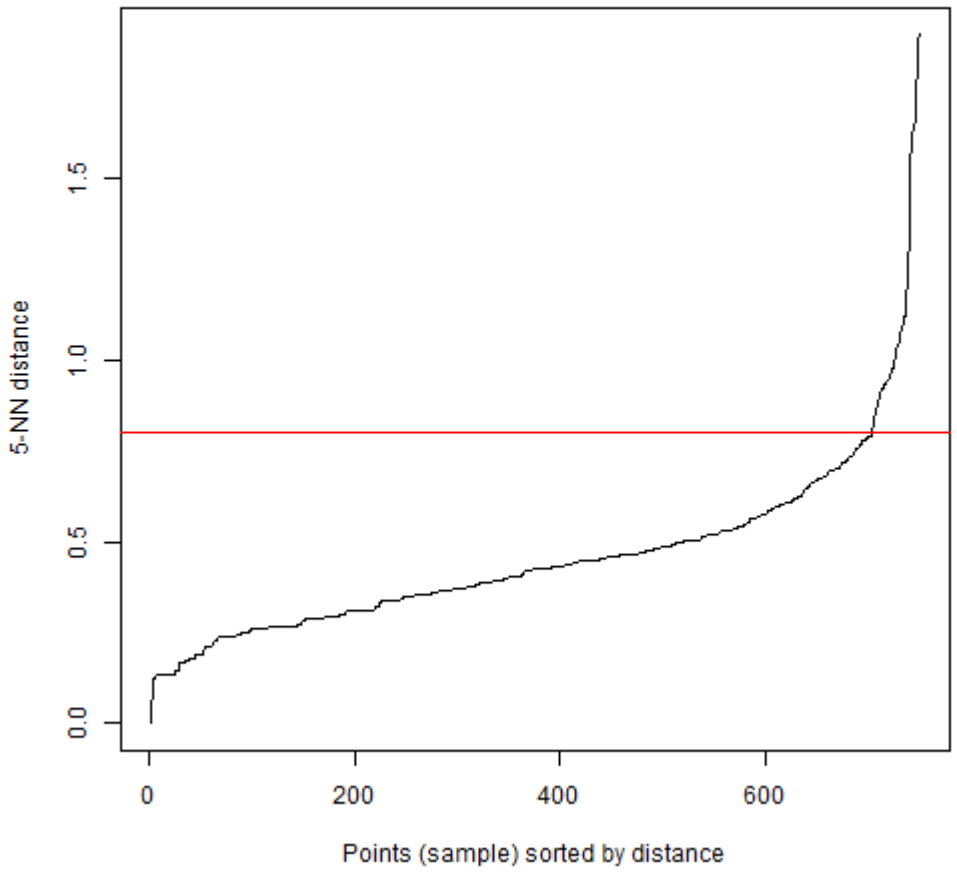
Density-Based Clustering Solutions

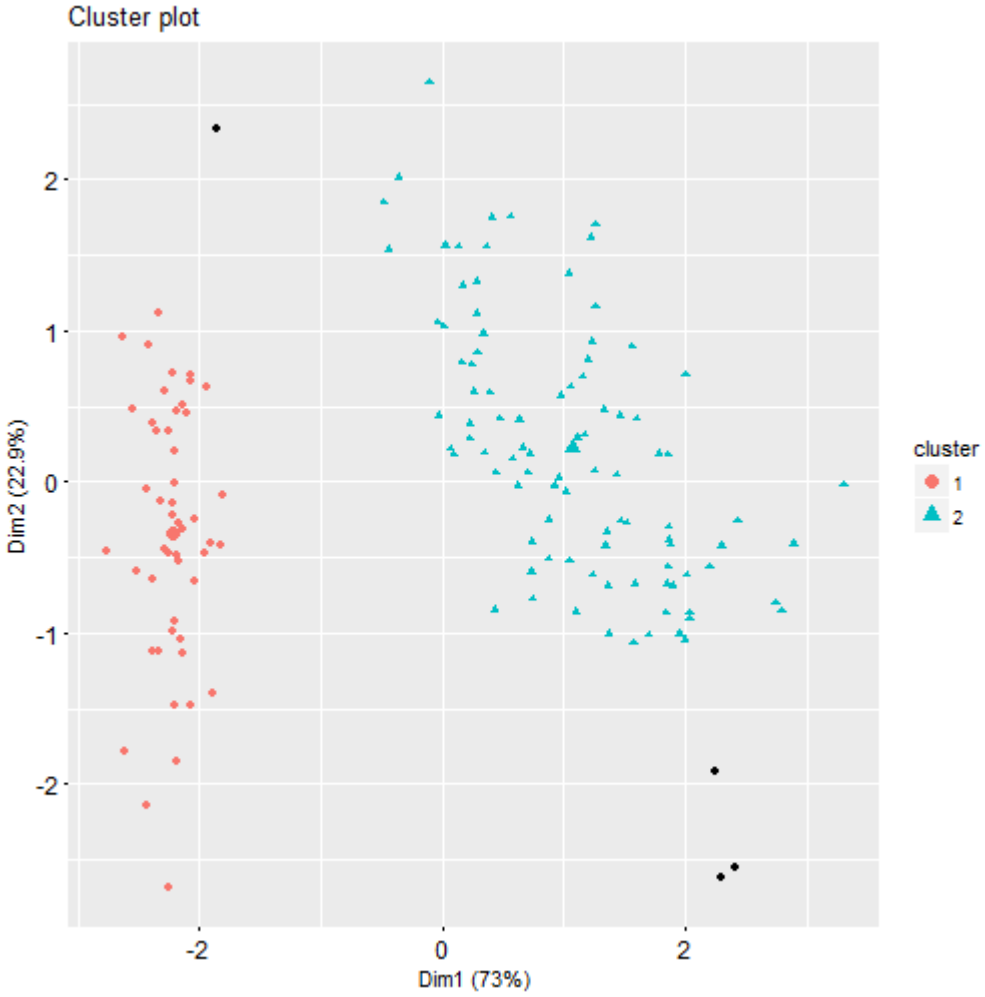
Below are the solutions to these exercises on density-based clustering.

```
####################
#                  #
#    Exercise 1    #
#                  #
####################
df <- iris[, -ncol(iris)]

####################
#                  #
#    Exercise 2    #
#                  #
####################
df <- scale(df)
