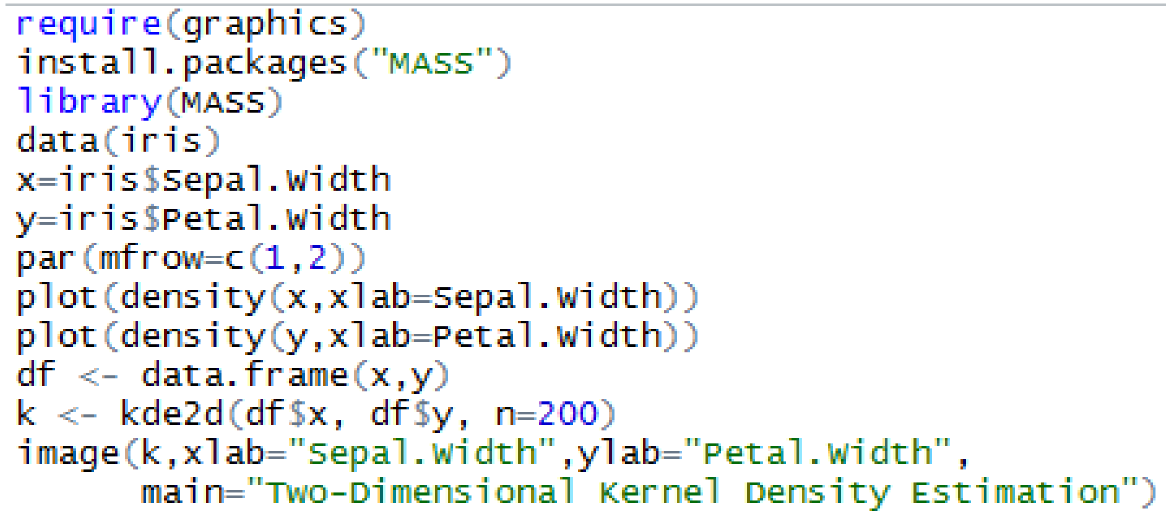
Kernel density estimation

- Density estimation walks the line between unsupervised learning, feature engineering, and data modeling.
- The very basic technique of density estimation is histogram which may lead to different results due to choice of binning problem.
- Gaussian function is the most commonly used one in comparison with the other available kernel forms.
- The main application of kernel density estimation is to know about the distribution of observations by visualizing them.
- Other kernel forms that can be used in kernel density estimation are exponential, linear, etc.

Kernel density clustering

- Kernel density clustering performs cluster analysis based on nonparametric density estimation.
- This method clusters data such a way components with estimated above threshold are maximally connected.
- The number of clusters is automatically selected according to the number of modes of estimated density.
- Kernel density clustering can be applied to either low or high dimensional data.

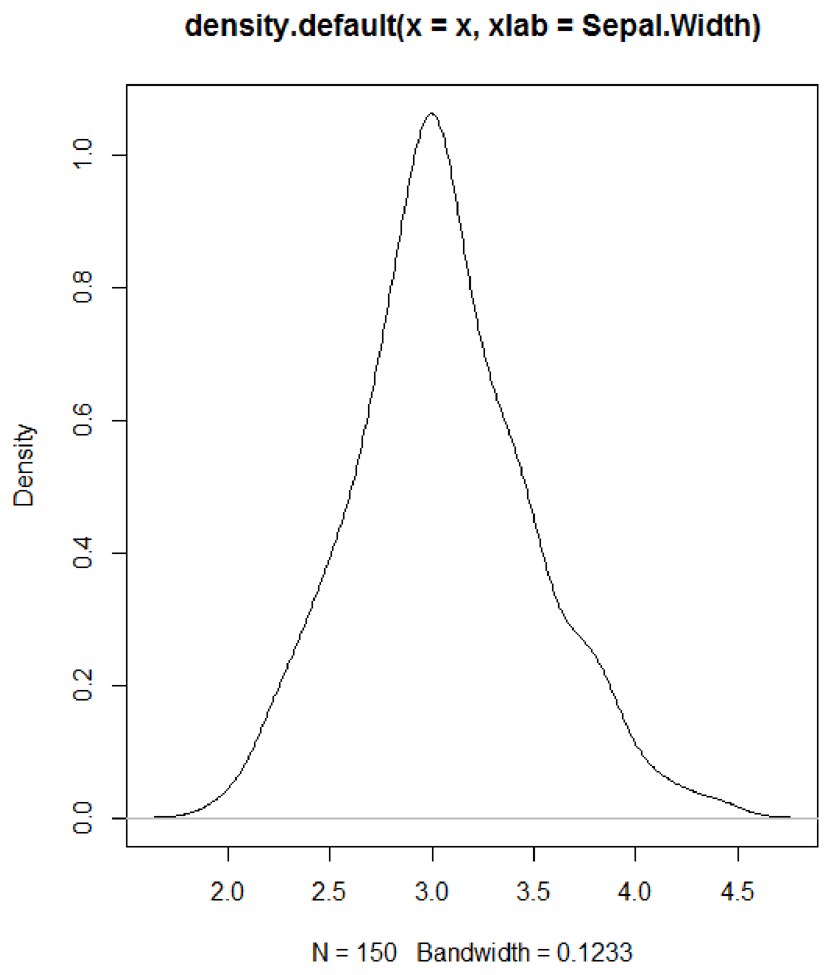
Kernel density estimation (R)

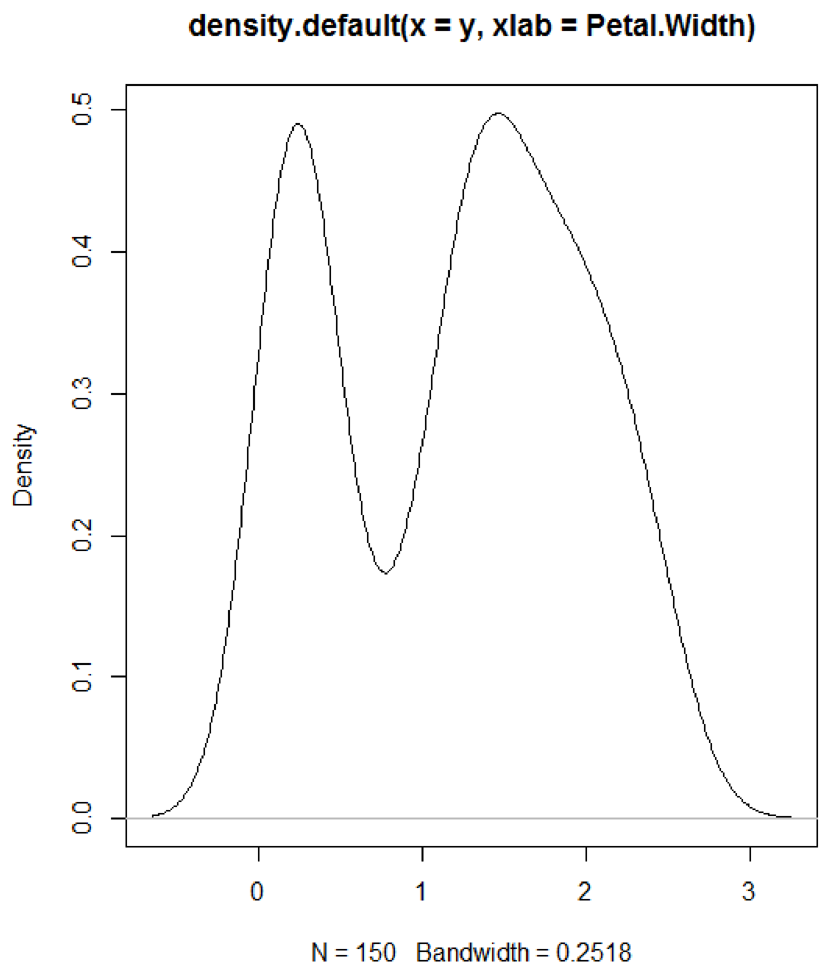


```
require(graphics)
install.packages("MASS")
library(MASS)
data(iris)
x=iris$Sepal.Width
y=iris$Petal.Width
par(mfrow=c(1,2))
plot(density(x,xlab=Sepal.Width))
plot(density(y,xlab=Petal.Width))
df <- data.frame(x,y)
k <- kde2d(df$x, df$y, n=200)
image(k,xlab="Sepal.Width",ylab="Petal.Width",
      main="Two-Dimensional Kernel Density Estimation")
```

See ***Kernel_Density_Estimation_iris.R***

Kernel density estimation (R)

