Lab #5

Purpose: This is our last lab in this semester. In this lab, we perform clustering algorithms.

1. Load Data iris.csv to the object iris.

iris.csv is a multivariate data set introduced by Ronald Fisher (1936): The use of multiple measurements in taxonomic problem. It contains 3 classes of a total 150 instances. Each class refers to a type of iris plant. There are five variables in this data set: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width and Species.

```
iris<-read.csv(file.choose(),header=T)

names(iris)

attach(iris)

dim(iris)

summary(iris)
```

Each instance is labeled with a class. We do not use the class in performing clustering, as it is an unsupervised technique. But after performing clustering, we can check the extent to which these classes agree with the result of the unsupervised technique. Accordingly, we store the label of each instance in the object iris.labs.

```
iris.labs=iris[,5]

iris.data=iris[,1:4]

dim(iris.data)
```

table(iris.labs) shows that we have 50 instances in each class.

```
table(iris.labs)
```

2. Hierarchical clustering

Since clustering relies on the distances between clusters, we use the function dist() to compute the 150 * 150 inter-observation Euclidean distance matrix and store it in the object data.dist.

```
data.dist=dist(iris.data)
```

The hclust() function implements hierarchical clustering. The first parameter in this function is the dissimilarity structure, i.e. the distance matrix. The second parameter is the linkage type, e.g. average, complete, single, or centroid. By default, it is the "complete" linkage type. Thus, hc1 stores the clustering dendrogram using the complete linkage clustering.

```
hc1=hclust(data.dist)
```

← complete (default)

```
hc2=hclust(data.dist, method="average")

hc3=hclust(data.dist, method="single")
```

We plot the obtained dendrograms using the plot() function. The numbers at the bottom of the plot identify each observation. We could see that the choice of linkage certainly affect the results obtained.

```
par(mfrow=c(1,3))

plot(hc1, main="Complete Linkage", xlab="", sub="",ylab="")

plot(hc2, main="Average Linkage", xlab="", sub="",ylab="")

plot(hc3, main="Single Linkage", xlab="", sub="",ylab="")
```

We need to determine where to cut the dendrogram so as to identify the labels for each observation. This decision has a strong impact on the clustering results. We usually need to try several choices, and use the one with the highest interpretability. In this case, as we already know there are a total of three classes, we can cut them to obtain three clusters using the cutree() function.

```
hc.clusters1=cutree(hc1,3)

hc.clusters2=cutree(hc2,3)

hc.clusters3=cutree(hc3,3)
```

We want 3 clusters (In this case, we already know that there're 3 clusters)

We then compare the cluster labels from all three methods with the original labels. All three linkage methods successfully identify all the flowers of species setosa into cluster 1. But they do not differentiate between virginica and versicolor quite well. The average linkage type looks better than the other two types.

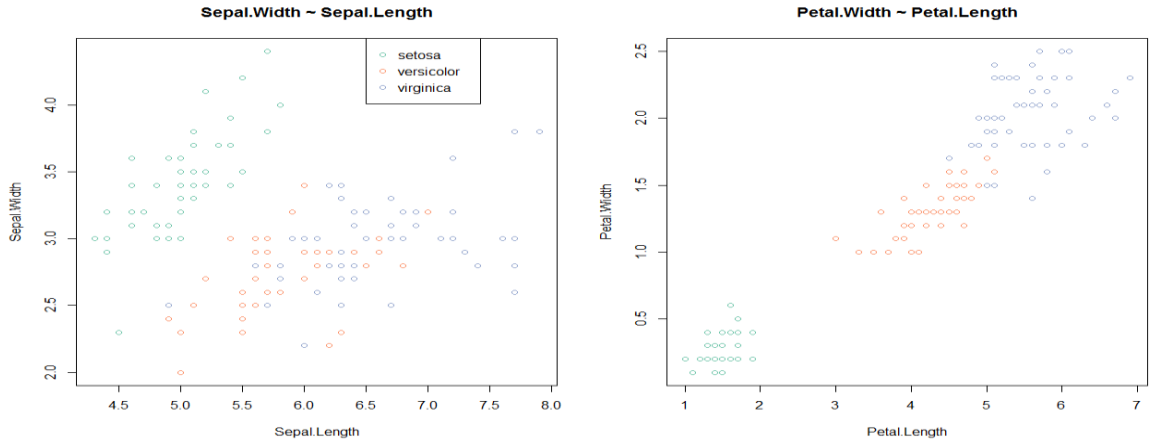
```
table(hc.clusters1,iris.labs)

table(hc.clusters2,iris.labs)

table(hc.clusters3,iris.labs)
```

average in the case works better (has the lowest error)

The plots showing the different species with each variable can tell why it is more difficult to differentiate between virginica and versicolor.