df <- as.data.frame(df)

####################
#                  #
#    Exercise 3    #
#                  #
####################
require(dbscan)
kNNdistplot(df, k = 5)
abline(h = 0.8, col = "red")
```



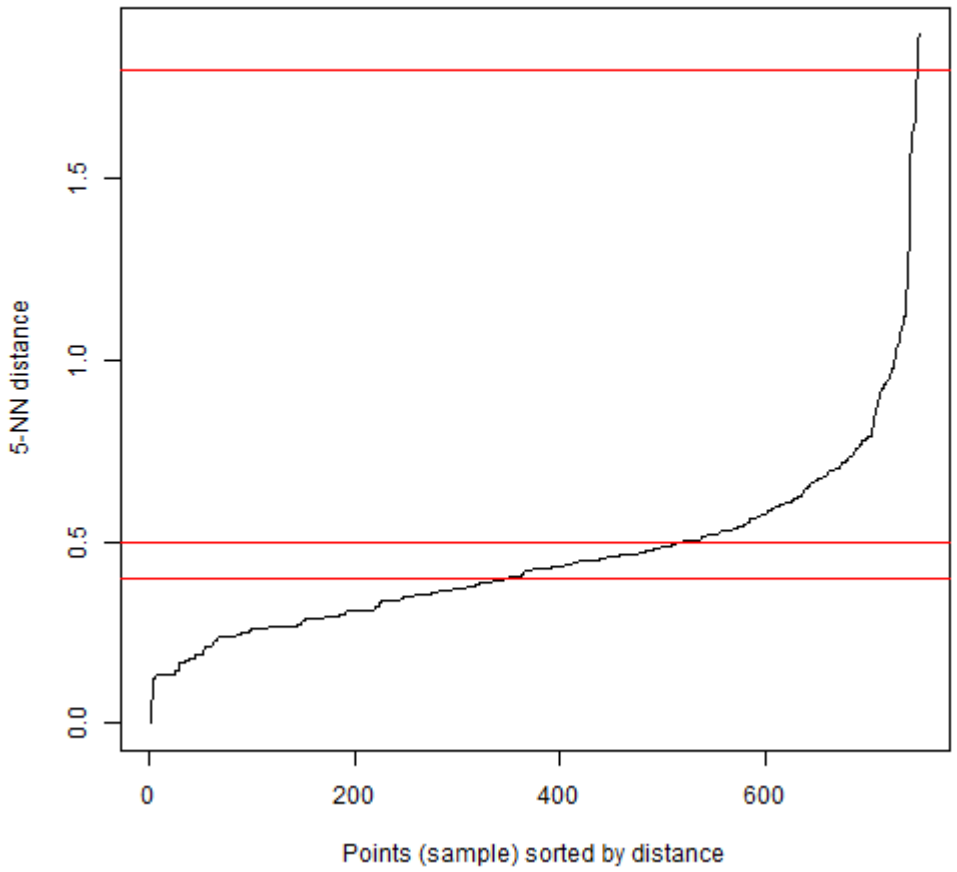

```
#####################
#                   #
#    Exercise 4     #
#                   #
#####################
require(dbscan)
db_clusters_iris <- dbscan(df, eps=0.8, minPts=5)
print(db_clusters_iris)

