Class 7: Clustering and PCA

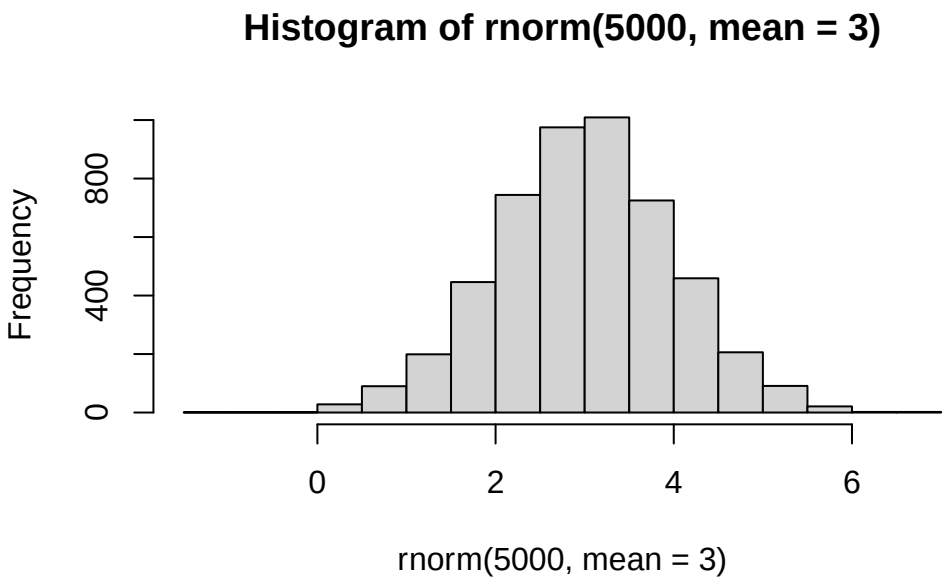
Zichen "Cardiff" Jiang

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.

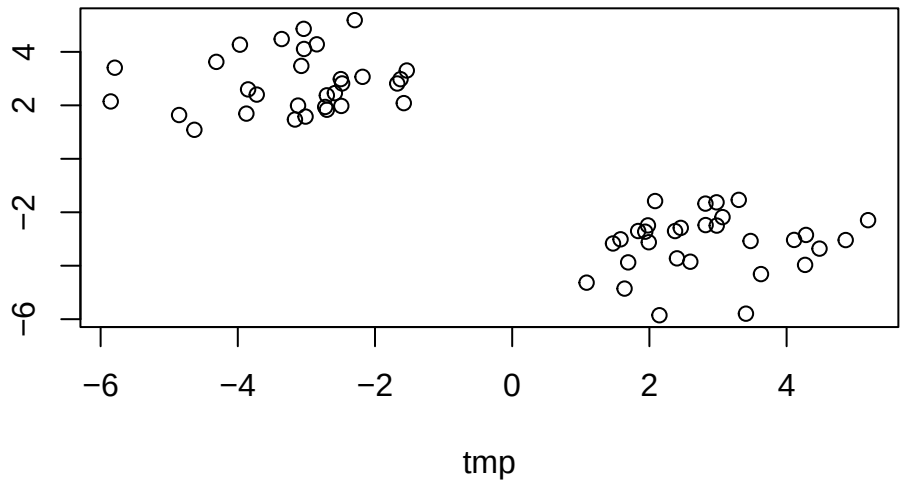
We can use `rnorm()` function to get random numbers from a normal distribution around a given `mean`.

```
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 clustering method, especially for big data set, 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  2.830775 -3.153614
2 -3.153614  2.830775

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 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] 70.31949 70.31949
 (between_SS / total_SS =  88.4 %)

Available components:

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

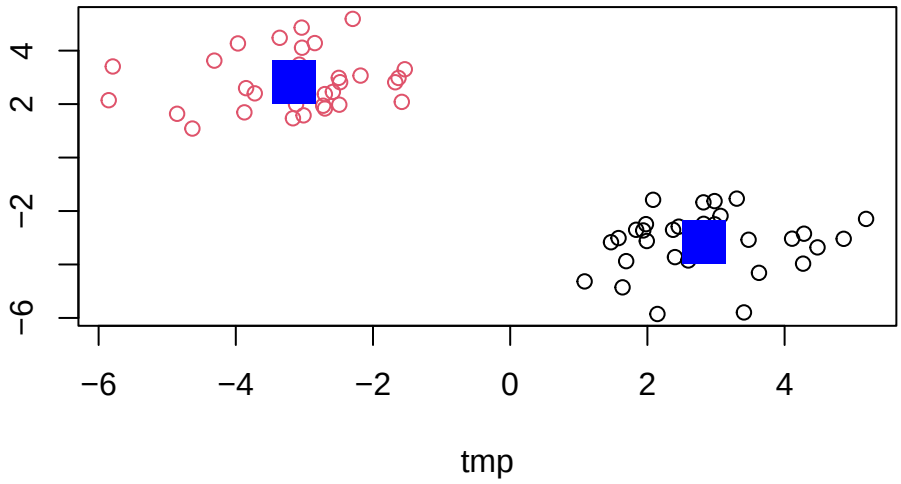
There're 30 points in each cluster

Cluster size is km$size

Cluster membership is km$cluster

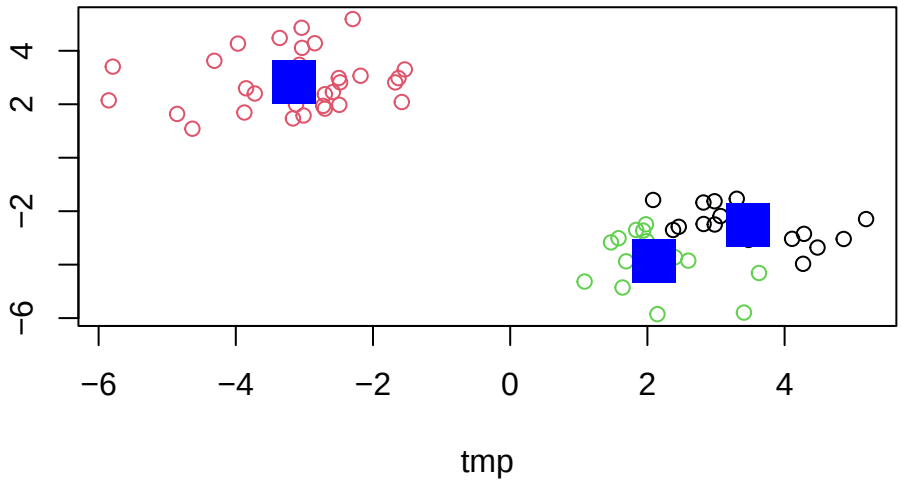
Cluster centers is km$centers

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



Q Let's cluster into 3 groups on the same `x`

