Class07

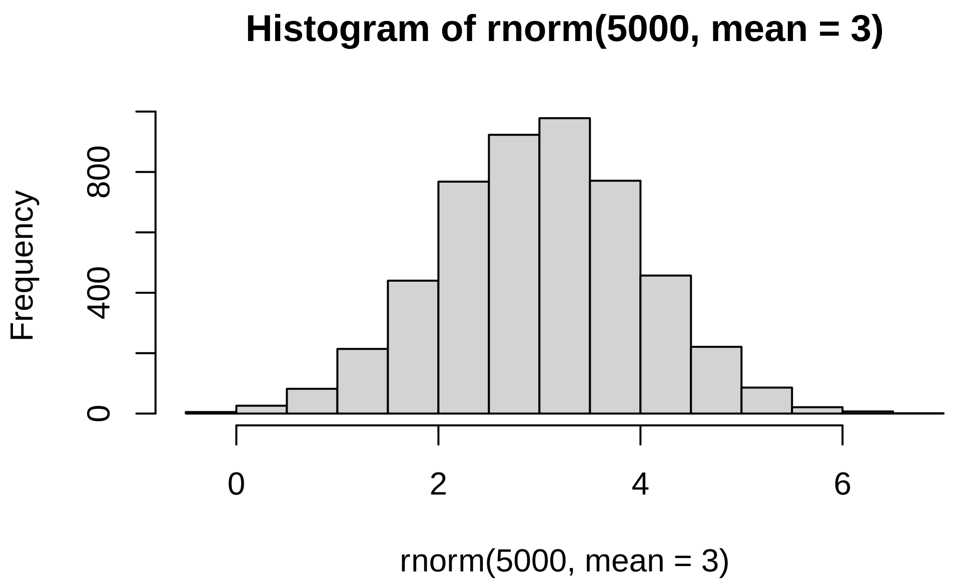
BIMM 143 Gen Dantay

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 the `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.

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

```
 [1]  2.7493809  3.4924039  4.9109461  5.8640985  2.0648637  3.3446873
 [7]  2.2573537  1.7228932  4.2471896  2.4962793  3.0360820  3.8239763
[13]  2.4513452  2.4123092  3.2624188  3.9835316  3.6290175  3.6242148
[19]  4.4521309  3.3642960  3.8736508  3.2889447  2.1725402  2.2395890
[25]  1.0578490  3.0938190  3.8672876  3.4085704  4.5605406  2.6495131
[31] -2.9682640 -3.9207955 -3.3851161 -2.9356157 -3.3874430 -2.9202339
[37] -1.7954534 -1.9453838 -4.0382002 -1.4958362 -1.5926376 -2.9481173
[43] -4.4220549 -1.3654968 -1.7058994 -1.4950154 -2.2596222 -3.0276633
[49] -0.3429288 -3.1251945 -2.2047894 -5.4783034 -2.5240095 -2.7435051
[55] -4.8136898 -4.3121857 -3.8246415 -3.1462086 -4.1453907 -3.9160331
```

Trying `rev()`:

```
rev(c(1, 2, 3, 4, 5))
```

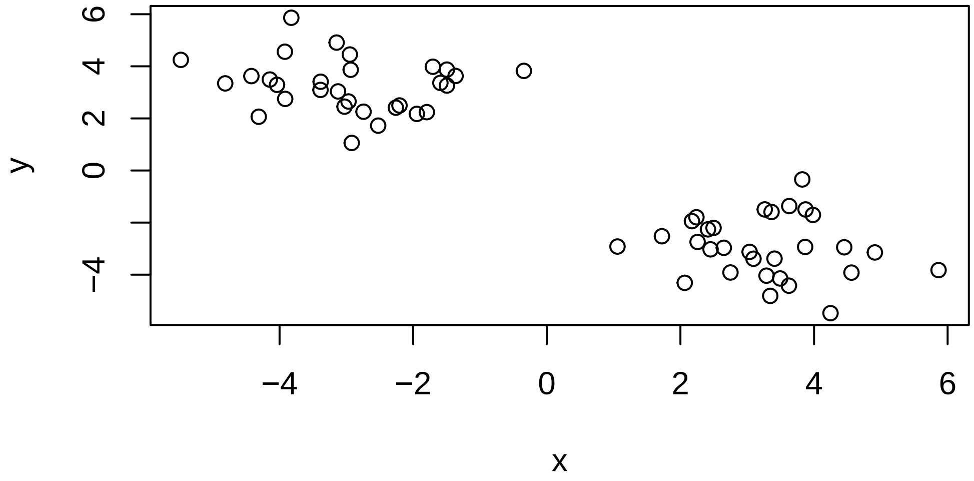
```
[1] 5 4 3 2 1
```

```
cbind(c(1, 2, 3, 4, 5), rev(c(1,2,3,4,5)))
```

```
     [,1] [,2]
[1,]    1    5
[2,]    2    4
[3,]    3    3
[4,]    4    2
[5,]    5    1
```