## DBSCAN clustering for 150 objects.
## Parameters: eps = 0.8, minPts = 5
## The clustering contains 2 cluster(s) and 4 noise points.
##
##  0  1  2
##  4 49 97
##
## Available fields: cluster, eps, minPts

#####################
```

```
#                     #
#    Exercise 5       #
#                     #
#######################
require(factoextra)
fviz_cluster(db_clusters_iris, df, ellipse = FALSE, geom =
"point")
```

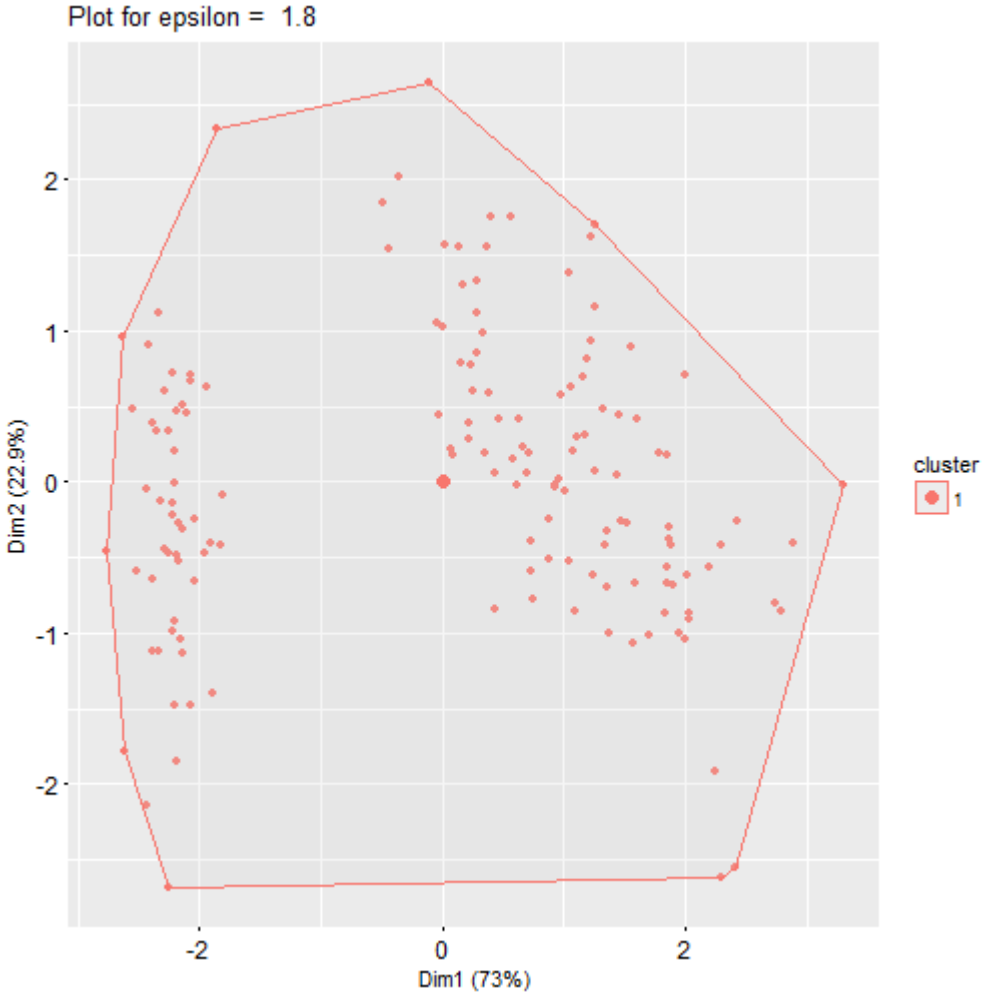


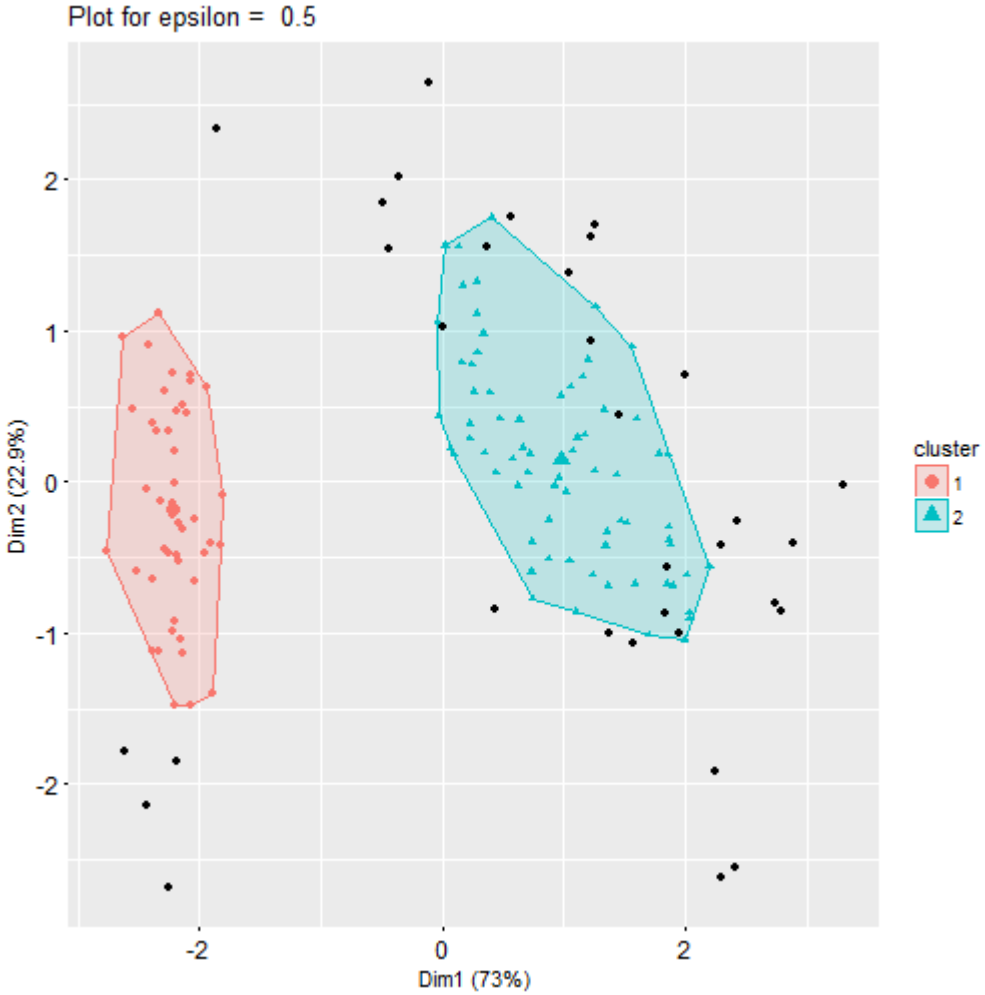

```
#######################