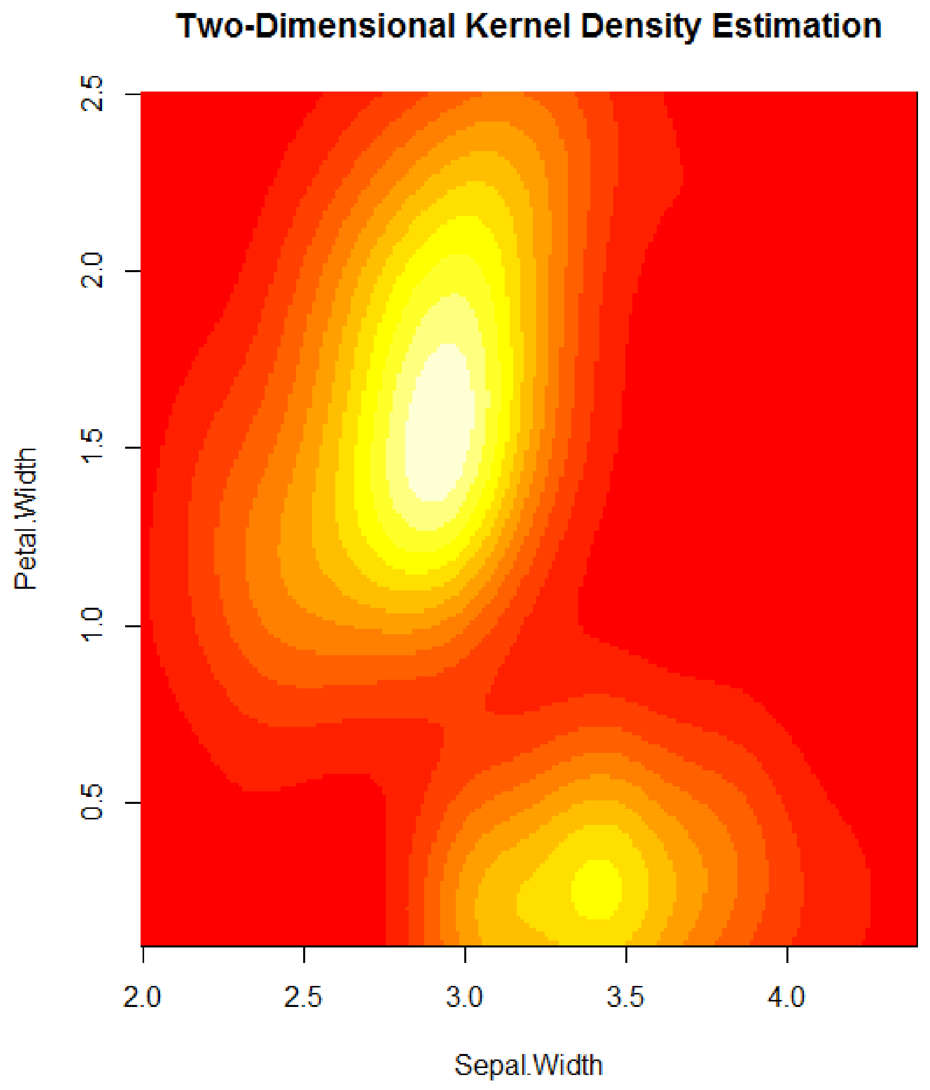




See ***Kernel_Density_Estimation_iris.R***

QuantUniversity, LLC
www.quantuniversity.com

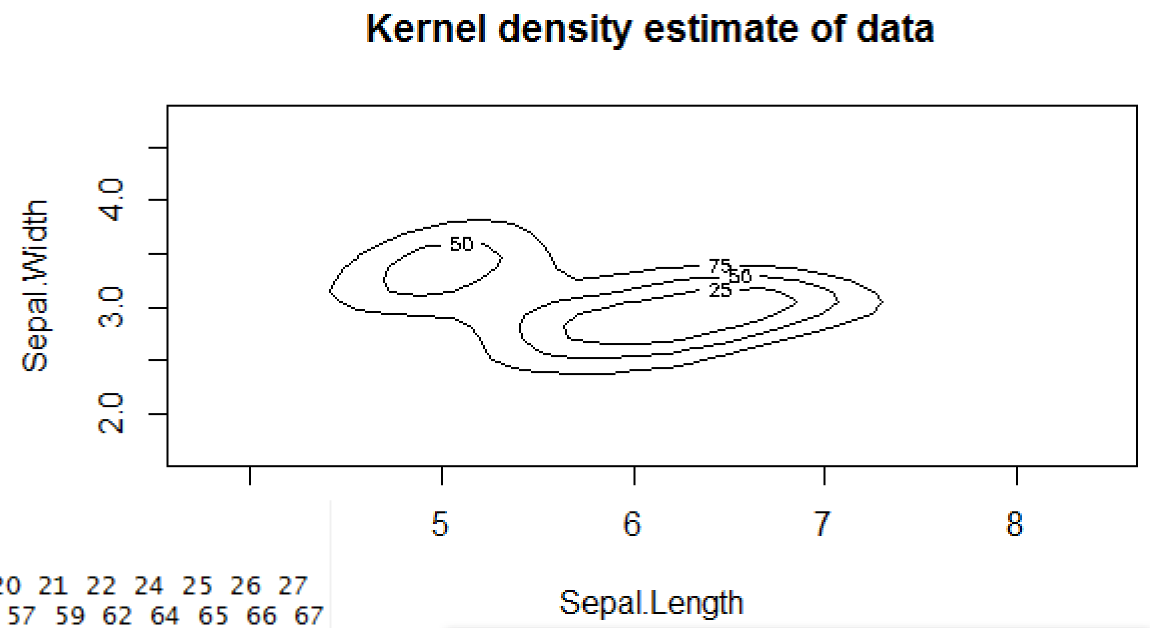
Kernel density estimation (R)



See *Kernel_Density_Estimation_iris.R*

Kernel density clustering (R)

```
### Needed package
install.packages("pdfCluster")
library("pdfCluster")
### Dataset
iris # Chosen dataset
iris <- iris[,-5] # Dropping last column
iris # New dataset
### 2d Clustering
x <- iris[,c(1,2)]
pdf <- kepdf(x)
summary(pdf)
plot(pdf)
```



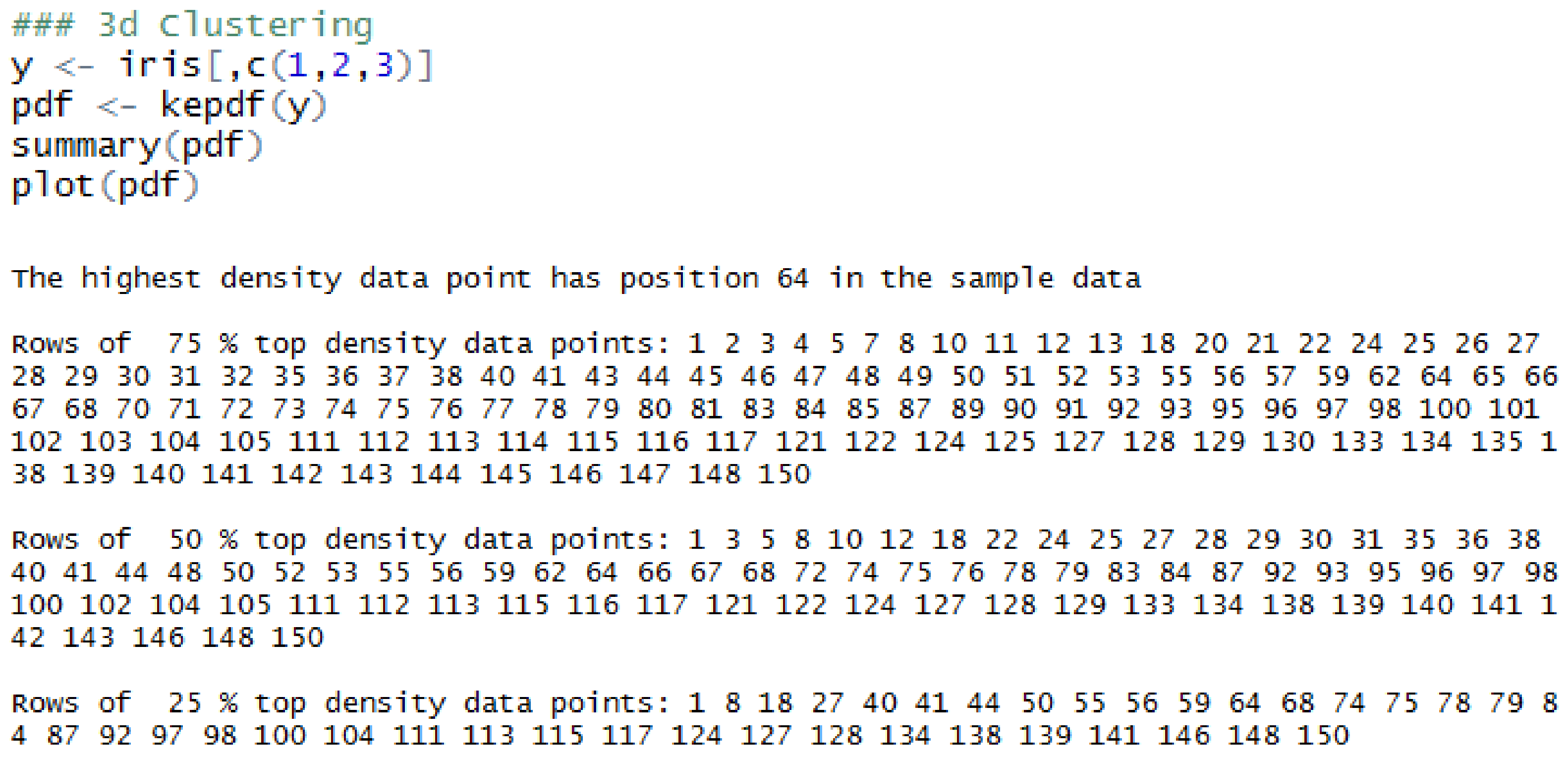
```
The highest density data point has position 98 in the sample data

Rows of  75 % top density data points: 1 2 3 4 5 7 8 10 11 12 13 18 20 21 22 24 25 26 27
28 29 30 31 32 35 36 37 38 40 41 44 45 46 47 48 49 50 51 52 53 55 56 57 59 62 64 65 66 67
68 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 87 89 90 91 92 93 95 96 97 98 100 101
102 103 104 105 111 112 113 114 115 116 117 121 122 124 125 127 128 129 130 133 134 135 1
38 139 140 141 142 143 144 145 146 147 148 150

Rows of  50 % top density data points: 1 3 5 8 12 18 24 25 27 28 29 30 36 40 41 44 50 52
53 55 56 59 62 64 65 66 67 68 72 74 75 76 78 79 80 83 84 87 89 91 92 93 95 96 97 98 100 1
02 104 105 111 112 113 115 116 117 121 122 124 127 128 129 133 134 135 138 139 140 141 14
2 143 144 146 148 150

Rows of  25 % top density data points: 55 56 59 62 64 66 68 72 74 75 76 78 79 83 84 87 92
97 98 100 102 104 105 113 115 117 127 128 129 133 134 138 139 141 143 146 148 150
```

