Additional Material for Unsupervised Learning

Before starting with the elbow method, let us understand **clustering** more deeply:

- Clustering groups similar objects into clusters.
- One of its applications is to discover distinct customer groups in the markets.

The following are the 3 major types of **clustering**:

- k-means clustering.
- Hierarchical Clustering.
- Data-Depth based Clustering.

In the **Unsupervised Learning Spoken Tutorial**, we implemented `k-means` clustering on the `iris` dataset. Here is a brief description about it:

- It splits the data into k distinct sub-groups/clusters.
- Each data point belongs to the cluster with the closest centroid.

Here some drawbacks of `k-means` clustering:

- It is dependent on initialisation of cluster centroids.
- It doesn't handle noisy points and outliers well.

Now we will understand the **elbow method** for determining the **optimal** number of clusters. For this, we will use the built in `iris` dataset.

Here is a brief introduction to the **elbow method**:

- This is used to find the optimal number of clusters.
- It plots variation against a number of clusters.

```
data("iris")
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
elbow=iris[,c(3,4)]
```

Our `iris` dataset has 5 columns, but the ones that are most relevant to us are `Petal.Length` and `Petal.Width`. Thus we subset the dataset to take these columns.

```
n_max=20
set.seed(200)
```

To perform **elbow method**, we should perform `k-means` clustering over a certain number of clusters and find the one where **within sum of squares** distance is optimal. We need to know the **maximum** number of clusters that we will use for testing. Here we set that value to 20. We also set a `seed` so that we can reproduce these results.

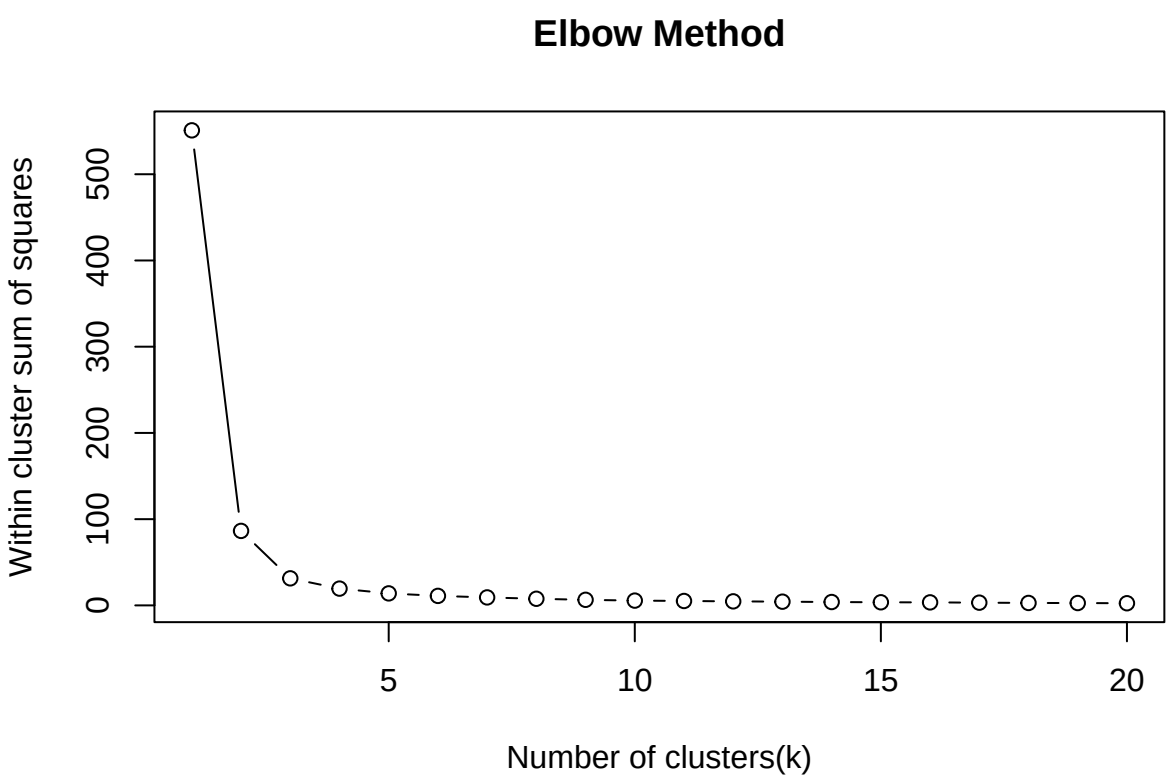
```
wss = sapply(1:n_max, function(k){kmeans(elbow, k, nstart = 20, iter.max = 20)$tot.withinss})
```

The **elbow method** has been implemented with the previous command. A brief explanation of what we have done:

- `sapply` : A `function` that allows us to iterate over a `vector` without using a `for` loop.
- `function(k)` : Performs `k-means` clustering over 1 to 20 clusters to determine the number of clusters that give the best results.
- `iter.max` : Sets max number of iterations to 20.
- `$tot.withinss` : Sums up **within sum of squares** values.

Now the results of the elbow method are plotted.

```
plot(1:n_max, wss, type = "b", xlab = "Number of clusters(k)",
     ylab = "Within cluster sum of squares"
     , main="Elbow Method")
```



A brief explanation of what we have done:

- `1:n_max`: Plotting **within sum of squares** value from 1 to 20.
- `type=b` : Plot a point graph with a line connecting all the points.
- `xlab` : Our **x-label** is Number of clusters.
- `ylab` : Our **y-label** is within cluster sum of squares.

From the plot we got this output, we notice a clear **elbow-like** bend at **n=3**. This will be taken as **optimal** number of clusters.