Putting two together(code above the one above this):

```
x <- cbind(x=tmp, y=rev(tmp))
plot(x)
```



K-means clustering.

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

```r
# 2 clusters:
# Cluster vector says which cluster they belong to.
# Available components is the stuff needed to work with this answer.
km <- kmeans(x, centers = 2)
km
```

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

Cluster means:
          x         y
1  3.246724 -2.939524
2 -2.939524  3.246724

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 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

Within cluster sum of squares by cluster:
```

```
[1] 70.19966 70.19966
 (between_SS / total_SS =  89.1 %)

Available components:

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

```
# Questions:
#Cluster size:
km$size
```

```
[1] 30 30
```

```
#Cluster assignment/membership:
km$cluster
```

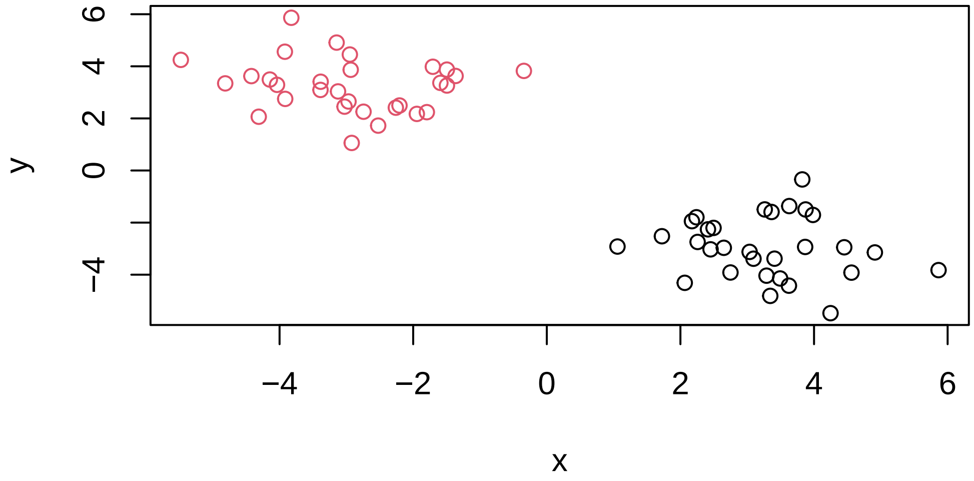
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
km$centers
```