Kepdf (kernel estimate of a probability density function) estimates density of uni-variate or multivariate data by the kernel method.

See ***Kernel_Density_Clustering_iris.R***

QuantUniversity, LLC
www.quantuniversity.com

Kernel density clustering (R)



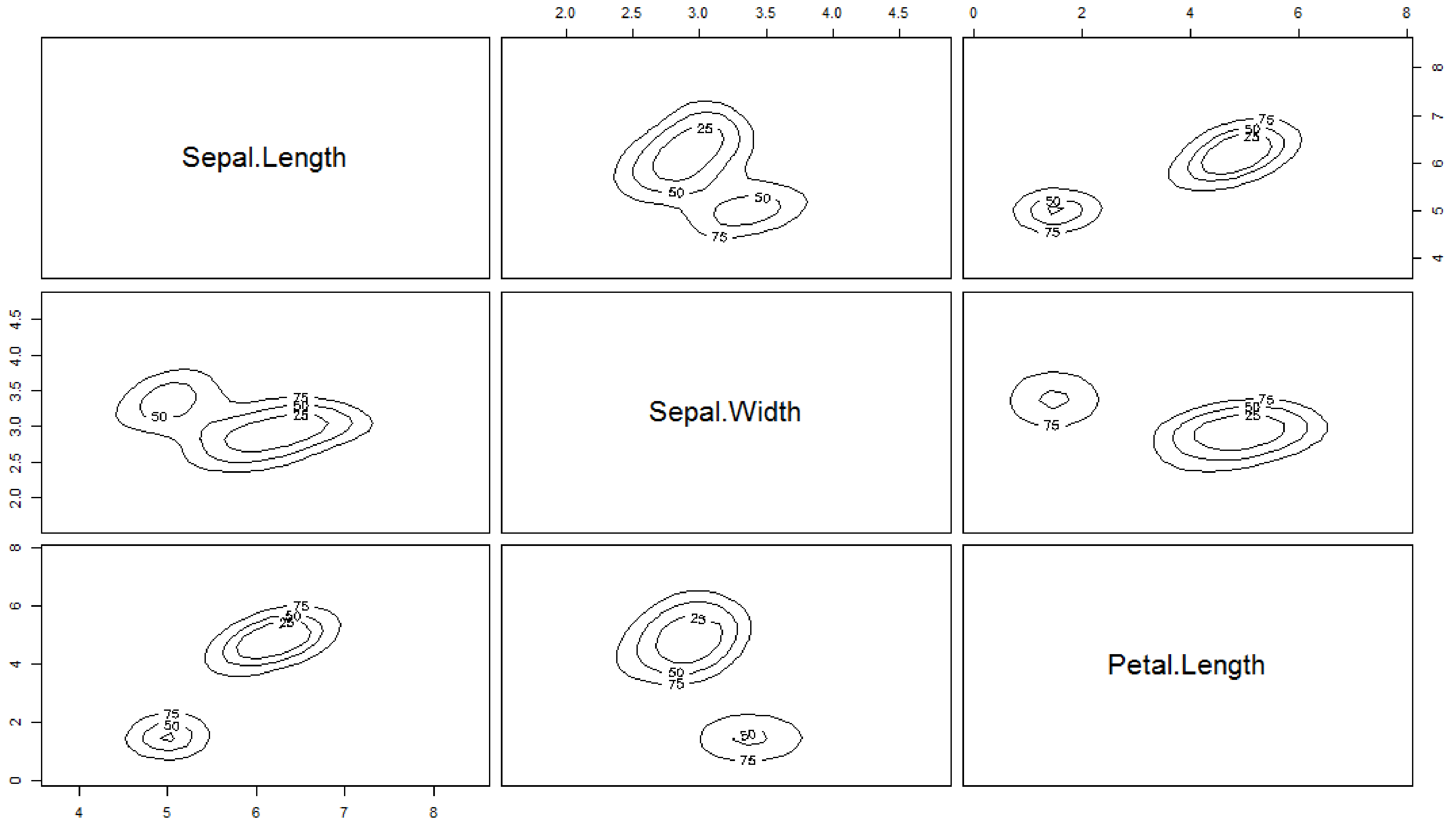
```
### 3d Clustering
y <- iris[,c(1,2,3)]
pdf <- kepdf(y)
summary(pdf)
plot(pdf)
```

```
The highest density data point has position 64 in the sample data

Rows of  75 % top density data points: 1 2 3 4 5 7 8 10 11 12 13 18 20 21 22 24 25 26 27
28 29 30 31 32 35 36 37 38 40 41 43 44 45 46 47 48 49 50 51 52 53 55 56 57 59 62 64 65 66
67 68 70 71 72 73 74 75 76 77 78 79 80 81 83 84 85 87 89 90 91 92 93 95 96 97 98 100 101
102 103 104 105 111 112 113 114 115 116 117 121 122 124 125 127 128 129 130 133 134 135 1
38 139 140 141 142 143 144 145 146 147 148 150

Rows of  50 % top density data points: 1 3 5 8 10 12 18 22 24 25 27 28 29 30 31 35 36 38
40 41 44 48 50 52 53 55 56 59 62 64 66 67 68 72 74 75 76 78 79 83 84 87 92 93 95 96 97 98
100 102 104 105 111 112 113 115 116 117 121 122 124 127 128 129 133 134 138 139 140 141 1
42 143 146 148 150

Rows of  25 % top density data points: 1 8 18 27 40 41 44 50 55 56 59 64 68 74 75 78 79 8
4 87 92 97 98 100 104 111 113 115 117 124 127 128 134 138 139 141 146 148 150
```

See ***Kernel_Density_Clustering_iris.R***

QuantUniversity, LLC
www.quantuniversity.com

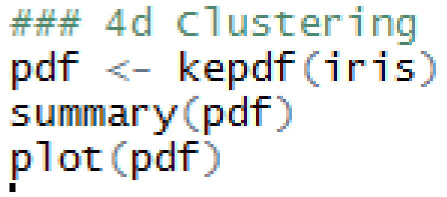
Kernel density clustering (R)



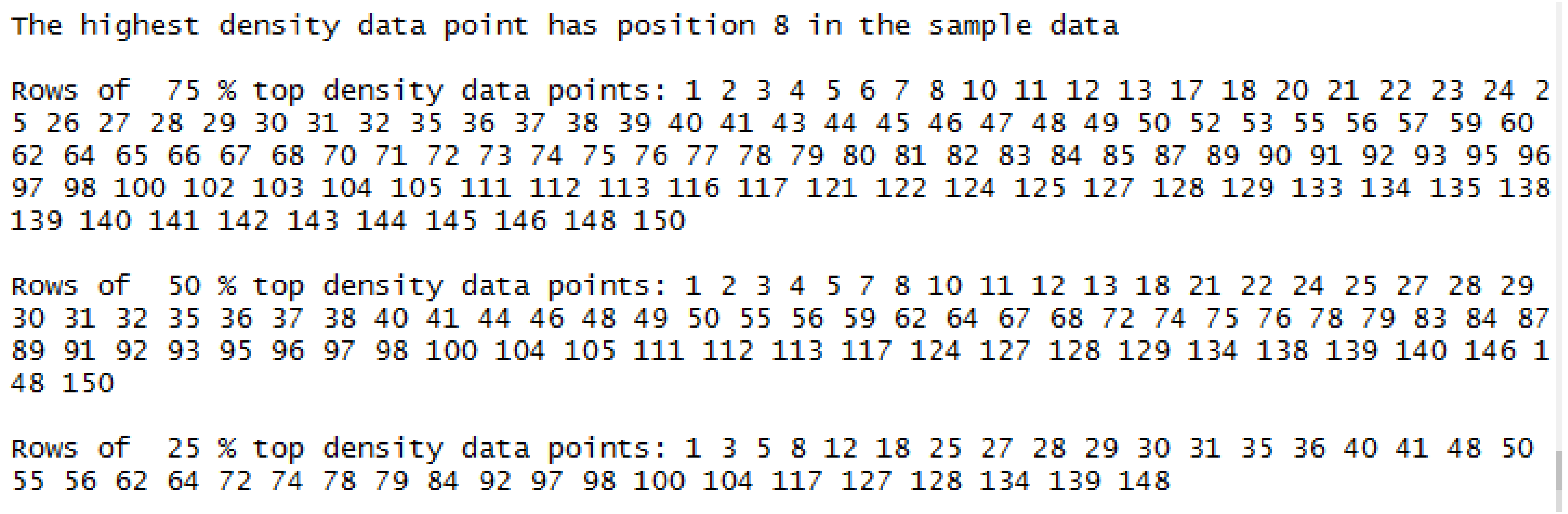
See ***Kernel_Density_Clustering_iris.R***

