Assignment, R (Big Data Programming, IT 715A)
Demonstration 1b

Version 2016-11-08:

In this assignment, we will perform hierarchical clustering on customer data, which involves segmenting customers into different groups. You can download the data from Scio/Resources look for "customers.csv" in "data" folder (this URL if you're logged into SCIO https://scio.his.se/x/vRKfnH).

PART I: Hierarchical clustering

Task: "Clustering of customers"

Load the data into R:

```
customer= read.csv('customer.csv', header=TRUE)
head(customer)
```

Examine the data, to understand what we have:

```
str(customer)
```

To be able to cluster, we need to normalize the data:

```
customer = scale(customer[,-1])
```

Besides single-linkage, complete-linkage and average-linkage, we can use the Ward method (check out what that means). This refers to the sum of the squared distance from each point to the mean of the merged clusters (where μ is the mean vector of Ci UNION Cj, and Cx are cluster points):

$$\text{dist}(C_i, C_j) = \sum_{a \in C_i \cup C_j} \|a - \mu\|$$

You can read more about the Ward method here:
http://sites.stat.psu.edu/~ajw13/stat505/fa06/19_cluster/09_cluster_wards.html

DEMONSTRATION 1:

Now, you use `hclust()` to cluster (by agglomerative hierarchical clustering) distances between customers, using Euclidian distance and Ward clustering.

```
hc = hclust(NNNNNNN, method="ward.D2")
```

Replace `NNNNNNN` above with the **correct distance function** (find it out yourself).
Typing `hc`, you should now see:

```
Cluster method : ward.D2
Distance : Euclidean
Number of objects: 60
```

Be prepared to **explain the Ward method** during the demonstration.

Plot the dendogram for this distance tree:

```
plot(hc, hang = -0.01, cex = 0.7)
```

Instead, now try the `single` method and the simpler distance measure `dist(customer).` Store in the R object `hc2.`

Now, plot the dendogram for hc2 as well:

```
plot(hc2, hang = -0.01, cex = 0.7)
```

What **differences** do you see between these dendograms and **why**?

So what happended here?

Here we perform hierarchical clustering on customer data. First, we load the data from customer.csv, and then load it into the customer data frame. Within the data, we find five variables of customer account information, which are ID, number of visits, average expense, sex, and age. As the scale of each variable varies, we use the scale function to normalize the scale.

After the scales of all the attributes are normalized, we perform hierarchical clustering using the `hclust` function. We use the Euclidean distance as distance metrics, and use Ward's minimum variance method to perform agglomerative clustering. Finally, we use the plot function to plot the dendrogram of hierarchical clusters. We specify hang to display labels at the bottom of the dendrogram, and use `cex` to shrink the label to 70 percent of the normal size. In order to compare the differences using the `ward.D2` and single methods to generate a hierarchy of clusters, we draw another dendrogram using *single* in the preceding figure. We do not yet have clusters, but we have a hierarchical tree of the data based on distances.

Task: "Cutting the tree into clusters"

With a distance tree between customers, we can now group the data of the tree into clusters, using `cutree()`. Let's use 4 groups as a start:

```
fit = cutree(hc, k = 4)
```

Now, for all 60 data points, we can see which cluster the data point belongs to, by checking cluster labels:

```
fit

[1] 1 1 2 1 2 1 2 2 1 1 1 2 2 1 1 1 2 1 2 3 4 3 4 3 3 4 4 3 4
[30] 4 4 3 3 3 4 4 3 4 4 4 4 4 4 4 3 3 4 4 4 3 4 3 3 4 4 4 3 4
[59] 4 3
```

Also, we can see how many points there are in each cluster:

```
table(fit)

fit
1    2    3    4
11   8    16   25
```

We can also visualize how clusters are related in the tree:

```
plot(hc)
rect.hclust(hc, k = 4 , border="red")
```

We can also highlight a certain cluster (here, cluster 2):

```
rect.hclust(hc, k = 4 , which =2, border="red")
```

PART II: k-means clustering

Task: "Clustering of customers"

Here we use k-means clustering to cluster customer data. In contrast to hierarchical clustering, k-means clustering requires the user to input the number of k first. In the following example, we use `k=4`. In the generated fitted model we can see several describing parameters: the size of each cluster, the cluster means of the generated clusters, the cluster vectors (data point membership), the within-cluster-sum-of-squares by each cluster, and other available components.

k-means clustering is a *flat* clustering technique, which produces only one partition of the data, with k clusters. Unlike hierarchical clustering, which does not require a user to determine the number of clusters at the beginning, the k-means method requires this to be determined first. However, k-means clustering is much faster than hierarchical clustering as the construction of a hierarchical tree is very time consuming.