```r
km_3 <- kmeans(x, centers = 3)
plot(x, col=km_3$cluster)
points(km_3$centers, col="blue", pch=15, cex=3)
```

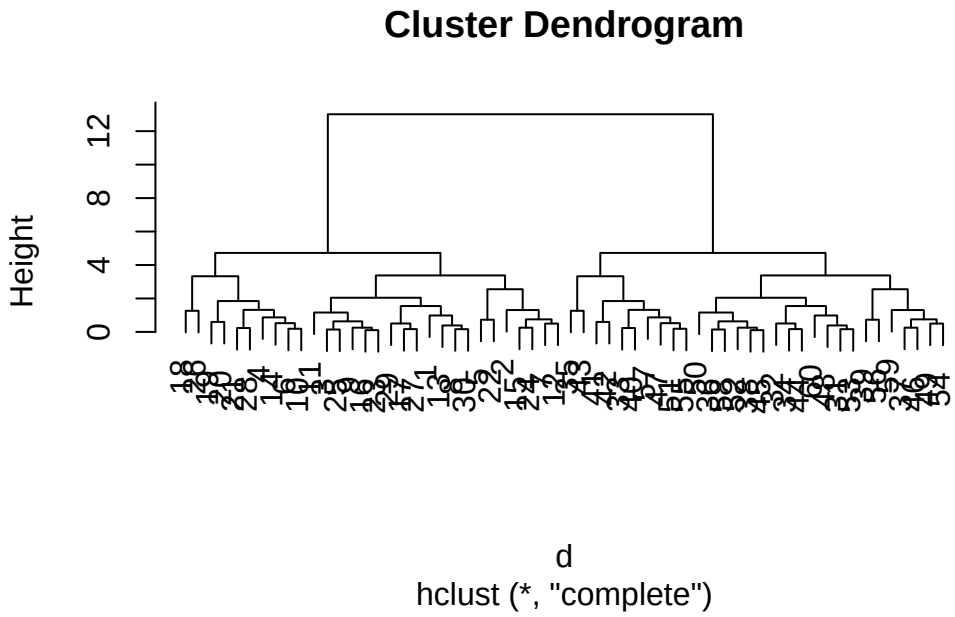


Hierarchical clustering

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

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