QuantUniversity, LLC
www.quantuniversity.com

Kernel density clustering (R)



```
### 4d Clustering
pdf <- kepdf(iris)
summary(pdf)
plot(pdf)
```

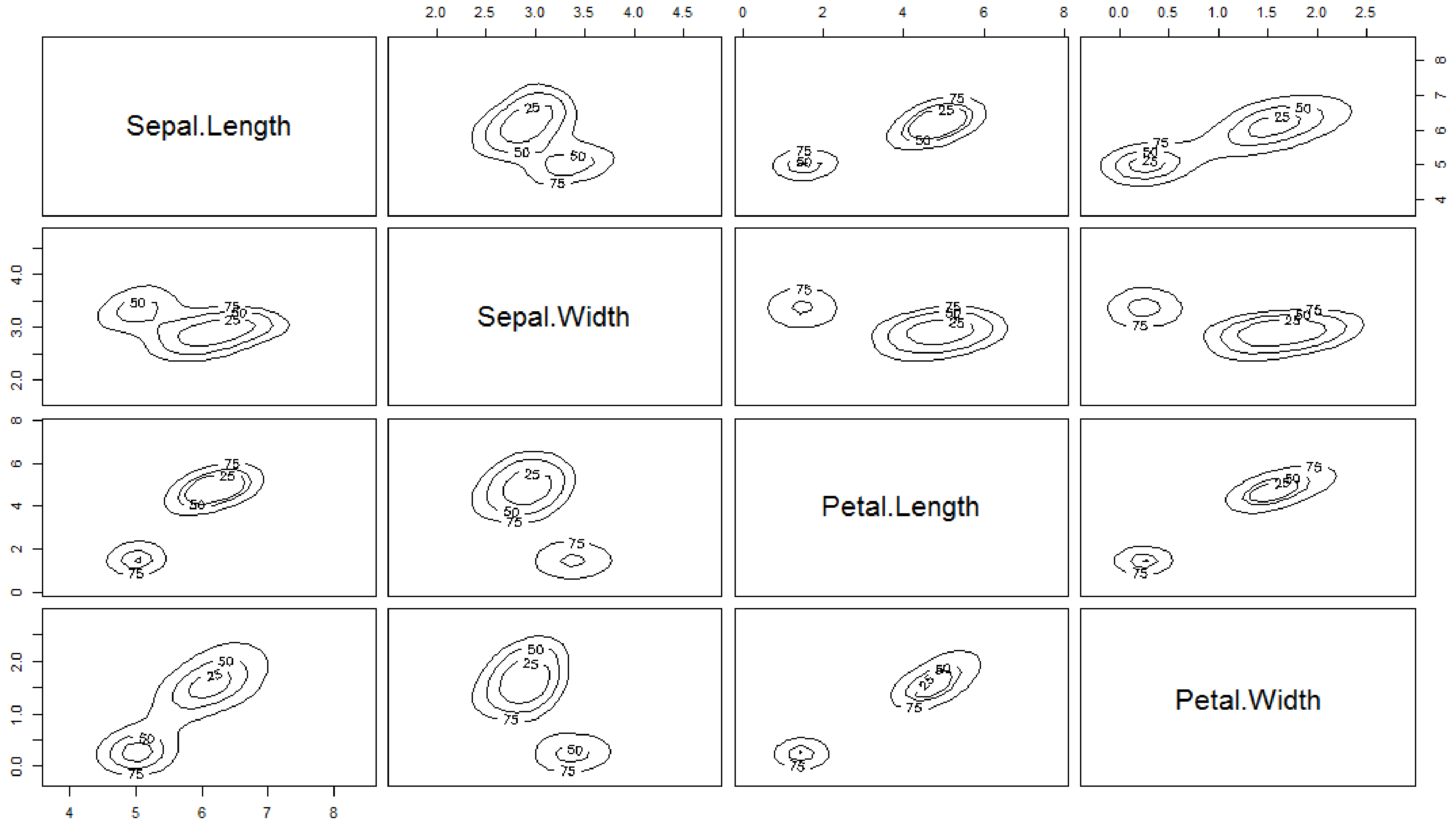


```
The highest density data point has position 8 in the sample data

Rows of  75 % top density data points: 1 2 3 4 5 6 7 8 10 11 12 13 17 18 20 21 22 23 24 2
5 26 27 28 29 30 31 32 35 36 37 38 39 40 41 43 44 45 46 47 48 49 50 52 53 55 56 57 59 60
62 64 65 66 67 68 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 87 89 90 91 92 93 95 96
97 98 100 102 103 104 105 111 112 113 116 117 121 122 124 125 127 128 129 133 134 135 138
139 140 141 142 143 144 145 146 148 150

Rows of  50 % top density data points: 1 2 3 4 5 7 8 10 11 12 13 18 21 22 24 25 27 28 29
30 31 32 35 36 37 38 40 41 44 46 48 49 50 55 56 59 62 64 67 68 72 74 75 76 78 79 83 84 87
89 91 92 93 95 96 97 98 100 104 105 111 112 113 117 124 127 128 129 134 138 139 140 146 1
48 150

Rows of  25 % top density data points: 1 3 5 8 12 18 25 27 28 29 30 31 35 36 40 41 48 50
55 56 62 64 72 74 78 79 84 92 97 98 100 104 117 127 128 134 139 148
```

See ***Kernel_Density_Clustering_iris.R***

QuantUniversity, LLC
www.quantuniversity.com

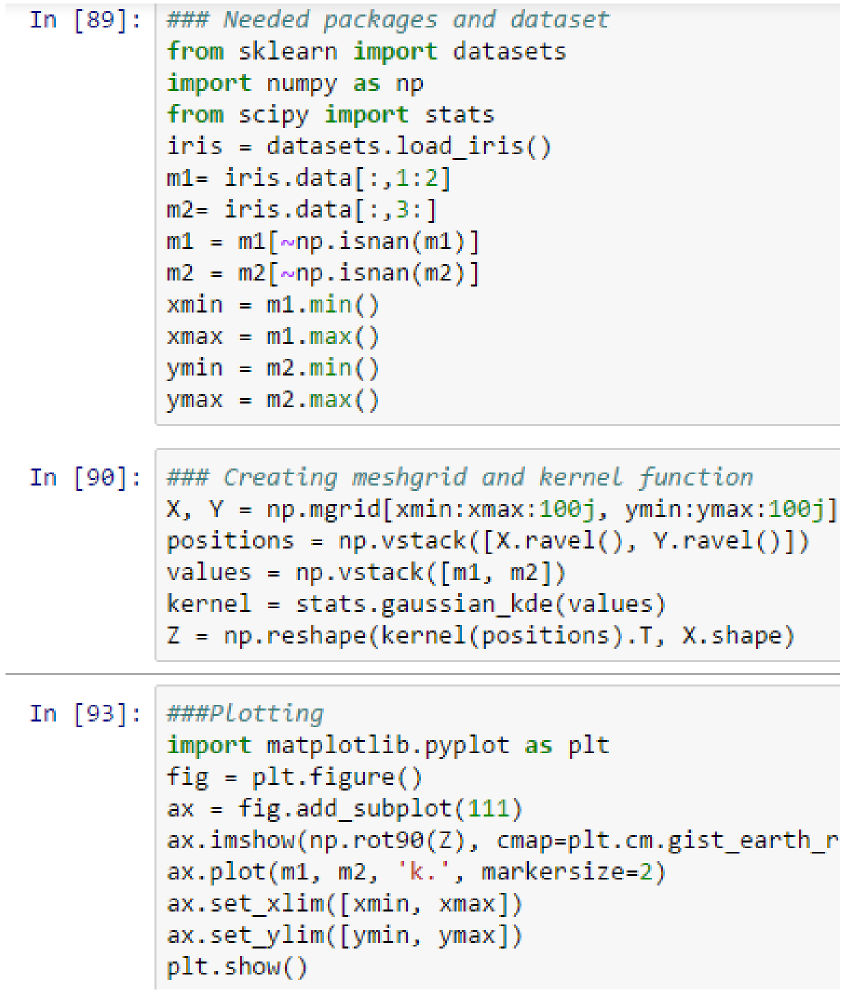
Kernel density clustering (R)