#                     #
#    Exercise 6       #
#                     #
#######################
df_copy <- df
df_copy[['cluster']] <- db_clusters_iris[['cluster']]
print(head(df_copy))

##   Sepal.Length Sepal.Width Petal.Length Petal.Width cluster
## 1   -0.8976739  1.01560199    -1.335752   -1.311052       1
```

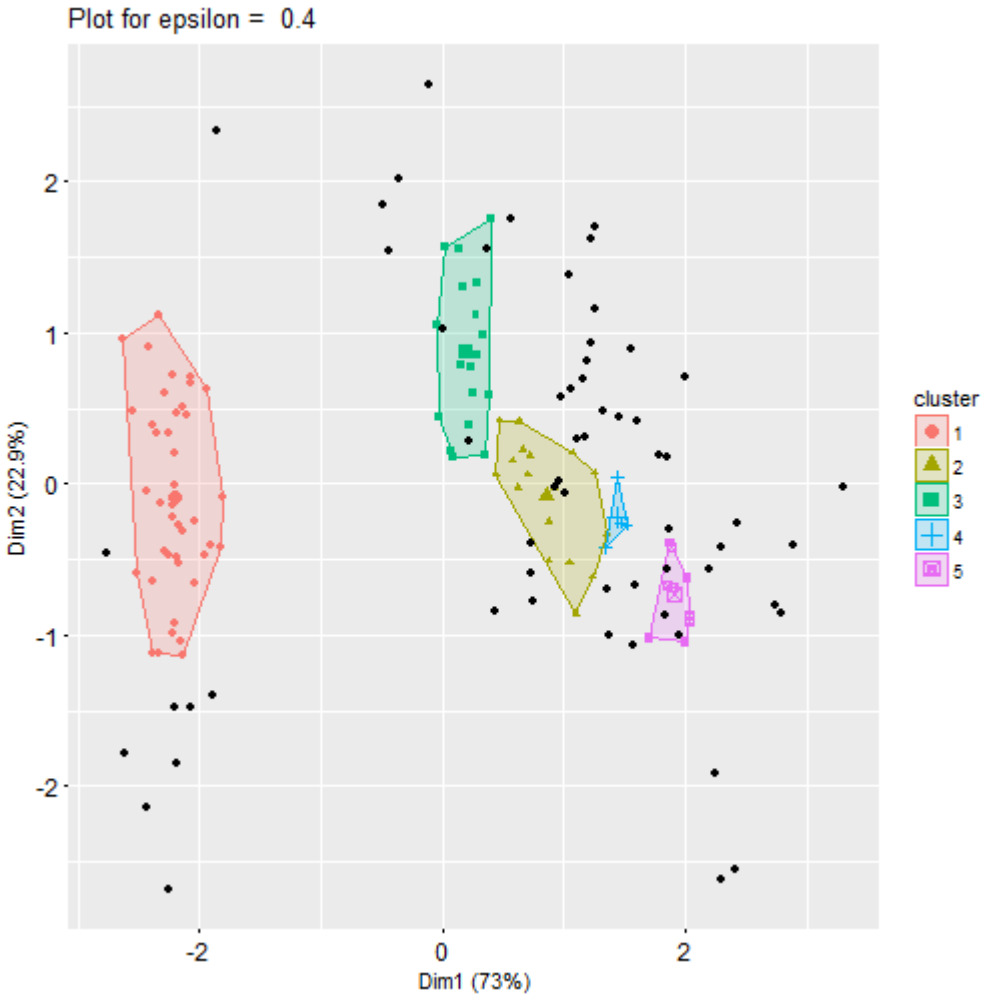
```
## 2   -1.1392005 -0.13153881    -1.335752   -1.311052       1
## 3   -1.3807271  0.32731751    -1.392399   -1.311052       1
## 4   -1.5014904  0.09788935    -1.279104   -1.311052       1
## 5   -1.0184372  