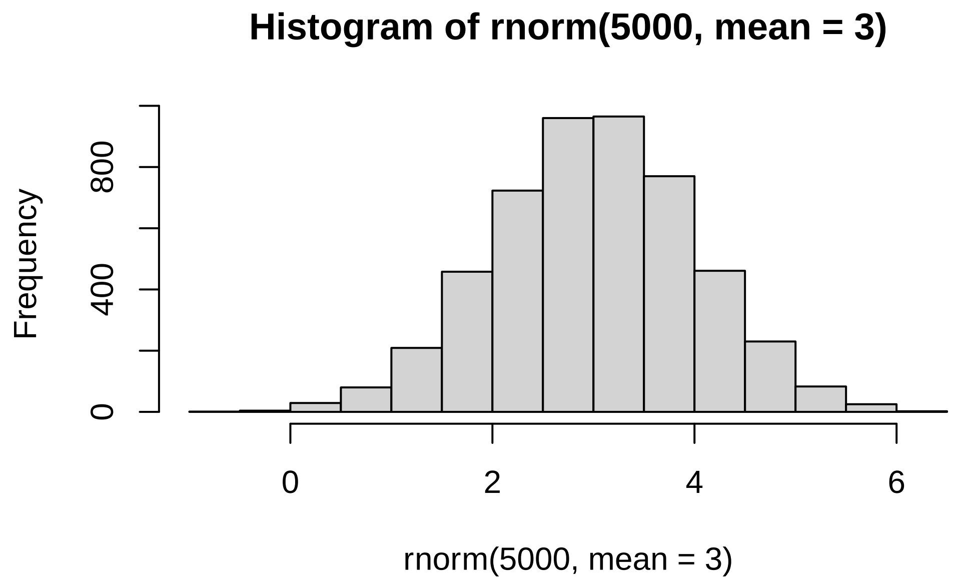
class07

Clustering

First let's make up some data to cluster so we can get a feel for these methods and how to work with them.

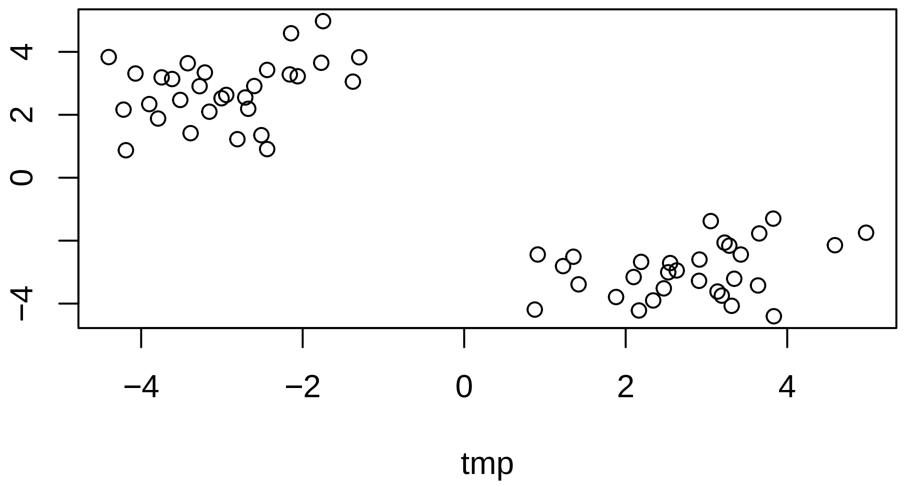
We can use `rnorm()` function to get random numbers from a normal distribution around a given `mean`.

```
hist( rnorm(5000, mean=3))
```



Let's get 30 points with a mean of 3. ANother 30 with a mean of -3. Then make a matrix whose plot has two clusters at two opposite corners in the Euclidean space.

```r
tmp <- c(rnorm(30, mean=3), rnorm(30, mean=-3))
x <- cbind(tmp, rev(tmp))
plot(x)
```