```
          x         y
1  3.246724 -2.939524
2 -2.939524  3.246724
```

Q. Plot x colored by the kmeans cluster assignment and…

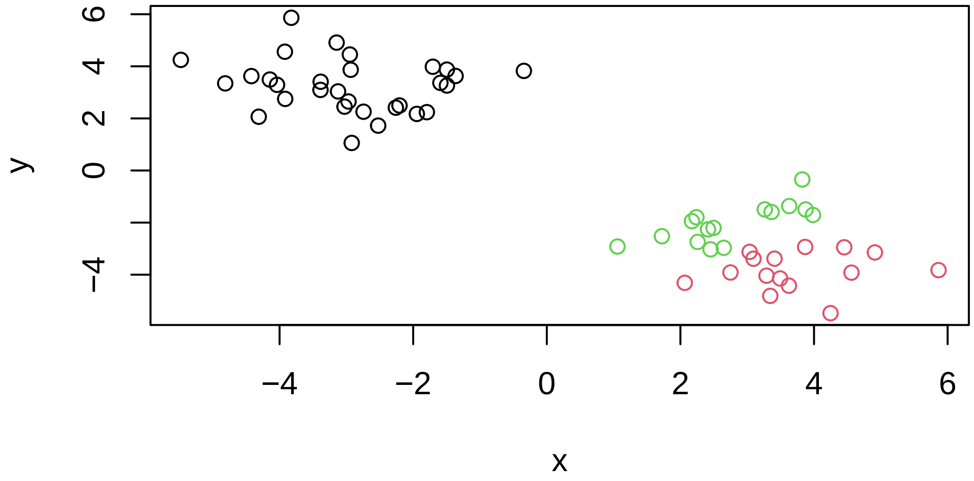
```
mycols <- c(1, 2)
plot(x, col=km$cluster)
```



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

```
help(kmeans)
```

```
km<- kmeans(x, centers=3)
plot(x, col=km$cluster)
```



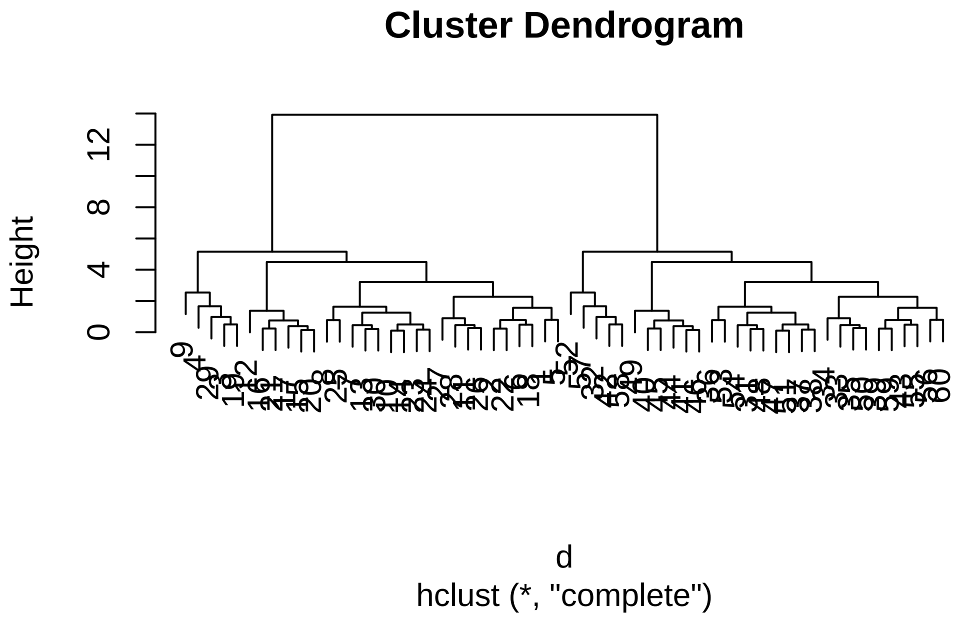
Heirarchical Clustering

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

we will use the `dist()` function to start with:

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

```
plot(hc)
```



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

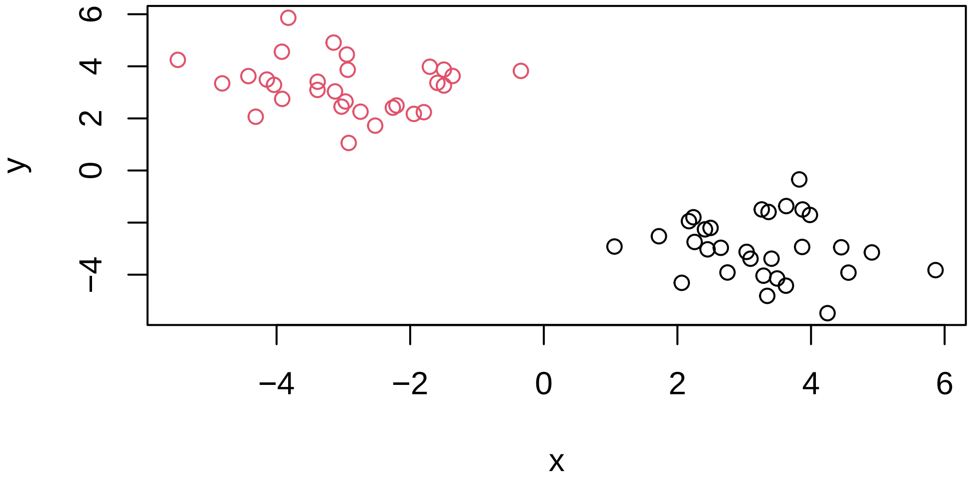
```
cutree(hc, h=8)
```