1.24503015    -1.335752   -1.311052       1
## 6   -0.5353840  1.93331463    -1.165809   -1.048667       1

####################
#                  #
#    Exercise 7    #
#                  #
####################
require(dbscan)
require(factoextra)

# create a vector of epsilon values
epsilon_values <- c(1.8, 0.5, 0.4)

# plot the distribution of distances
kNNdistplot(df, k = 5)

# plot lines at epsilon values
for (e in epsilon_values) {
  abline(h = e, col = "red")
}
```



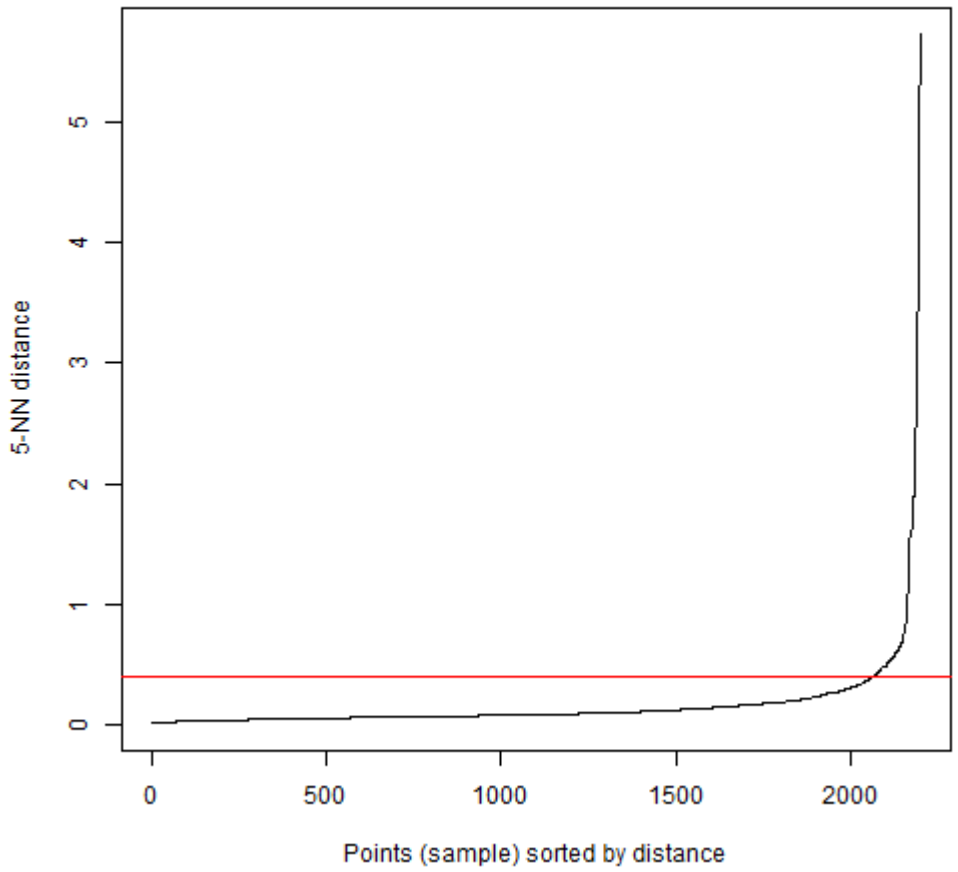

```
