class07: Clustering and PCA

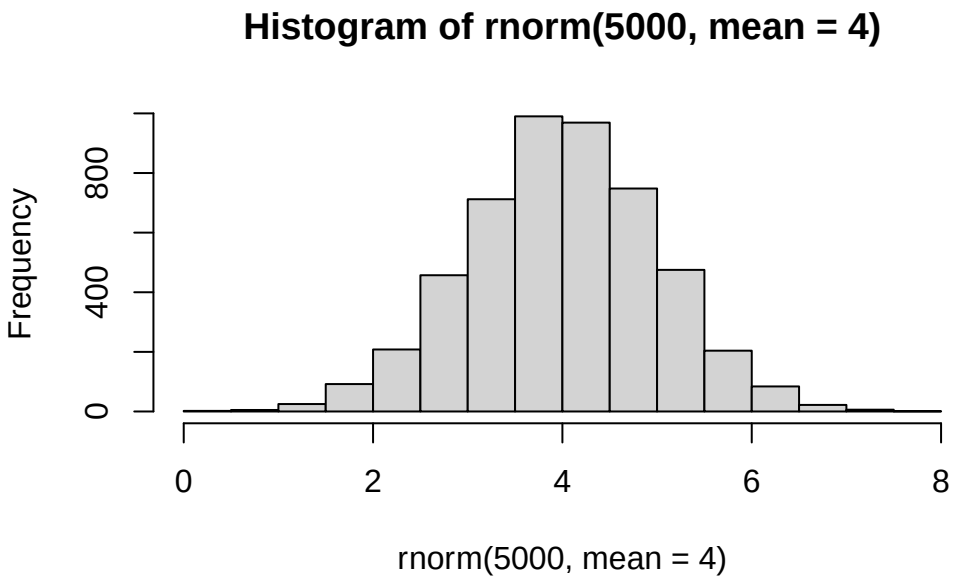
Jenny Zhou

Clustering

First let's make up some data to cluster, so we can get a feel for these methods.

we can use `rnorm()` function to get random numbers froma normal distribution around a given mean.

```
#generates a vector of normally distributed random numbers
hist(rnorm(5000, mean = 4))
```

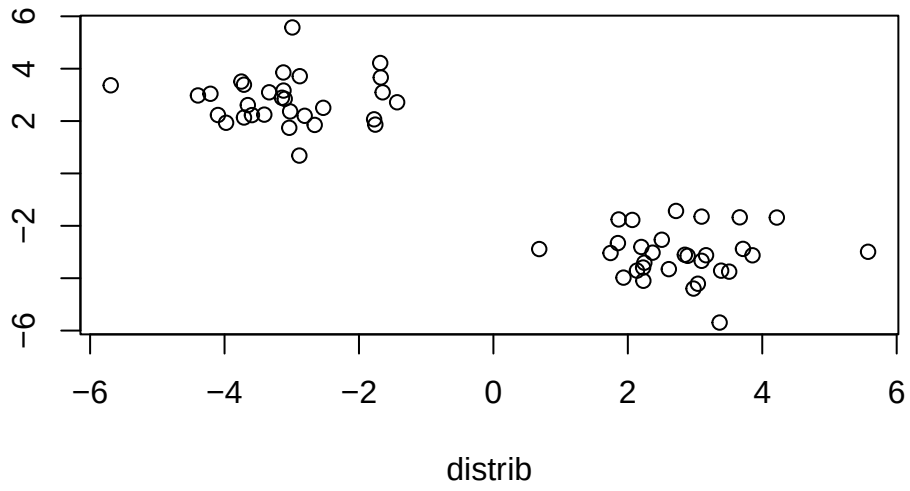


Let's get 30 points with a mean of 3 and another 30 points with a mean of -3. Combine two results into a vector.

```
distrib <- c(rnorm(30, mean=3),rnorm(30, mean=-3))
```

Combine the vector and its reverse version into a matrix

```
x <- cbind(distrib,rev(distrib))
plot(x)
```



k-means clustering

very popular clustering method that we can use with the `kmeans()` function in base R.

```
km <- kmeans(x,centers = 2)
km
```

```
K-means clustering with 2 clusters of sizes 30, 30

Cluster means:
    distrib
1  2.792763 -3.094210
2 -3.094210  2.792763
```

```
Clustering vector:
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

Within cluster sum of squares by cluster:
[1] 51.19157 51.19157
 (between_SS / total_SS =  91.0 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

Look at the available components

```
km$ifault
```

```
[1] 0
```

Q. How many points are in each clusters?