```
 [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.

```
grps<- cutree(hc, k=2)
```

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

```
dim(x)
```

```
[1] 17  4
```

There are 17 rows and 5 columns. We can use dim(x) to get both the ouputs of row and column, or we can use `nrow(x)` or `ncol(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?

the solutions are below, but I much prefer putting in `row.names=1` more because it is much more simple and more robust than the others.

```
rownames(x) <- x[,1]
x <- x[,-1]
head(x)
```

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

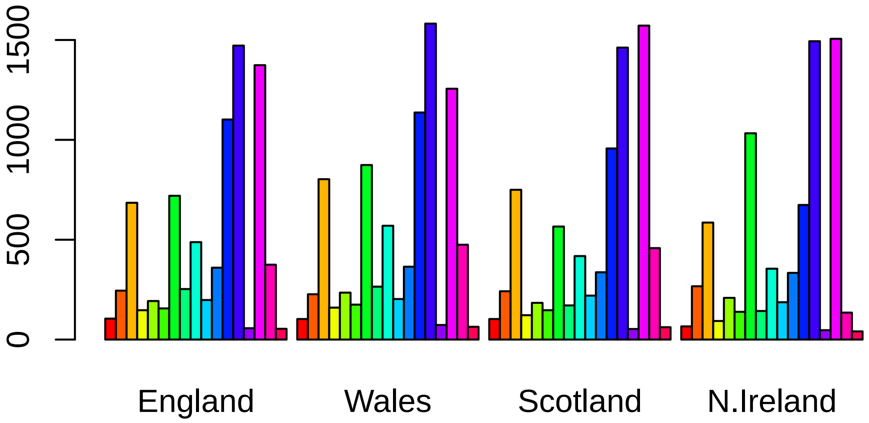
```
dim(x)
```

```
[1] 17  3
```

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

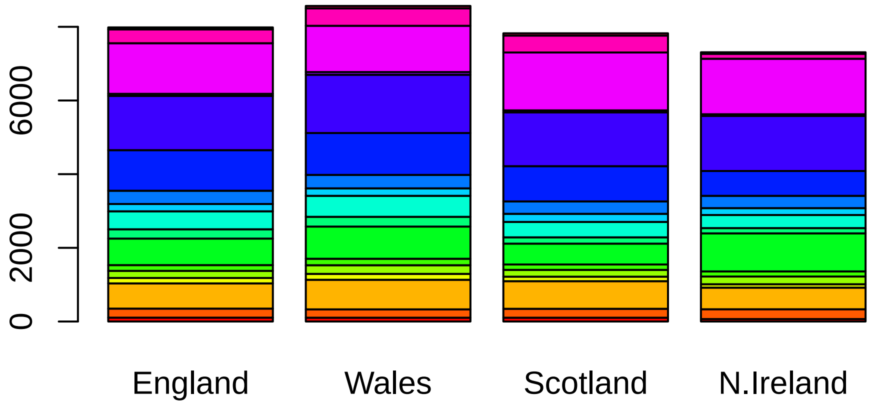
```r
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

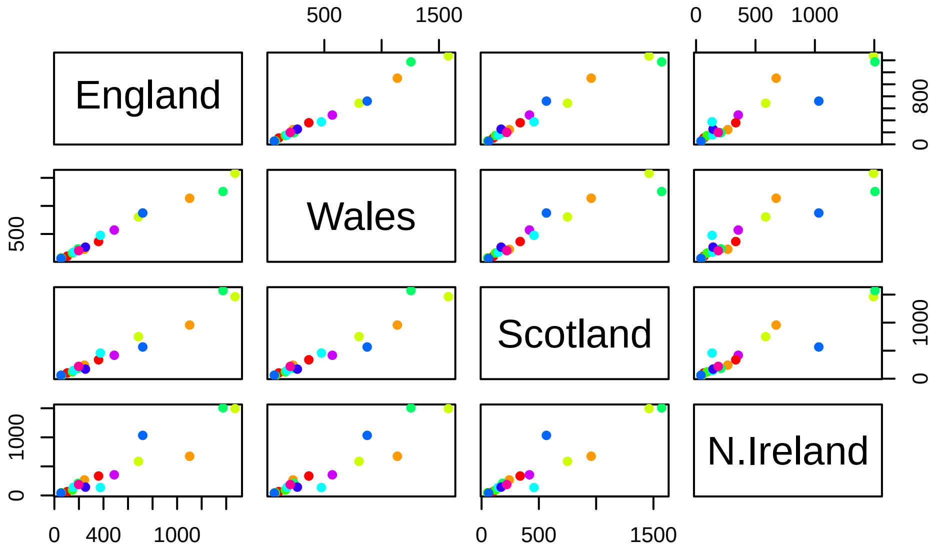
Changing beside from `T` to `F` will create the plot below.

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



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