# find clusters for each epsilon value and plot those clusters
for (e in epsilon_values) {
  db_clusters_iris <- dbscan(df, eps=e, minPts=4)
  title <- paste("Plot for epsilon = ", e)
  g <- fviz_cluster(db_clusters_iris, df, ellipse = TRUE, geom
= "point",
                    main = title)
  print(g)
}
```


Plot for epsilon = 1.8
2
1
0
-1
-2
Dim2 (22.9%)
-2
0
2
Dim1 (73%)
cluster
1

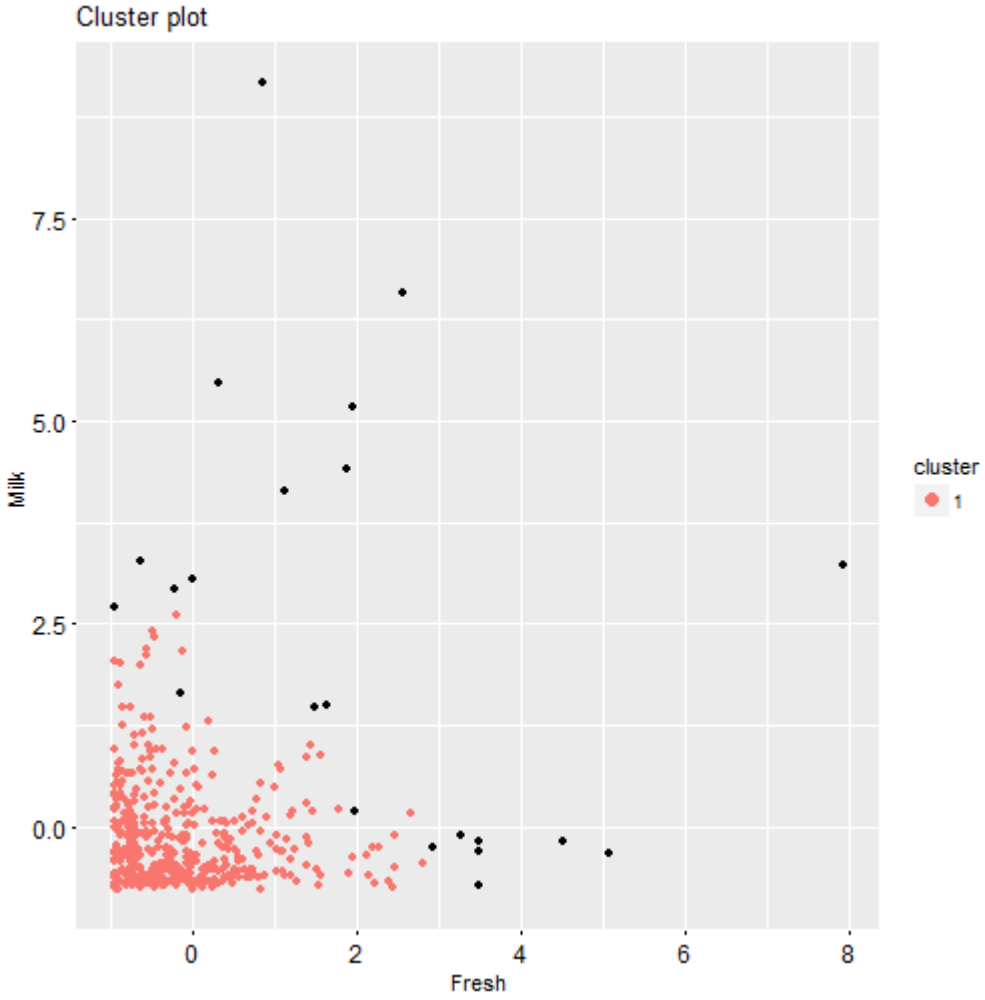

Plot for epsilon = 0.5
Dim2 (22.9%)
Dim1 (73%)
2
1
0
-1
-2
-2
0
2
cluster
1
2




```
#####################
#                   #
#    Exercise 8     #
#                   #
#####################