```
km$size
```

```
[1] 30 30
```

Q. What 'component' of your result object details? - cluster size - cluster assignment/membership - cluster center

```
km$size
```

```
[1] 30 30
```

```
km$cluster
```

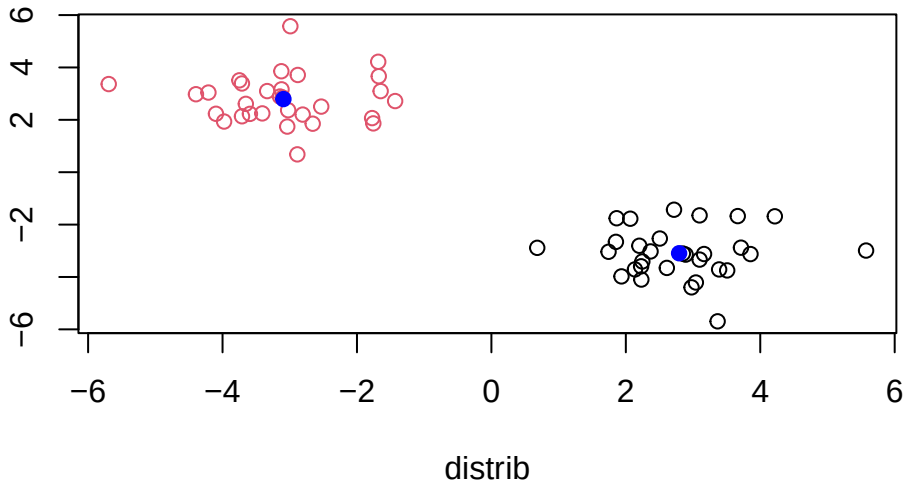
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
km$centers
```

```
    distrib
1  2.792763 -3.094210
2 -3.094210  2.792763
```

Q. Plot x colored by the kmeans cluster assignment and add cluster centers as blue points

```
plot(x, col=km$cluster)
points(km$centers, col = 'blue', pch=19)
```



Q Let's cluster into 3 groups or same x data and make a plot

```
km2 <- kmeans(x, centers=3)
km2
```

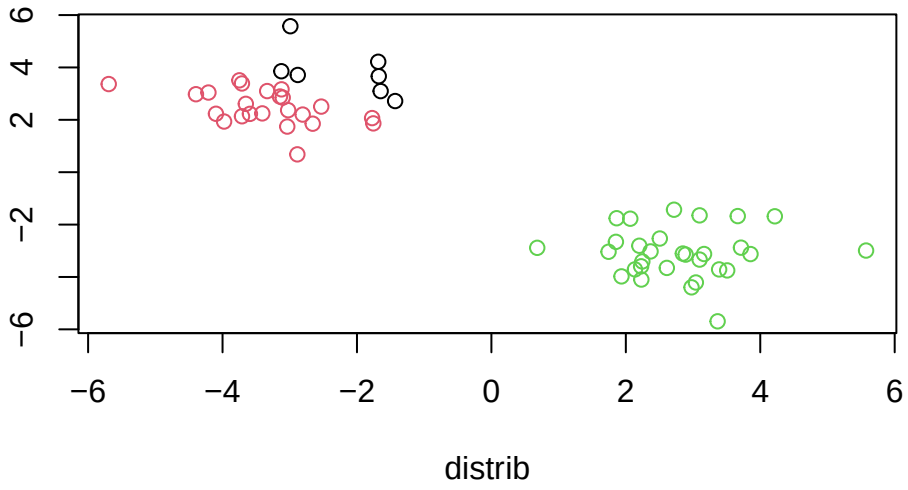
```
K-means clustering with 3 clusters of sizes 7, 23, 30

Cluster means:
    distrib
1 -2.205213  3.833940
2 -3.364775  2.475883
3  2.792763 -3.094210
```

```
Clustering vector:
 [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 1
[39] 2 1 2 2 2 2 2 1 1 2 2 2 2 1 2 1 2 2 2 2 1 2

Within cluster sum of squares by cluster:
[1]  8.390269 25.687514 51.191566
 (between_SS / total_SS =  92.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