Now, first we cluster the customer data set

```
# set some arbitrarily chosen seed
set.seed(22)
fit = kmeans(customer, 4)
fit
```

Clusters plotted in different colors (2-dimensional plot, the first two independent variables are plotted, Visit.Time and Average.Expense):

```
plot(customer, col = fit$cluster)
```

How can you plot the clustering for variables 3 (sex) and 4 (age)?

PART III: Comparing clusterings and finding number of clusters

Comparing three clusterings

Install the fpc package

```
install.packages("fpc")
library(fpc)
```

Prepare three clusterings (single/hierarchical, complete/hierarchical and k means), all using 4 clusters as start:

```
single_c = hclust(dist(customer), method="single")
hc_single = cutree(single_c, k = 4)

complete_c = hclust(dist(customer), method="complete")
hc_complete = cutree(complete_c, k = 4)

set.seed(22)
km = kmeans(customer, 4)
```

We want to know how "cohesive" the clusters are, and can use "within-sum-of-squares" (WSS) and the Silhouette metrics:

> **WSS** (Assuming x is the given set of observations, $S = \{ S_1, S_2 .. S_k \}$ denotes k partitions, and μ_i is the mean of S_i, then we can formulate the WSS function as follows):
>
> $$f = \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$$
>
> **Silhouette**:
>
> $$\text{Silhouette}(x) = \frac{b(x) - a(x)}{\max([b(x), a(x)])}$$
>
> *a(x)* is the average distance between x and all other points within the cluster, and *b(x)* is the minimum of the average distances between x and the points in the other clusters. The silhouette value usually ranges from 0 to 1; a value closer to 1 suggests the data is better clustered.

Check them out for the k-means cluster:

```
cs = cluster.stats(dist(customer), km$cluster)
cs[c("within.cluster.ss","avg.silwidth")]

sapply(list(kmeans = km$cluster, hc_single = hc_single,
hc_complete = hc_complete), function(c)
cluster.stats(dist(customer),
c)[c("within.cluster.ss","avg.silwidth")])
```

Now we can see the cluster parameters WSS and Silouette width for the three clusterings. How do you interpret these numbers?

Trying a good k for k means

We can try different k to see which settings are more useful than others. Let's try k 2 to 10. We check both the WSS and Silhouette metrics:

```
nk = 2:10
set.seed(22)

# WSS
WSS = sapply(nk, function(k) {kmeans(customer,
centers=k)$tot.withinss })
WSS
# plot it
plot(nk, WSS, type="l", xlab= "number of k", ylab="within sum of
squares")

# silhouette width
SW = sapply(nk, function(k) { cluster.stats(dist(customer),
kmeans(customer, centers=k)$cluster)$avg.silwidth})
SW
# plot it
plot(nk, SW, type="l", xlab="number of clusers", ylab="average
silhouette width")
```

Now, think about what these two plots tell us. What are the optimum settings for k, for the customer data set. You will be asked to explain and reason about these findings.

DEMONSTRATION 2:

In this demonstration, you respond to the following questions:

- Explain the **computational complexity** of Hierarchical clustering versus k-means clustering algorithms. Are they comparable or not? If so, how?

- How do you scatter plot k-means clustering for **other variables** than Visit.Time and Average.Expense?

- Reason about `within.cluster.ss` and `avg.silwidth` metrics that you got for the three clusterings above. What do they mean?

- The WSS metric captures the "cohesiveness" of the clusters. **Motivate the k** we should use, based on the WSS metric plot above.

- Also, **explain the intuition** for the Silhouette metrics and **motivate the best k** to be used for this data set based on the plot of the Silhouette plot above.