require(dbscan)
require(factoextra)

# load and prepare the data
customers <- read.csv("Wholesale customers data.csv")
customers <- customers[, c("Fresh","Milk")]
customers <- scale(customers)
customers <- as.data.frame(customers)

# plot the distribution of distances to the fifth nearest
neighbors
kNNdistplot(customers, k = 5)
abline(h = 0.4, col = "red")
```

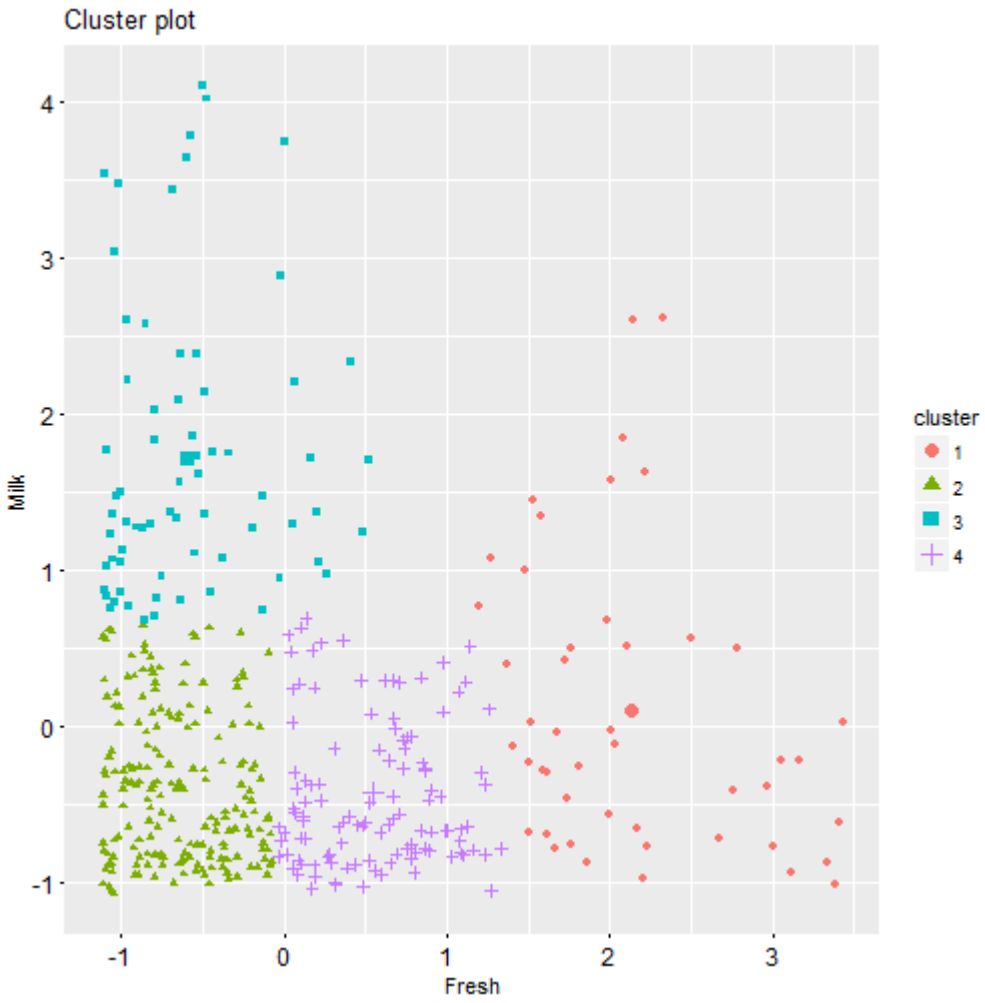



```
# find clusters
db_clusters_customers <- dbscan(customers, eps=0.4, minPts=5)
print(db_clusters_customers)