```
km2$size
```

```
[1]  7 23 30
```

```
plot(x, col = km2$cluster)
```



How do we know what number of centers(clusters) are the best

```
km2$totss
```

```
[1] 1142.077
```

Hierarchicalclustering

We can use the `hclust()` function for hierarchical clustering unlike `kmeans()`, where we could just pass in our data as input, weneed to give `hclust()` a "distance matrix".

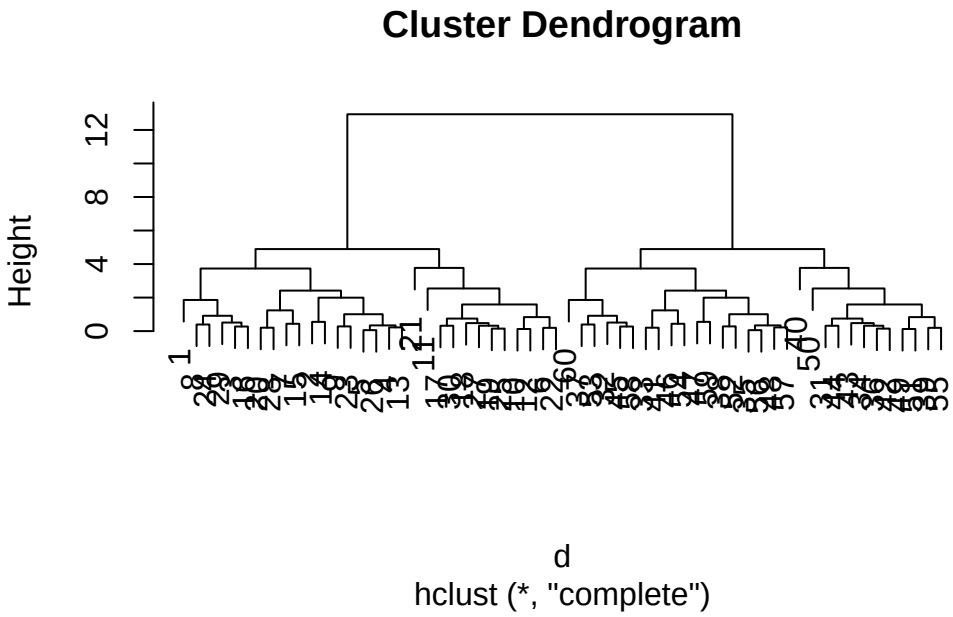
We will use a `dist()` function to start with.

```
d <-dist(x)
hc <- hclust(d)
hc
```

```
Call:
hclust(d = d)

Cluster method   : complete
Distance         : euclidean
Number of objects: 60
```

Generate dendrogram

```
plot(hc)
```

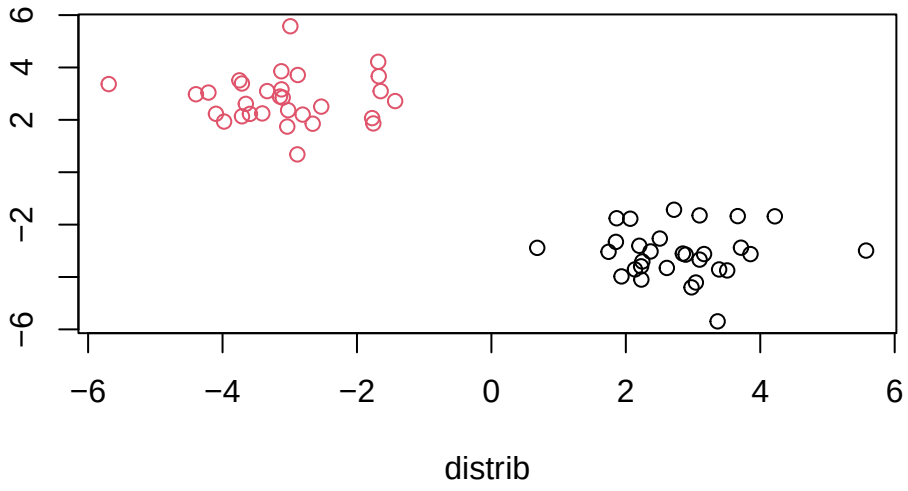


I can now "cut" my tree with the `cutree()` to yield a cluster membership vector