Based on the above two plots, variables Petal.Length and Petal.Width can better differentiate among the three species. Thus, we re-design all three clustering models using the only two variables.

```
iris.data2=iris[,3:4]

data.dist2=dist(iris.data2)

newhc1=hclust(data.dist2)

newhc2=hclust(data.dist2, method="average")
```

Petal Width Length

```
newhc3=hclust(data.dist2, method="single")

newhc.clusters1=cutree(newhc1,3)

newhc.clusters2=cutree(newhc2,3)

newhc.clusters3=cutree(newhc3,3)

table(newhc.clusters1,iris.labs)

table(newhc.clusters2,iris.labs)

table(newhc.clusters3,iris.labs)
```

Now we can see that the clustering algorithm with the average linkage does a much better job. Only five observations in versicolor and one in virginica do not fall in their own cluster.

We can plot the final dendrogram that produces the three clusters. The abline() function draws a horizontal line at height 1.3 on the dendrogram. newhc2 returns a summary of this object.

```
par(mfrow=c(1,1))

plot(newhc2, labels=iris.labs)

abline(h=1.3, col="red")

newhc2
```

3. K-Means clustering

In Step 1 of K-means, the initial cluster labels are assigned randomly, we first set a random seed. The function kmeans() performs K-means clustering. The first parameter is the data matrix. The second one is the pre-specified K, the number of clusters. In this case, we set it as 3. The third parameter, nstart, indicates the number of random assignments in Step 1. We first set nstart=1, which means that only run the function with one set of initial cluster assignment.

```
set.seed(1)

km.out1 =kmeans (iris.data,3, nstart =1)
```

value of k (clusters)

try initial assignment for 1 time

The kmeans function returns multiple values. km.out1$cluster returns the clustering results. km.out1$betweenss returns the between-cluster sum of squares. km.out1$withinss returns the within-cluster sum of squares for each cluster. km.out1$tot.withinss returns the total within-cluster sum of squares across all K clusters. km.out1$totss returns the total sum of squares in this data set.

=

tot. withinss + betweenss

```
km.out1

km.out1$cluster

km.out1$betweenss

km.out1$withinss

km.out1$tot.withinss
```

```
km.out1$totss
```

We compare the cluster labels with the original labels by using the function table().

```
table(km.out1$cluster,iris.labs)
```

Now let's explore if the initial cluster assignment affects the clustering results. We set a different random seed for Step 1 and then re-run the algorithm. Do we get the same clustering results? Doe the total within- cluster sum of squares change? Does the total sum of squares change?

```
set.seed(11)
```

use a different initial assignment.

```
km.out2 =kmeans (iris.data,3, nstart =1)

km.out2$betweenss

km.out2$withinss

km.out2$tot.withinss

km.out2$totss

table(km.out2$cluster,iris.labs)
```

We use nstart=1in the previous two models. If a value of nstart greater than one is used, then K-means clustering will be performed using multiple random assignments. It reports only the best results in terms of the total within-cluster sum of squares. Now let's try nstart=20. In practice, we usually try a larger value of nstart, e.g. 20, 50, to obtain the optimal model. Do we get the same clustering results as the previous two models? Doe the total within-cluster sum of squares change? Does the total sum of squares change?

4 dimensions

```
set.seed(1)

km.out3 =kmeans (iris.data,3, nstart =20)

km.out3$betweenss

km.out3$withinss

km.out3$tot.withinss

km.out3$totss

table(km.out3$cluster,iris.labs)
```

2 dims

As we learnt in hierarchal clustering, variables Petal.Length and Petal.Width can better differentiate among the three species. Thus, we run models using the only two variables.

```
iris.data2=iris[,3:4]
```

If we intend to standardize the variables to have mean zero and standard deviation one, we can use the scale() function. Again, do we get the same clustering results as the previous two models? Doe the total within-cluster sum of squares change? Does the total sum of squares change? Why?

?

```
set.seed(1)

sd.data=scale(iris.data2)
```

```
km.out4 =kmeans (sd.data,3, nstart =20)
km.out4$betweenss
km.out4$withinss
km.out4$tot.withinss
km.out4$totss
table(km.out4$cluster,iris.labs)
```

We can compare the results with the model without variable standardization.

```
km.out5 =kmeans (iris.data2,3, nstart =20)
km.out5$betweenss
km.out5$withinss
km.out5$tot.withinss
km.out5$totss
table(km.out5$cluster,iris.labs)
```

Once again, do we get the clustering results as the previous two models? Doe the total within-cluster sum of squares change? Does the total sum of squares change?

We use K=3 in this case because of our prior knowledge. Now let's use km.out5 to check if K=3 is an optimal choice (you can also check it using other models). We calculate the total within-cluster sum of squares from K=1 to K=10. Based on the plot, K=3 is a good choice.

```
wss = km.out5$totss
for (i in 2:10) wss[i] = sum(kmeans(iris.data2,centers=i)$withinss)
plot(1:10, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares", main="find the optimal value of K")
```

This concludes lab #5.