## DBSCAN clustering for 440 objects.
## Parameters: eps = 0.4, minPts = 5
## The clustering contains 1 cluster(s) and 22 noise points.
##
##   0   1
##  22 418
##
## Available fields: cluster, eps, minPts

# plot clusters
fviz_cluster(db_clusters_customers,  customers,  ellipse  =
FALSE, geom = "point")
```



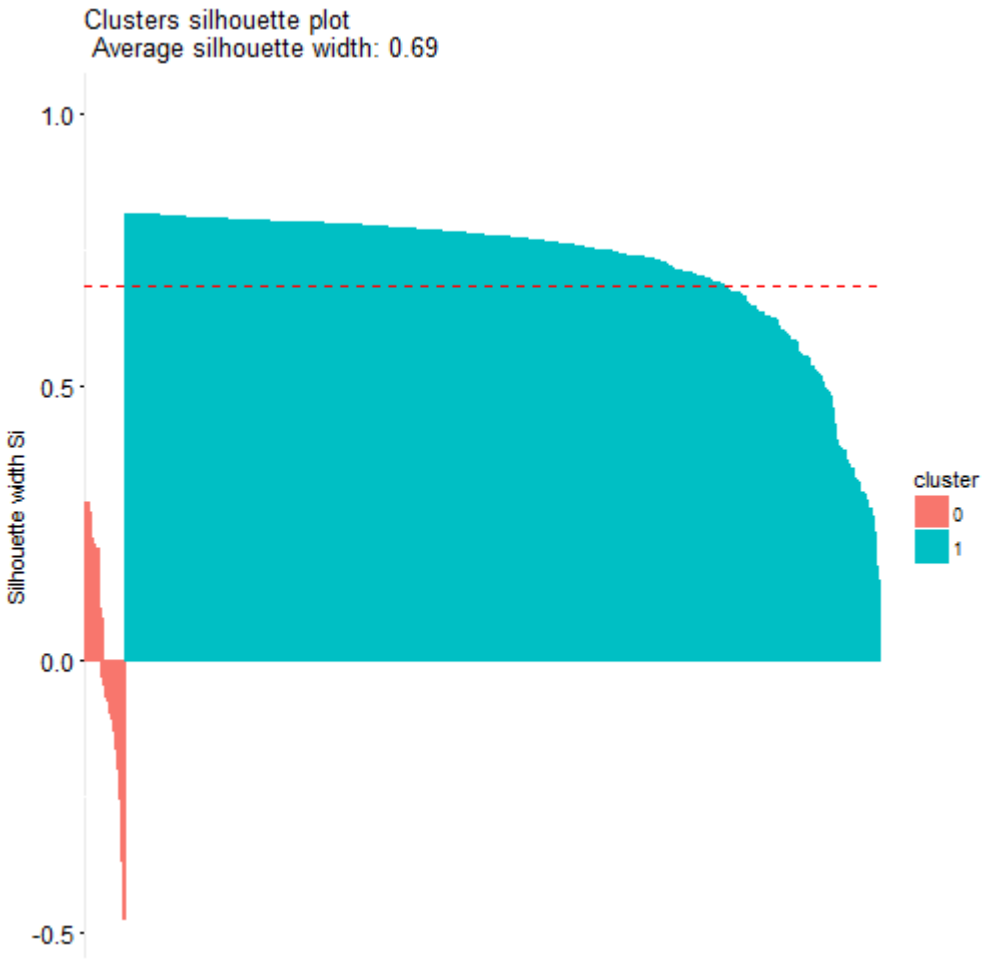

```
#####################
#                   #
#    Exercise 9     #
#                   #
#####################
require(factoextra)

# remove values beyond 2.5 standard deviations
customers_core <- customers[customers[['Fresh']] > -2.5 &
                                customers[['Fresh']] < 2.5, ]
customers_core  <-  customers_core[customers_core[['Milk']]  >
-2.5 &
                                     customers_core[['Milk']] <
2.5, ]

# find clusters and plot them
km_clusters_customers <- kmeans(customers_core, centers = 4,
nstart = 10)
fviz_cluster(km_clusters_customers,
```

```
            customers_core,
            ellipse = FALSE,
            geom = "point")
```




```
####################
#                  #
#    Exercise 10   #
#                  #
####################
require(dbscan)
require(cluster)
require(factoextra)

## DBSCAN results

# retrieve a vector of cluster assignments
db_clusters_vector <- db_clusters_customers[['cluster']]

# calculate distances between data points
db_distances <- dist(customers)
```

```
# get a silhouette information object
db_silhouette <- silhouette(db_clusters_vector, db_distances)

# plot the silhouette
fviz_silhouette(db_silhouette)

##   cluster size ave.sil.width
## 0       0   22          -0.02
## 1       1  418           0.72
```




```
## k-means results

# retrieve a vector of cluster assignments
km_clusters_vector <- km_clusters_customers[['cluster']]

# calculate distances between data points
km_distances <- dist(customers_core)

# get a silhouette information object
```

```
km_silhouette <- silhouette(km_clusters_vector, km_distances)

# plot the silhouette
fviz_silhouette(km_silhouette)

##   cluster size ave.sil.width
## 1       1   47          0.28
## 2       2  190          0.46
## 3       3   69          0.37
## 4       4  113          0.41
```