K-means clustering

Very popular clusterring method that we can use with the `kmeans()` function in base R

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

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

Cluster means:
        tmp
1 -2.953203  2.764833
2  2.764833 -2.953203

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

```
Within cluster sum of squares by cluster:
[1] 50.7812 50.7812
 (between_SS / total_SS =  90.6 %)

Available components:

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

```
km$size
```

```
[1] 30 30
```

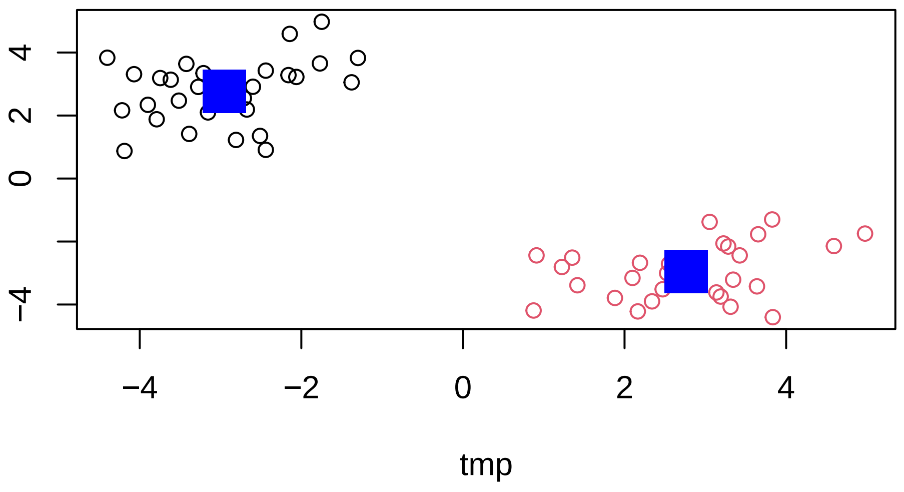
```
km$cluster
```

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

```
km$centers
```

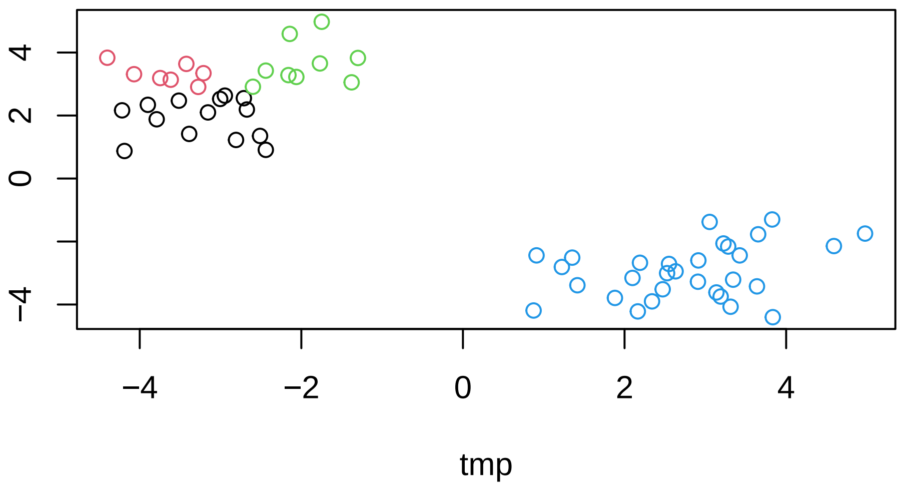
```
        tmp
1 -2.953203  2.764833
2  2.764833 -2.953203
```

```
plot(x, col=km$cluster)
points(km$centers, col="blue", pch=15, cex=3)
```



Q. Let’s cluster into 3 groups or same `x` data and make a plot