See *Kernel_Density_Clustering_iris.R*

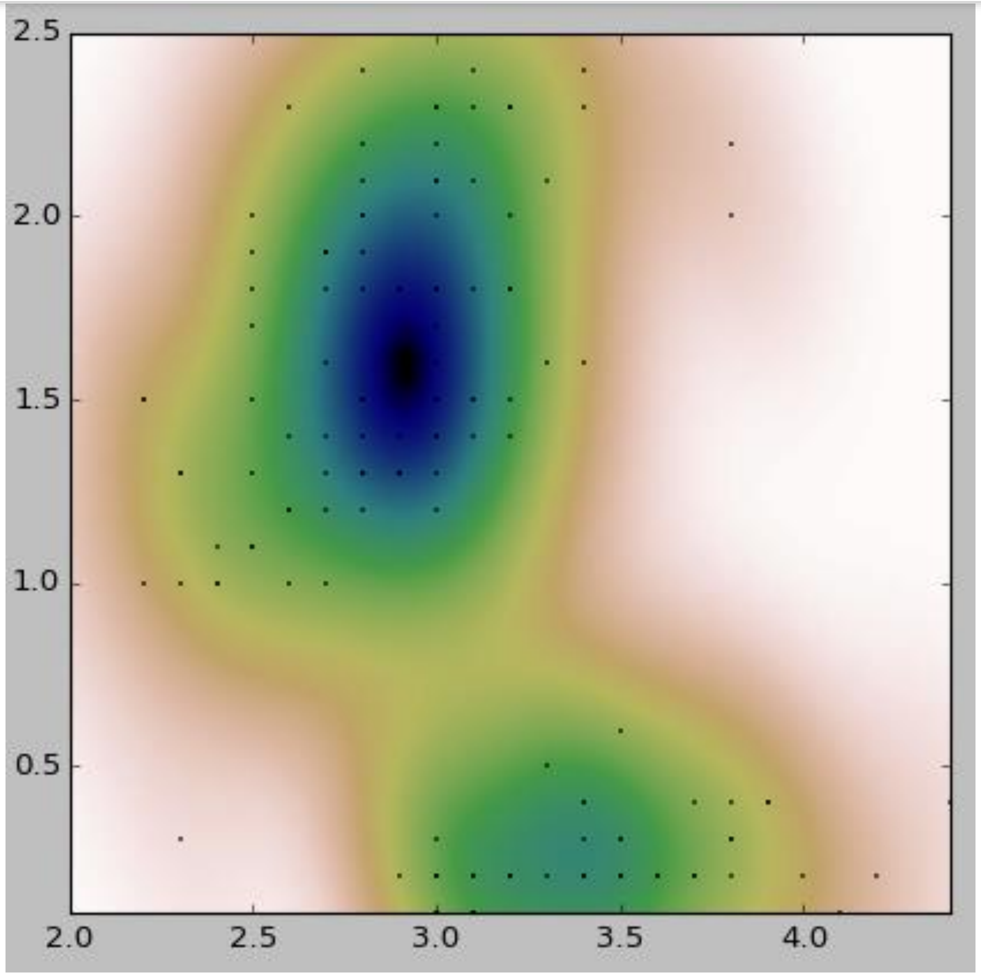
QuantUniversity, LLC
www.quantuniversity.com

Kernel density estimation (Python)



```
In [89]: ### Needed packages and dataset
         from sklearn import datasets
         import numpy as np
         from scipy import stats
         iris = datasets.load_iris()
         m1= iris.data[:,1:2]
         m2= iris.data[:,3:]
         m1 = m1[~np.isnan(m1)]
         m2 = m2[~np.isnan(m2)]
         xmin = m1.min()
         xmax = m1.max()
         ymin = m2.min()
         ymax = m2.max()

In [90]: ### Creating meshgrid and kernel function
         X, Y = np.mgrid[xmin:xmax:100j, ymin:ymax:100j]
         positions = np.vstack([X.ravel(), Y.ravel()])
         values = np.vstack([m1, m2])
         kernel = stats.gaussian_kde(values)
         Z = np.reshape(kernel(positions).T, X.shape)

In [93]: ###Plotting
         import matplotlib.pyplot as plt
         fig = plt.figure()
         ax = fig.add_subplot(111)
         ax.imshow(np.rot90(Z), cmap=plt.cm.gist_earth_r
         ax.plot(m1, m2, 'k.', markersize=2)
         ax.set_xlim([xmin, xmax])
         ax.set_ylim([ymin, ymax])
         plt.show()
```

Two dimensional kernel density estimation plot for iris dataset (X: Sepal width and Y: Petal width)



See ***Kernel_Density_Estimation_iris.ipynb***

QuantUniversity, LLC
www.quantuniversity.com

Association Rules

Association rules

- Is known as method of finding if-else type relationship (X => Y) among variables called as rule.
- There are three main criterion of customizing and sorting rules as:

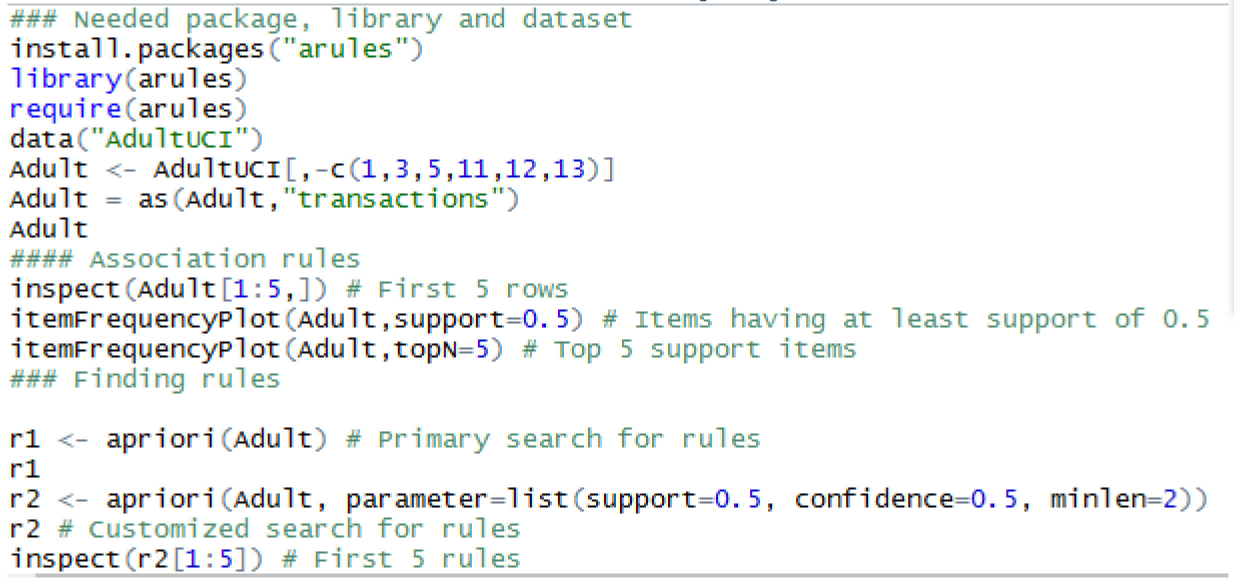
✓ Support (X => Y) = $\frac{frequency\ (X\ and\ Y)}{N}$

✓ Confidence (X => Y) = $\frac{frequency\ (X\ and\ Y)}{frequency(X)}$

✓ lift (X => Y) = $\frac{Support(X\ and\ Y)}{Support(X) * Support(Y)}$

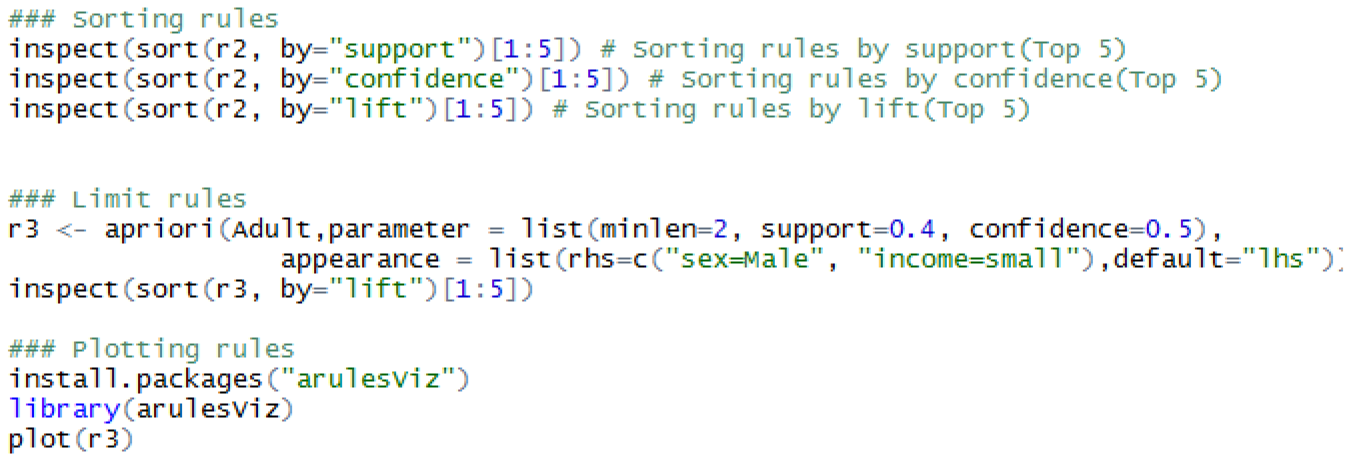
- Association rule method applies Apriori algorithm(https://en.wikipedia.org/wiki/Apriori_algorithm) to find the rules.
- The main application of association rule method is market basket analysis for point-of-sales datasets.