```r
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=10)
```

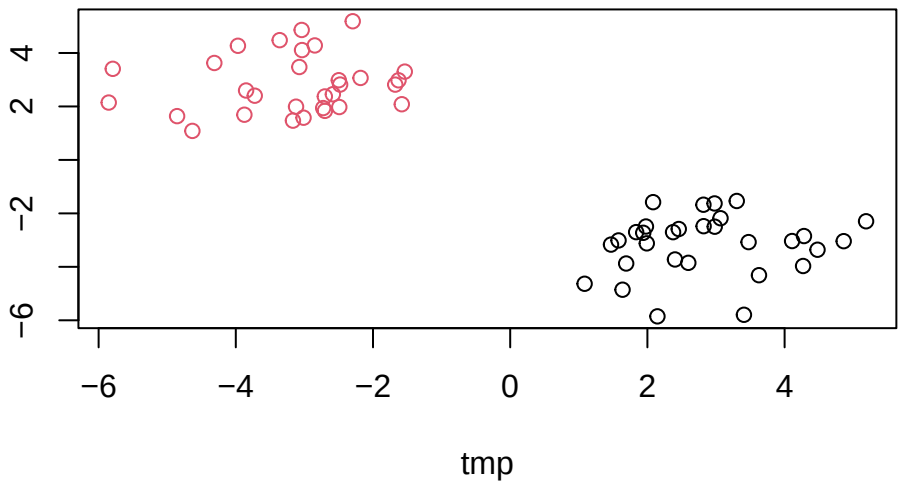
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

You can also tell `cutree()` to cut where it yield "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
```

```
plot(x, col=cutree(hc, k=2))
```



PCA

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

```
[1] 17  5
```

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), 17 rows, 5 columns

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

```
dim(x)
```

```
[1] 17  4
```

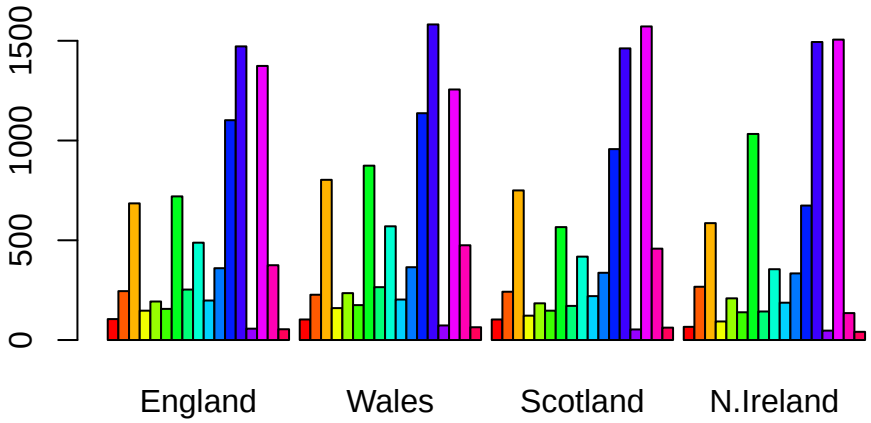
```
x <- read.csv(url, row.names=1)
head(x)
```

```
              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
```

> **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?