```
km <- kmeans(x, centers = 4)
plot(x, col=km$cluster)
```



Hierarchial Clustering

We can use the `hclust()` function for Hierarchial Clustering. Unlike `kmeans()`, where we could just pass in our data as input, we need to give `hclust()` a "distance matrix"

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

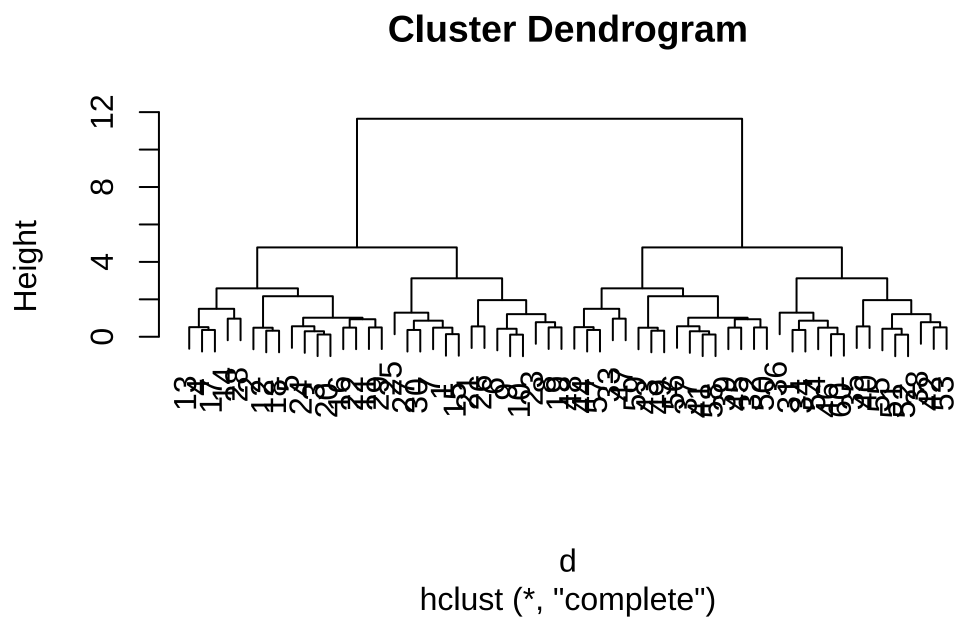
```
d <- dist(x)
hc <- hclust(d)
hc
```

```

Call:
hclust(d = d)

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

```
plot(hc)
```



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

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

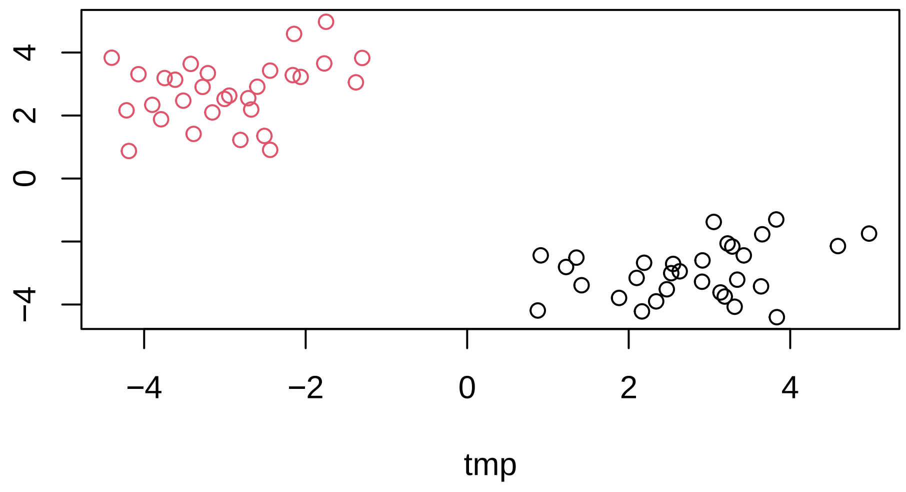
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