```
grps <- cutree(hc, h=8)
grps
```

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
plot(x, col=grps)
```



You can also tell `cutree()` to cut where it yields "k" groups

```
cutree(hc, k=2)
```

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Principal Component Analysis (PCA)

PCA of UK food data

```
uk <- read.csv("https://tinyurl.com/UK-foods", row.names=1)
uk
```

```
                England Wales Scotland N.Ireland
Cheese              105   103      103        66
Carcass_meat        245   227      242       267
Other_meat          685   803      750       586
Fish                147   160      122        93
Fats_and_oils       193   235      184       209
```

```
Sugars                156   175      147       139
Fresh_potatoes        720   874      566      1033
Fresh_Veg             253   265      171       143
Other_Veg             488   570      418       355
Processed_potatoes    198   203      220       187
Processed_Veg         360   365      337       334
Fresh_fruit          1102  1137      957       674
Cereals              1472  1582     1462      1494
Beverages              57    73       53        47
Soft_drinks          1374  1256     1572      1506
Alcoholic_drinks      375   475      458       135
Confectionery          54    64       62        41
```

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

```
dim(uk)
```

```
[1] 17  4
```

```
ncol(uk)
```

```
[1] 4
```

```
nrow(uk)
```

```
[1] 17
```

Preview the first 6 rows

```
head(uk)
```

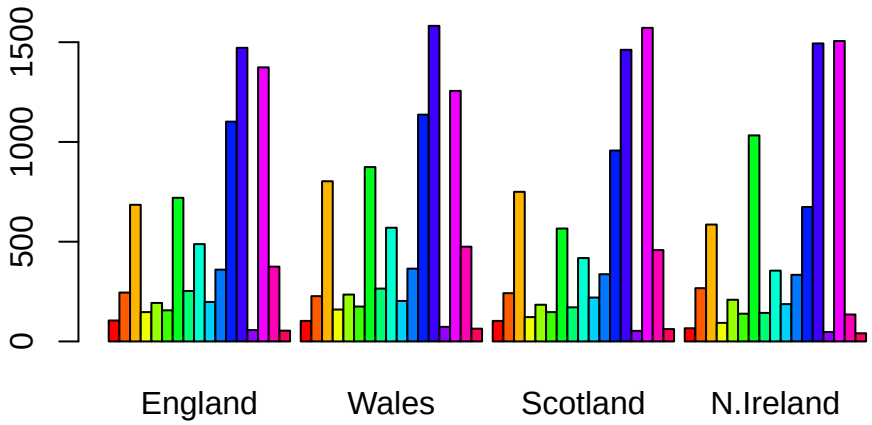
```
              England Wales Scotland N.Ireland
Cheese            105   103      103        66
Carcass_meat      245   227      242       267
Other_meat        685   803      750       586
Fish              147   160      122        93
Fats_and_oils     193   235      184       209
Sugars            156   175      147       139
```

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?

Using the row.names argument of read.csv(). Using - to delete rows will affect the original code and delete one row every time I run the code.

Generating a bar-plot for the data:

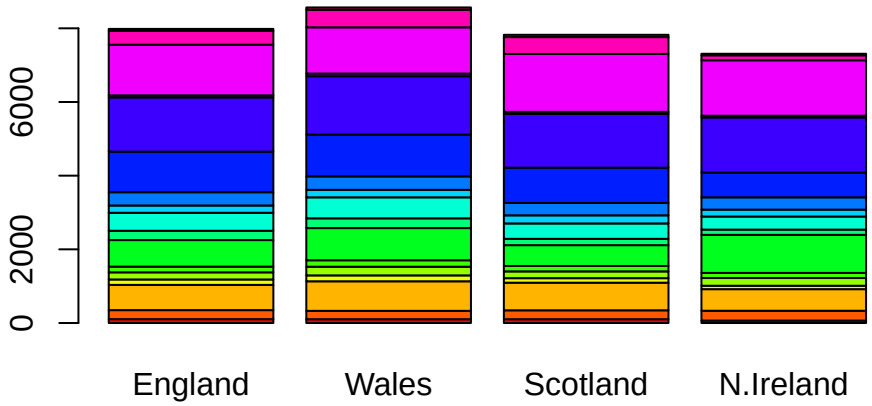
```
barplot(as.matrix(uk), beside=T, col=rainbow(nrow(uk)))
```