Association rules (R)



```
### Needed package, library and dataset
install.packages("arules")
library(arules)
require(arules)
data("AdultUCI")
Adult <- AdultUCI[,-c(1,3,5,11,12,13)]
Adult = as(Adult,"transactions")
Adult
#### Association rules
inspect(Adult[1:5,]) # First 5 rows
itemFrequencyPlot(Adult,support=0.5) # Items having at least support of 0.5
itemFrequencyPlot(Adult,topN=5) # Top 5 support items
### Finding rules

r1 <- apriori(Adult) # Primary search for rules
r1
r2 <- apriori(Adult, parameter=list(support=0.5, confidence=0.5, minlen=2))
r2 # Customized search for rules
inspect(r2[1:5]) # First 5 rules
```

Apriori function mines frequent item sets, association rules or association hyper edges using the Apriori algorithm. The Apriori algorithm employs level-wise search for frequent item sets.

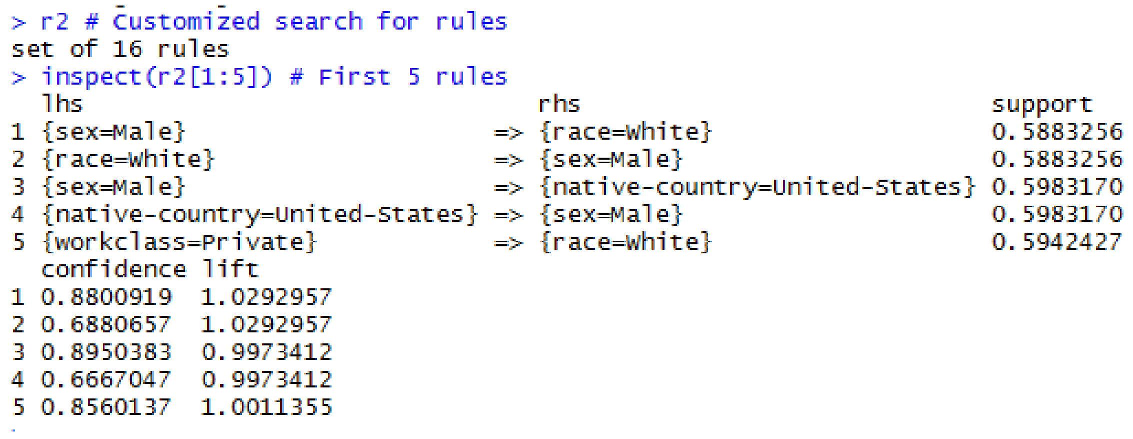


```
### Sorting rules
inspect(sort(r2, by="support")[1:5]) # Sorting rules by support(Top 5)
inspect(sort(r2, by="confidence")[1:5]) # Sorting rules by confidence(Top 5)
inspect(sort(r2, by="lift")[1:5]) # Sorting rules by lift(Top 5)


### Limit rules
r3 <- apriori(Adult,parameter = list(minlen=2, support=0.4, confidence=0.5),
                 appearance = list(rhs=c("sex=Male", "income=small"),default="lhs")
inspect(sort(r3, by="lift")[1:5])

### Plotting rules
install.packages("arulesViz")
library(arulesViz)
plot(r3)
```

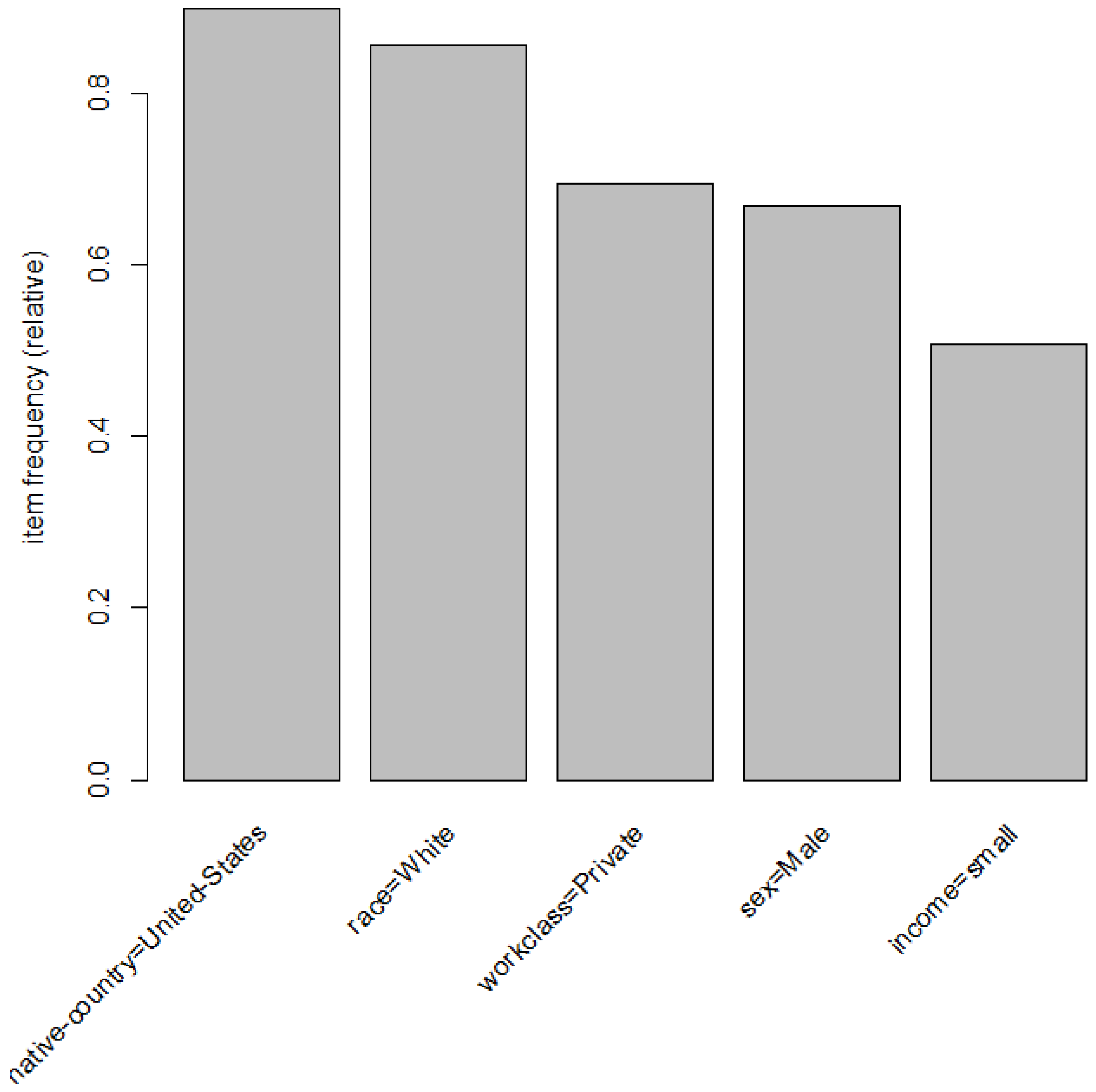
See *arule_AdultUCI.R*

QuantUniversity, LLC

Association rules (R)



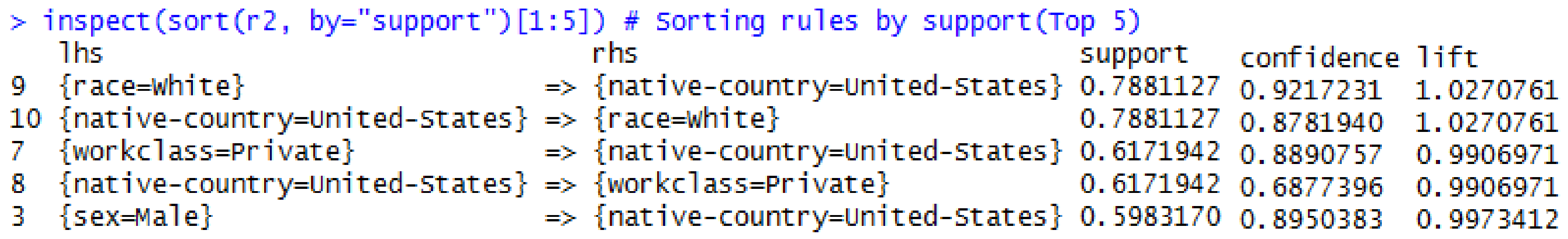
```
> r2 # Customized search for rules
set of 16 rules
> inspect(r2[1:5]) # First 5 rules
  lhs                                rhs                              support
1 {sex=Male}                      => {race=White}                     0.5883256
2 {race=White}                    => {sex=Male}                       0.5883256
3 {sex=Male}                      => {native-country=United-States} 0.5983170
4 {native-country=United-States} => {sex=Male}                       0.5983170
5 {workclass=Private}             => {race=White}                     0.5942427
  confidence lift
1 0.8800919  1.0292957
2 0.6880657  1.0292957
3 0.8950383  0.9973412
4 0.6667047  0.9973412
5 0.8560137  1.0011355
```