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

```r
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 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
```

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



Principal Component Analysis (PCA)

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

```
                    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. 17 rows, 4 columns

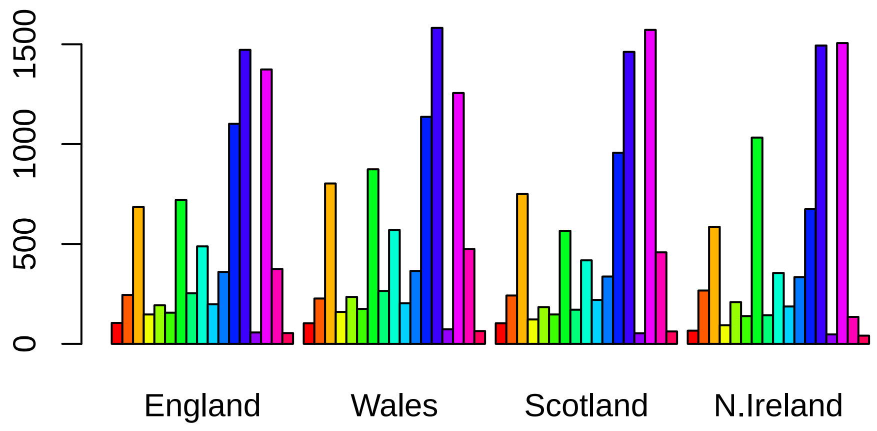
```
dim(x)
```

```
[1] 17  4
```

Q2. I prefer the row.names=1 approach because it is simpler and more convenient with larger data sets.

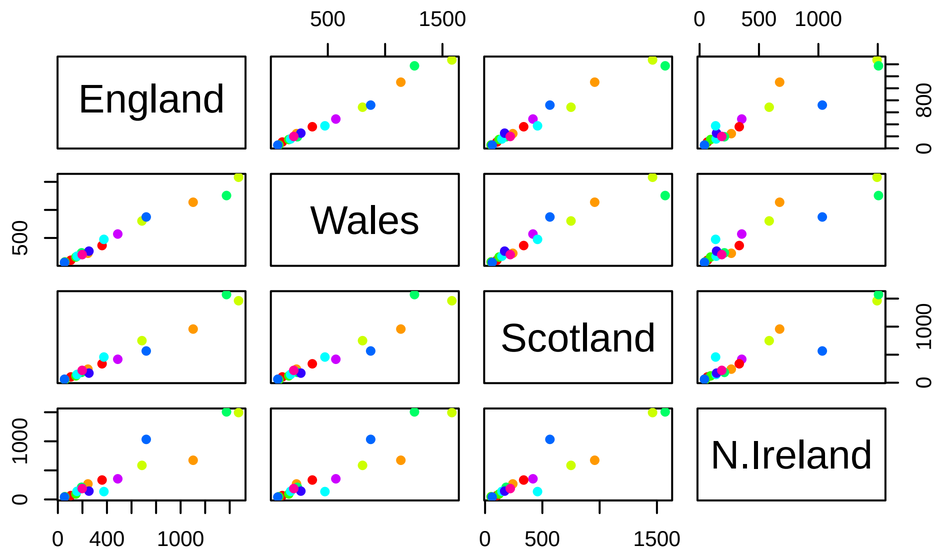
Exploratory analysis

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



Q3. Setting beside=FALSE in the barplot() function results in the other type of graph.