Q3: Changing what optional argument in the above barplot() function results in the following plot?

change beside from T to F

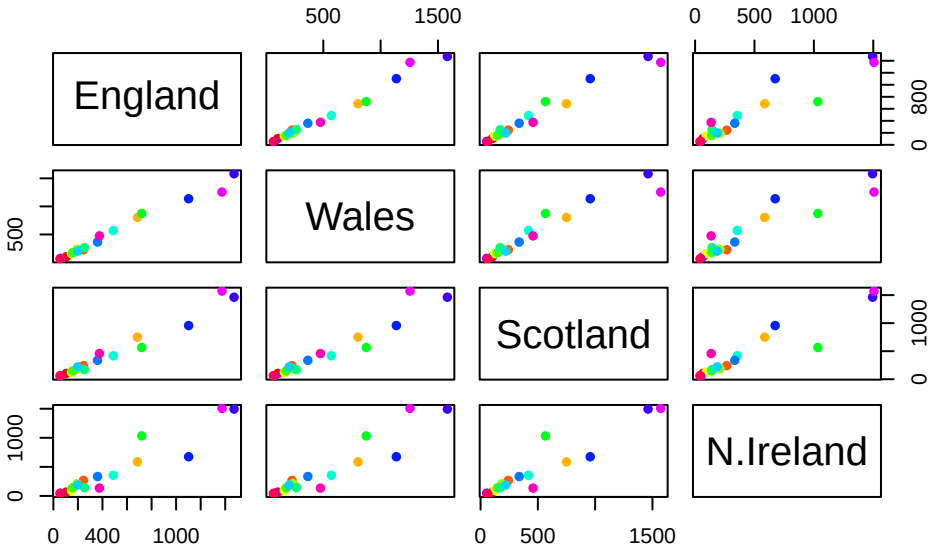
```
barplot(as.matrix(uk), beside=F, col=rainbow(nrow(uk)))
```



Q5: Generating all pairwise plots may help somewhat. Can you make sense of the following code and resulting figure? What does it mean if a given point lies on the diagonal for a given plot.

`pairs()` produce a matrix of scatterplot.

```
pairs(uk, col=rainbow(17), pch=16)
```



Points lying on the diagonal means that the consumption of foods for two countries are similar.

> Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set?

It has the most divergent values on the consumption of foods, as in the graphs it has least number of dots on the diagonal.

Use PCA

Use the `prcomp()` PCA function, it expect the transpose of our data. `t()` returns a transpose of data.frame (col becomes row, row becomes col).

```
pca <- prcomp( t(uk) )
summary(pca)
```

```
Importance of components:
                            PC1      PC2      PC3        PC4
Standard deviation     324.1502 212.7478 73.87622 4.189e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

Proportion of variance means how much dimentional variance is captured by the axis PC1.

Collectively PC1 and PC2 together capture 96% of the original 17 dimensional variance. Thus these first two new principal axis (PC1 and PC2) represent useful ways to view and further investigate our data set.

```
attributes(pca)
```

```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points. Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
pca$x #stands for the position of each country on all the generated axis
```

```
                PC1         PC2         PC3           PC4
England   -144.99315    2.532999 -105.768945  2.842865e-14
Wales     -240.52915  224.646925   56.475555  7.804382e-13
Scotland   -91.86934 -286.081786   44.415495 -9.614462e-13
N.Ireland  477.39164   58.901862    4.877895  1.448078e-13
```

```
pca$x[,1] # stands for the position of each country on PC1
```

```
   England      Wales   Scotland  N.Ireland
-144.99315 -240.52915  -91.86934  477.39164
```

```
pca$x[,2] # stands for the position of each country on PC2
```

```
    England       Wales    Scotland   N.Ireland
   2.532999  224.646925 -286.081786   58.901862
```

```
plot(pca$x[,1],pca$x[,2], col=c("red", "orange", "green", "blue"),
     pch=1,
     xlab = "PC1", ylab = "PC2")
text(pca$x[,1], pca$x[,2], colnames(uk),
     col=c("red", "orange", "green", "blue"))
```