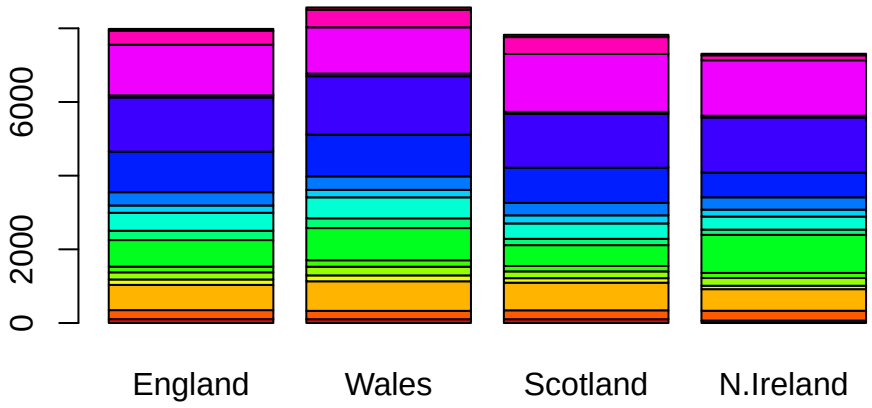
I prefer the second approach because it's less code. The second approach is more robust because you don't have to modify the x variable

```
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?

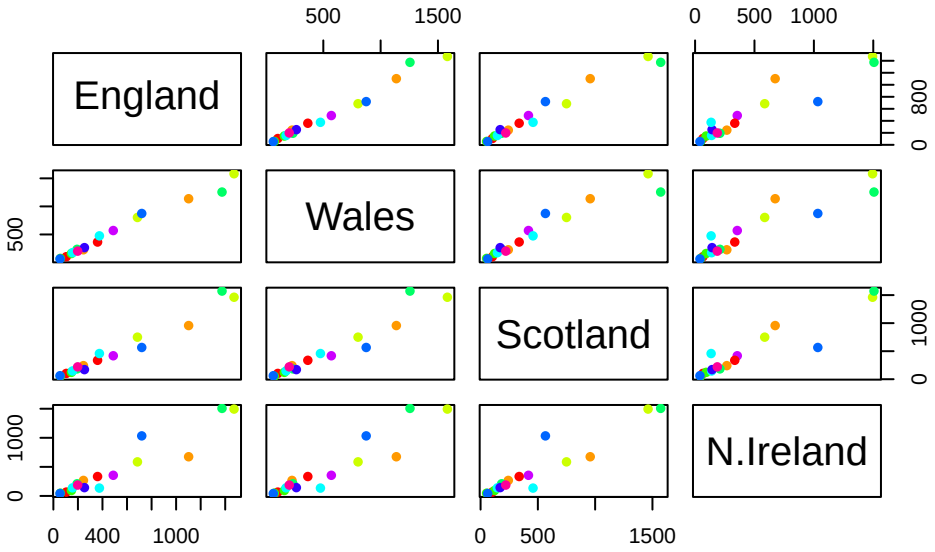
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



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?

Diagonal means two countries consume a food around the same level.

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



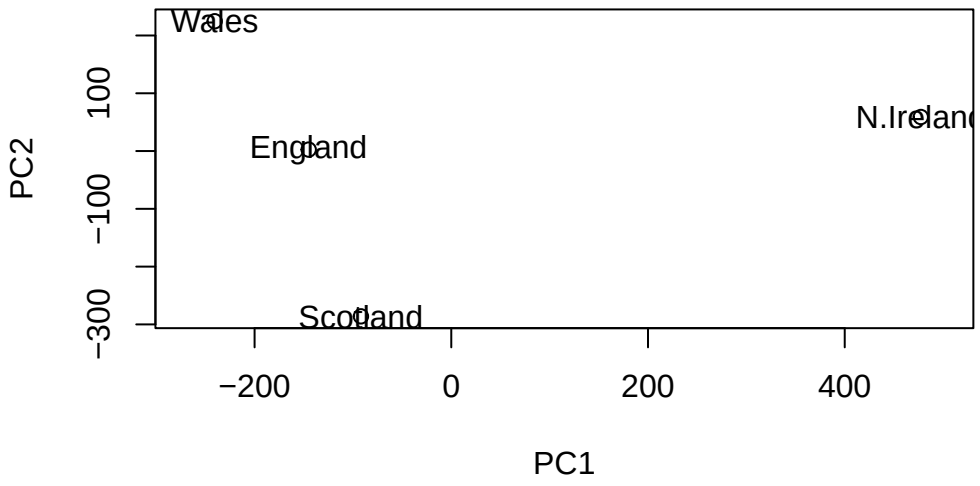
Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set?

For the orange dot, England consumes more. For the blue dot, England consumes less.

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

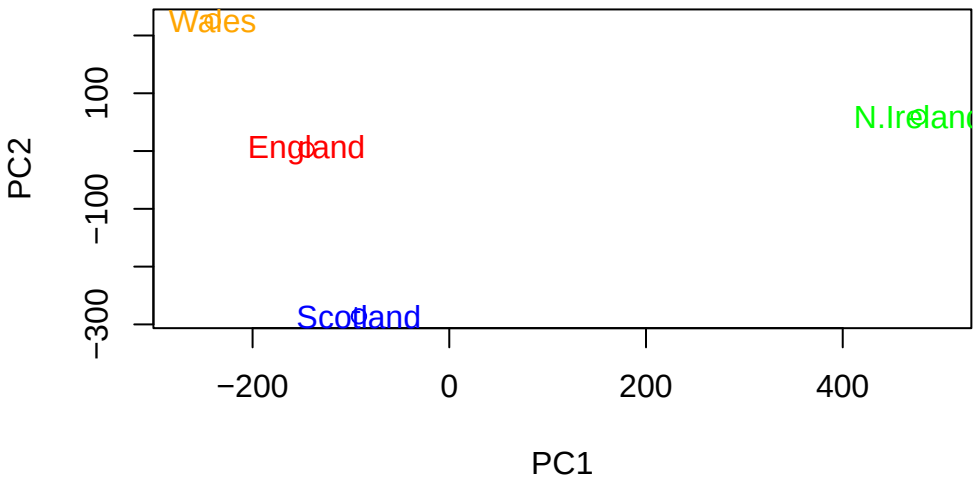
```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 5.552e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

```
# 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))
```



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.

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


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