```
pairs(x, col=rainbow(10), pch=16)
```



> Q5. If the point lies on the diagonal for a given plot it means the points are equal.

The main PCA function in base R is called `prcomp()` it expects the transpose of our data

> Q6. Since the blue point is below the diagonal, Northern island has higher value than Scotland, England, and Wales.

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

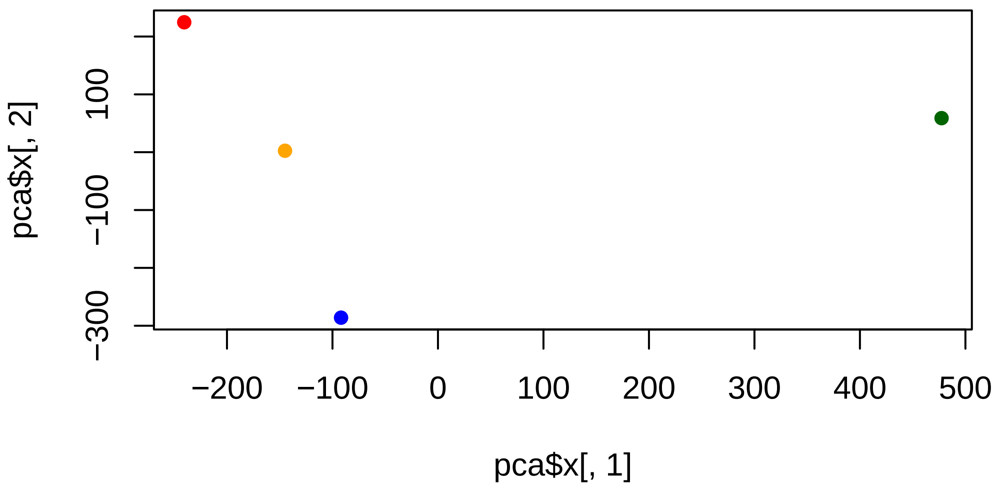
```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 4.189e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

```
pca$x
```

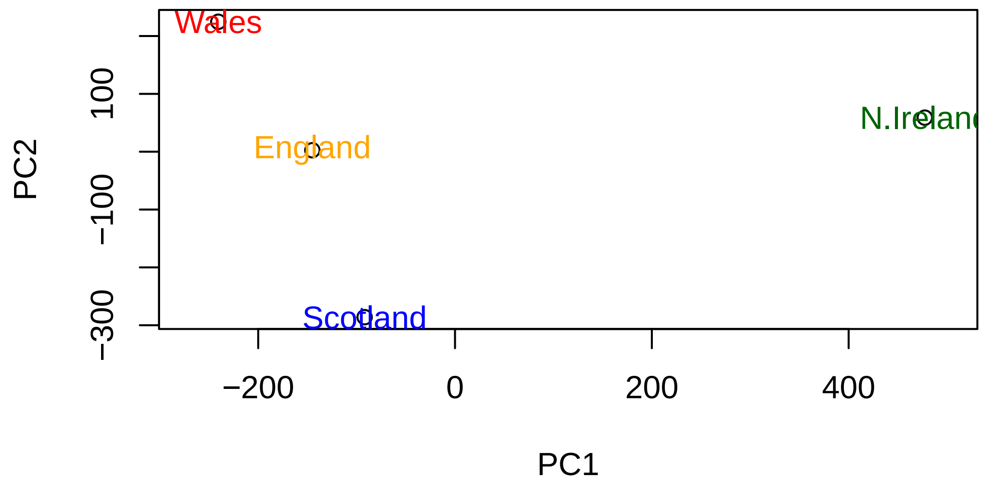
```
                PC1         PC2         PC3           PC4
England  -144.99315    2.532999 -105.768945  2.842865e-14
Wales    -240.52915  224.646925   56.475555  7.804382e-13
```

```
Scotland   -91.86934 -286.081786   44.415495 -9.614462e-13
N.Ireland  477.39164   58.901862    4.877895  1.448078e-13
```

```
plot(pca$x[,1], pca$x[,2],
     col=c("orange", "red", "blue", "darkgreen"), pch=16)
```



```
# Plot PC1 vs PC2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x), col=c("orange", "red", "blue", "darkgreen"))
```


Wales
England
N.Ireland
Scotland
PC2
100
−100
−300
−200
0
200
400
PC1