````
---
title: "Class 07 Machine Learning 1"
author: "Nicole Alfonso"
date: "2024-02-04"
output: pdf_document
---

# First up, kmeans()

Demo of using kmeans() function in R. FIrst make up some data with a known structure.

```{r}
# We will generate random numbers
tmp <- c(rnorm(30, -3), rnorm(30, 3))
tmp
# This creates two columns - Points that are (3,-3) and (-3, 3)
x <- cbind(x=tmp, y=rev(tmp))
# Plot a graph of the two columns
plot(x)
```

Now that we have made up data in 'x', we will see what kmeans() does to the data

```{r}
# We will use the function kmeans() to create 2 clusters, with 20 iterations (i.e. it is
running through the algorithim 20 diff times)
k <- kmeans(x, centers = 2, nstart = 20)
k
```
When we print k, 2 clusters are created, with each cluster containing 30 points each. It
also tells us our cluster means, which are the (x, y) coordinates of the means of the 2
clusters.

There are different available components that can help us learn more about the cluster.

> Q. How many points are in each cluster?

```{r}
k$size
```

> Q. How do we get to the cluster membership/assignment?

```{r}
k$cluster
```

> Q. What about cluster centers?

```{r}
k$centers
```

Now that we have returned the main results of our clusters, we will use them to generate a
plot of our kmeans().

```{r}
# To color the plot according to the cluster
plot(x, col=k$cluster)
# Ti distinguish the center value of each cluster with different color, in this case, blue
points(k$centers, col="blue", pch=15)
```
## Now for hclust()
````

We will cluster the same data 'x' with the `hclust()` function. In this case `hclust() requires a distance matrix as an input.

```{r}
hc <- hclust(dist(x))
hc
```
Let's plot our hclust result!

```{r}
plot(hc)
```

To get our cluster membership vector we need to "cut" the tree with `cutree()`

```{r}
groups <- cutree(hc, h=8)
groups
```

Now plot our data with the hclust() results.

```{r}
plot(x, col=groups)
```

Hands on with Principle Component Analysis (PCA)

First, we need to import our data.

Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

```{r}
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url)
```

```{r}
nrow(x)
ncol(x)
```

Checking your data. It is always a good idea to examine your imported data to make sure it meets your expectations.

```{r}
#To view the entire data frame
View(x)
#To view the first 6 rows of the data frame
head(x)
#To view the last 6 rows of the data frame
tail(x)
```

It appears that the data is not set properly, as the first column is labeled as 'X', giving us 5 variables not 4. To fix this we use the function rownames().

```{r}
#To class for the first column
rownames(x) <- x[,1]
#To remove the first column
x <- x[,-1]
head(x)
```

```

Another way to do it is by calling read.csv()

x <- read.csv(url, row.names=1)
head(x)

```{r}
#To find out the dimensions (x, y) of the data frame: dim()
dim(x)
```

Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

I think the solution for the 'row-names problem' that I prefer is the: 'x <- read.csv(url, row.names=1)' approach, as you have more control as to which column that you are changing. I think using the x[,-1] method would work if you only had to adjust the first column, because you might continue to erase more variables.

Spotting major differences and trends

```{r}
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```

```{r}
cols <- rainbow(nrow(x))
barplot( as.matrix(x), col=cols)
#It is hard to compare the data in this format.
```

```{r}
pairs(x, col=cols)
```

PCA to the rescue!
The main R PCA function is called `prcomp()`. We will need to give it the transpose of our input data.

```{r}

pca <- prcomp(t(x))
pca
```
```{r}
#Like kmeans(), there are different attributes for prcomp()
attributes(pca)
#Example: calling the center values.
pca$center
```

To make our new PCA plot (PCA score plot) we access `pca$x`

```{r}
pca$x
country_cols <- c("orange", "red", "blue", "green")
plot(pca$x[,1], pca$x[,2], xlab = "PC1", ylab = "PC2")
text(pca$x[,1], pca$x[,2], colnames(x), col = country_cols)
```

PCA of RNA-seq Data

First, we have to read our data.
```

```{r}
url2 <- "https://tinyurl.com/expression-CSV"
rna.data <- read.csv(url2, row.names=1)
head(rna.data)
```
```{r}
pca <- prcomp(t(rna.data))
summary(pca)
plot(pca)
```
```{r}
plot(pca$x[,1], pca$x[,2], xlab = "PC1", ylab = "PC2")
text(pca$x[,1], pca$x[,2], colnames(rna.data))
```