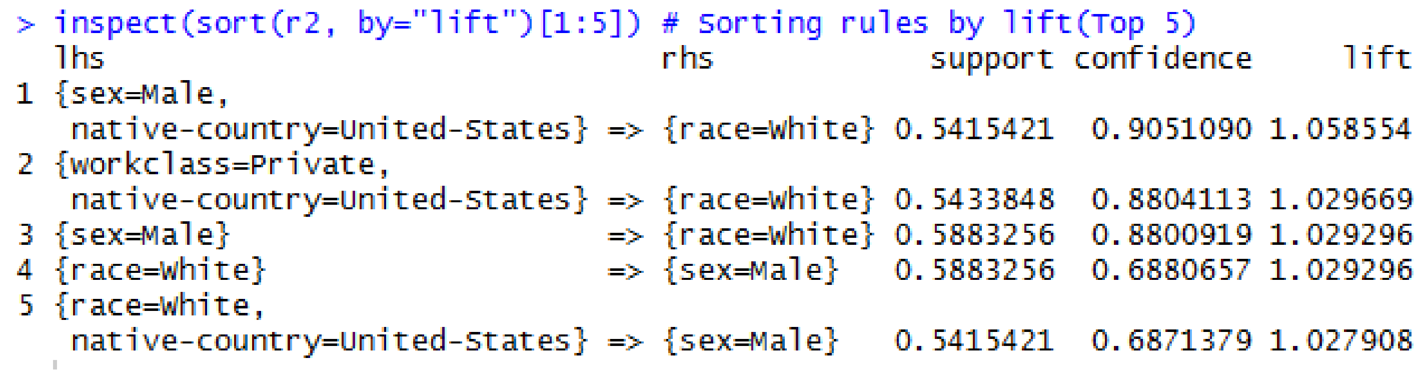
See *arule_AdultUCI.R*

QuantUniversity, LLC
www.quantuniversity.com

Association rules (R)



```
> inspect(sort(r2, by="support")[1:5]) # Sorting rules by support(Top 5)
   lhs                              rhs                             support   confidence lift
9  {race=White}                  => {native-country=United-States} 0.7881127 0.9217231  1.0270761
10 {native-country=United-States} => {race=White}                  0.7881127 0.8781940  1.0270761
7  {workclass=Private}           => {native-country=United-States} 0.6171942 0.8890757  0.9906971
8  {native-country=United-States} => {workclass=Private}           0.6171942 0.6877396  0.9906971
3  {sex=Male}                    => {native-country=United-States} 0.5983170 0.8950383  0.9973412
```

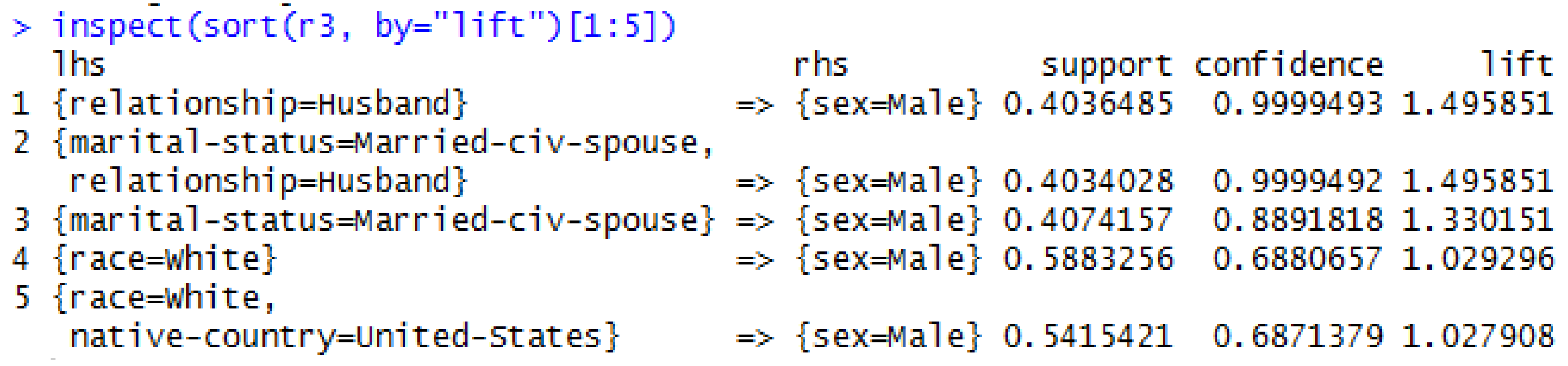


```
> inspect(sort(r2, by="lift")[1:5]) # Sorting rules by lift(Top 5)
  lhs                               rhs            support confidence     lift
1 {sex=Male,
   native-country=United-States} => {race=White} 0.5415421  0.9051090 1.058554
2 {workclass=Private,
   native-country=United-States} => {race=White} 0.5433848  0.8804113 1.029669
3 {sex=Male}                     => {race=White} 0.5883256  0.8800919 1.029296
4 {race=White}                   => {sex=Male}   0.5883256  0.6880657 1.029296
5 {race=White,
   native-country=United-States} => {sex=Male}   0.5415421  0.6871379 1.027908
```

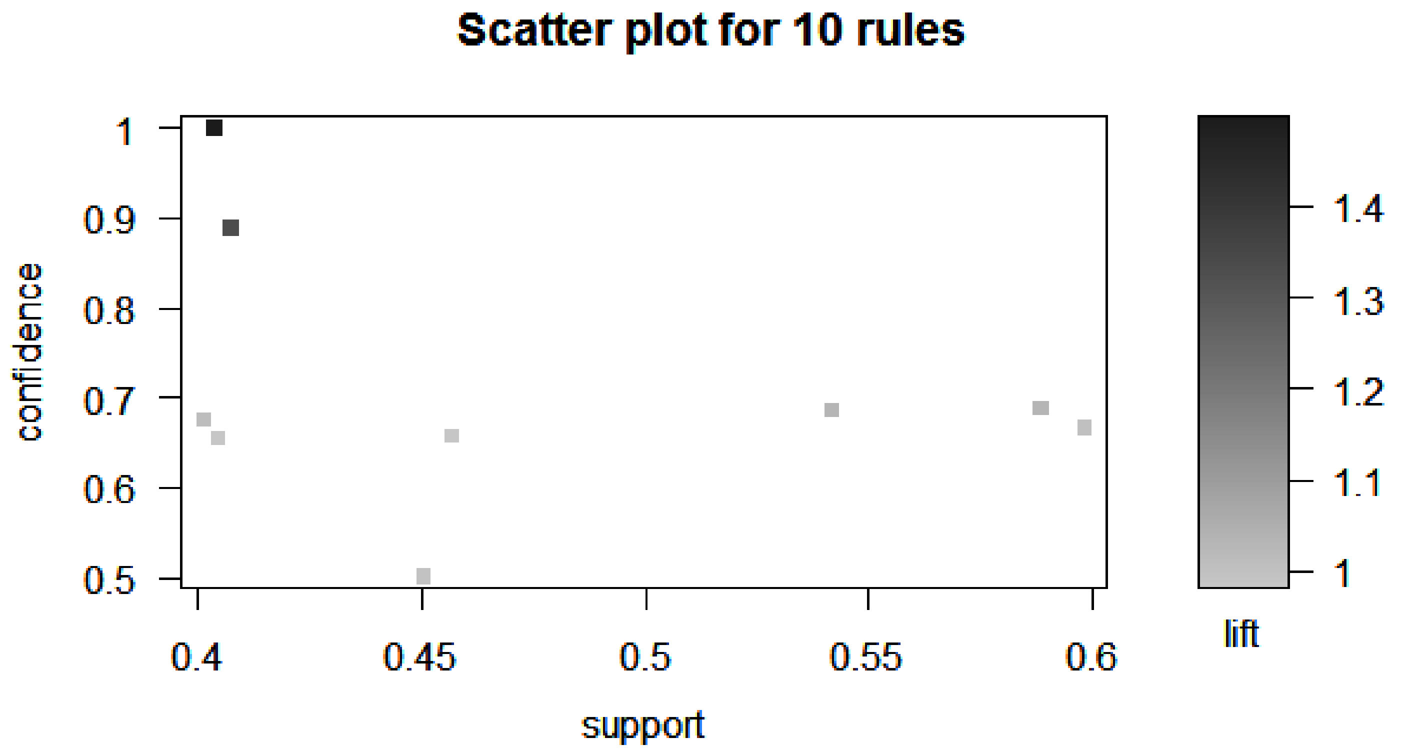
See ***arule_AdultUCI.R***

QuantUniversity, LLC
www.quantuniversity.com

Association rules (R)