It is called a pair plot, because it is all countries paired together. From the plot, we see that N.Ireland, compared to England, is very different in terms of one of the food categories consumed. If a point lands on a diagonal that would mean that the specific food consumption is similar between two countries.

```
?prcomp()
```

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

```
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
```

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

```
pca$x
```

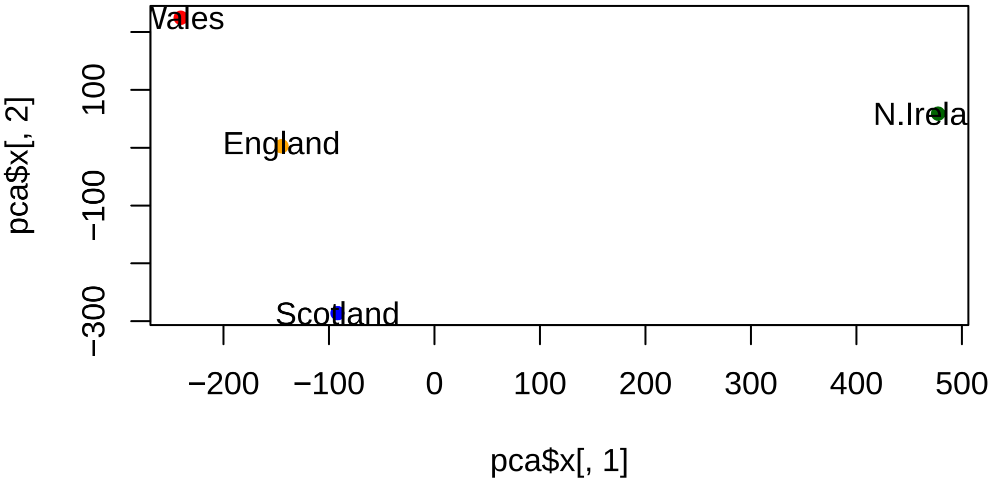
```
                PC1        PC2         PC3           PC4
England   -144.99315   2.532999 -105.768945  2.842865e-14
Wales     -240.52915 224.646925   56.475555  7.804382e-13
Scotland   -91.86934 -286.081786   44.415495 -9.614462e-13
N.Ireland  477.39164  58.901862    4.877895  1.448078e-13
```

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

while the beginning graphs show almost no obvious deviations, from the pair plot, we see that whatever the blue point is different seems more like an outlier from the other countries. Due to this, we probably end up with the results in the graph below:

> Q7 and Q8 are the graphs below:

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


Vales
England
N.Irela
Scotland
100
−100
−300
−200
−100
0
100
200
300
400
500
pca$x[, 1]
pca$x[, 2]