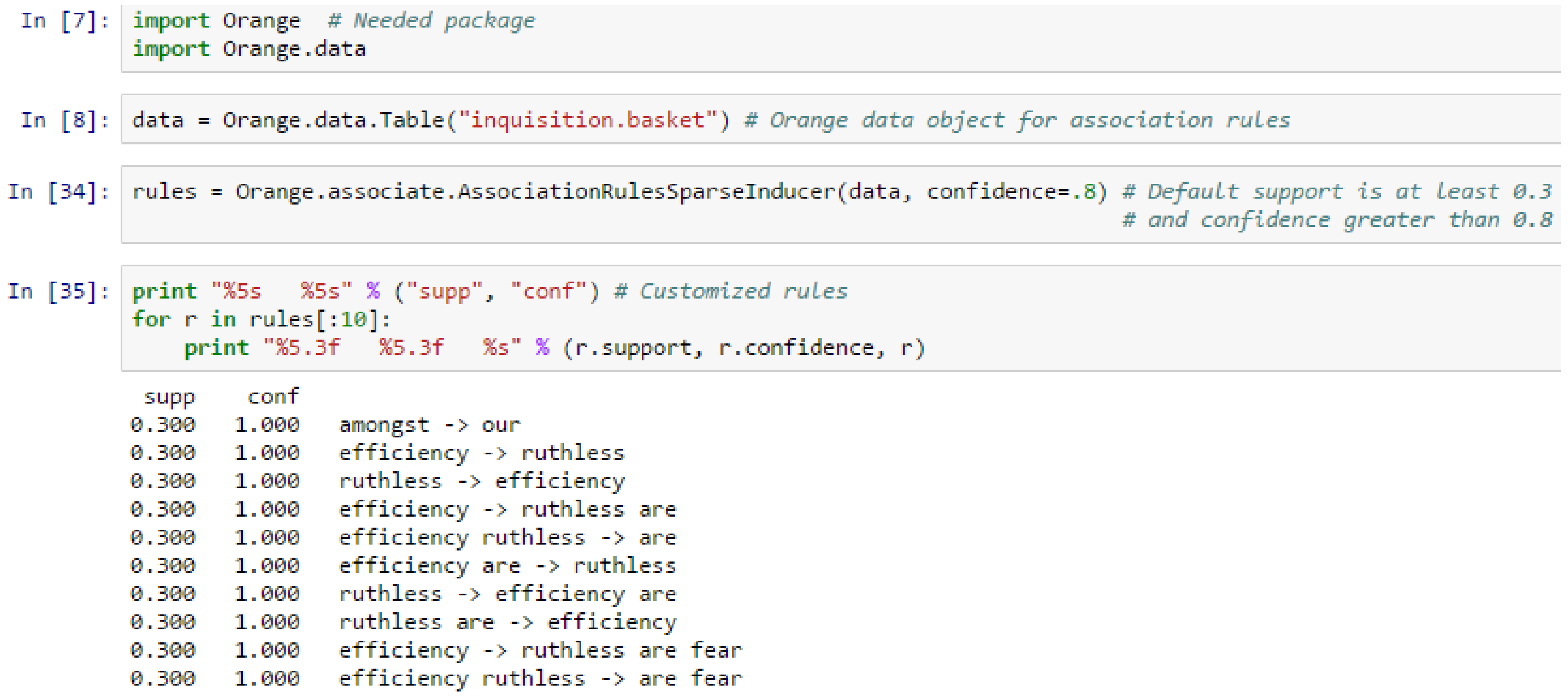


```
> inspect(sort(r3, by="lift")[1:5])
  lhs                                     rhs          support confidence     lift
1 {relationship=Husband}                => {sex=Male} 0.4036485  0.9999493 1.495851
2 {marital-status=Married-civ-spouse,
   relationship=Husband}                => {sex=Male} 0.4034028  0.9999492 1.495851
3 {marital-status=Married-civ-spouse} => {sex=Male} 0.4074157  0.8891818 1.330151
4 {race=White}                          => {sex=Male} 0.5883256  0.6880657 1.029296
5 {race=White,
   native-country=United-States}       => {sex=Male} 0.5415421  0.6871379 1.027908
```



See *arule_AdultUCI.R*

Association rules (Python)



```
In [7]:  import Orange  # Needed package
         import Orange.data
```

```
In [8]:  data = Orange.data.Table("inquisition.basket") # Orange data object for association rules
```

```
In [34]: rules = Orange.associate.AssociationRulesSparseInducer(data, confidence=.8) # Default support is at least 0.3
                                                                                     # and confidence greater than 0.8
```

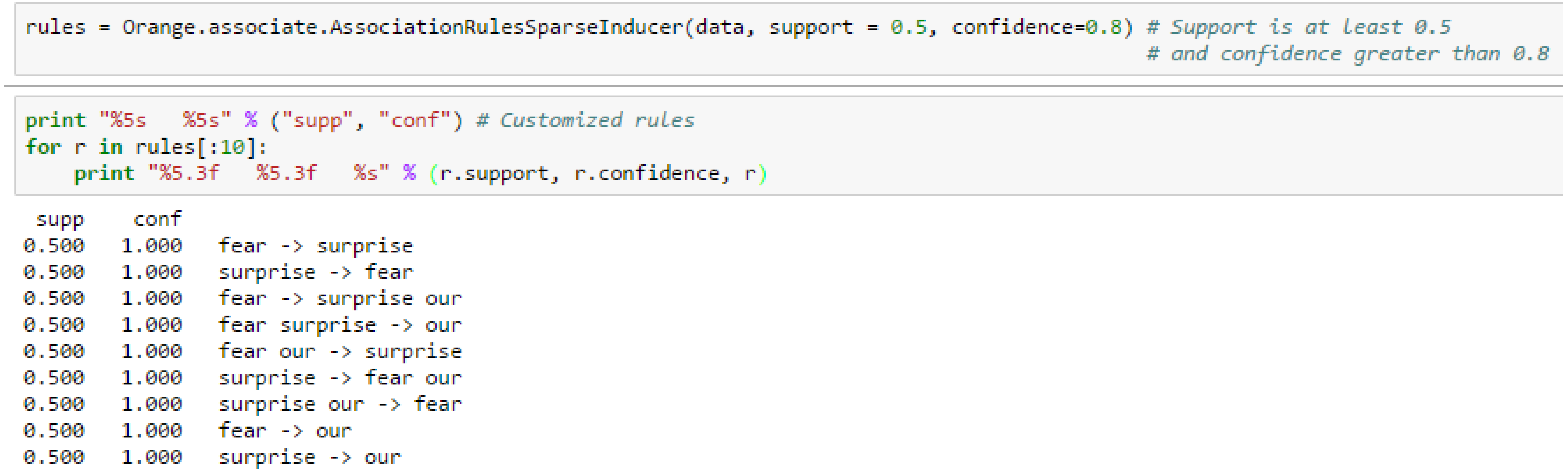
```
In [35]: print "%5s   %5s" % ("supp", "conf") # Customized rules
         for r in rules[:10]:
             print "%5.3f   %5.3f   %s" % (r.support, r.confidence, r)
```

```
 supp    conf
0.300   1.000   amongst -> our
0.300   1.000   efficiency -> ruthless
0.300   1.000   ruthless -> efficiency
0.300   1.000   efficiency -> ruthless are
0.300   1.000   efficiency ruthless -> are
0.300   1.000   efficiency are -> ruthless
0.300   1.000   ruthless -> efficiency are
0.300   1.000   ruthless are -> efficiency
0.300   1.000   efficiency -> ruthless are fear
0.300   1.000   efficiency ruthless -> are fear
```

See ***arules_inquising_basket.ipynb***

QuantUniversity, LLC
www.quantuniversity.com

Association rules (Python)



```
rules = Orange.associate.AssociationRulesSparseInducer(data, support = 0.5, confidence=0.8) # Support is at least 0.5
                                                                                           # and confidence greater than 0.8
```

```
print "%5s   %5s" % ("supp", "conf") # Customized rules
for r in rules[:10]:
    print "%5.3f   %5.3f   %s" % (r.support, r.confidence, r)
```

```
 supp    conf
0.500   1.000   fear -> surprise
0.500   1.000   surprise -> fear
0.500   1.000   fear -> surprise our
0.500   1.000   fear surprise -> our
0.500   1.000   fear our -> surprise
0.500   1.000   surprise -> fear our
0.500   1.000   surprise our -> fear
0.500   1.000   fear -> our
0.500   1.000   surprise -> our
```

See ***arules_inquising_basket.ipynb***

Summary

We have covered	Unsupervised learning methods
Clustering methods	✓ Definition of unsupervised learning methods ✓ K-Means clustering and its features such as linkage methods and distance function form ✓ Hierarchical clustering and its property as Dendrogram ✓ How to choose number of clusters by using elbow chart and bend graph ✓ To evaluate the density of the data by Kernel method ✓ Clustering based on Kernel density estimation
Association rules mining	✓ The rules (relationships) between variables as well as support, confidence and lift indices ✓ How to customize, sort and limit rules

Q&A

QuantUniversity, LLC

www.quantuniversity.com

Thank you!

Contact

Sri Krishnamurthy, CFA, CAP
Founder and CEO
QuantUniversity LLC.

LinkedIn srikrishnamurthy

www.QuantUniversity.com