Lab 7 - Tue 1/28

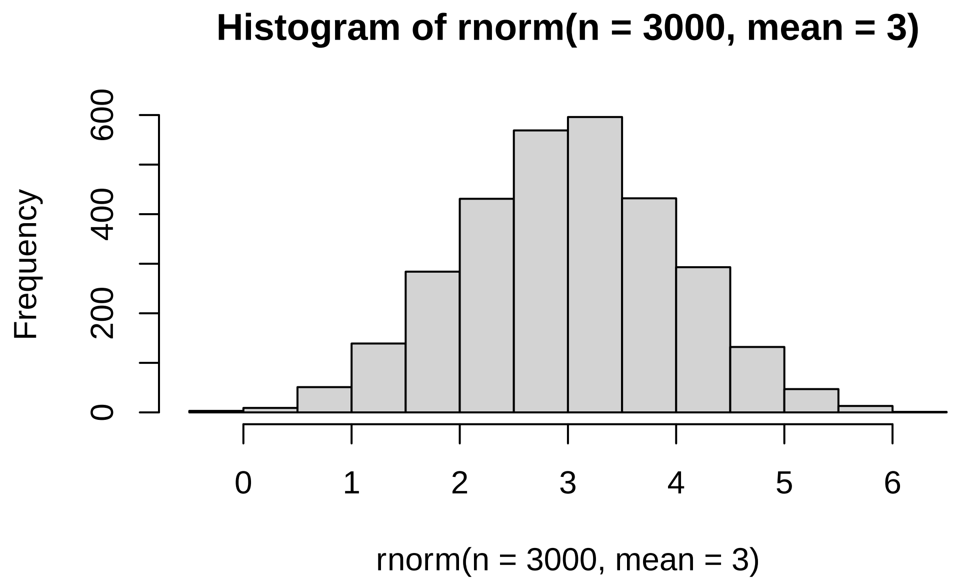
Jessica Le (PID: A17321021)

Today we will explore unsupervised machine learning methods which include clustering annd dimensionality reduction methods.

Let's start by making up some data (where we know there are clear groups) that we can use to test out different clustering methods.

We can use `rnorm()` function to help us:

```r
hist(rnorm(n=3000, mean=3))
```

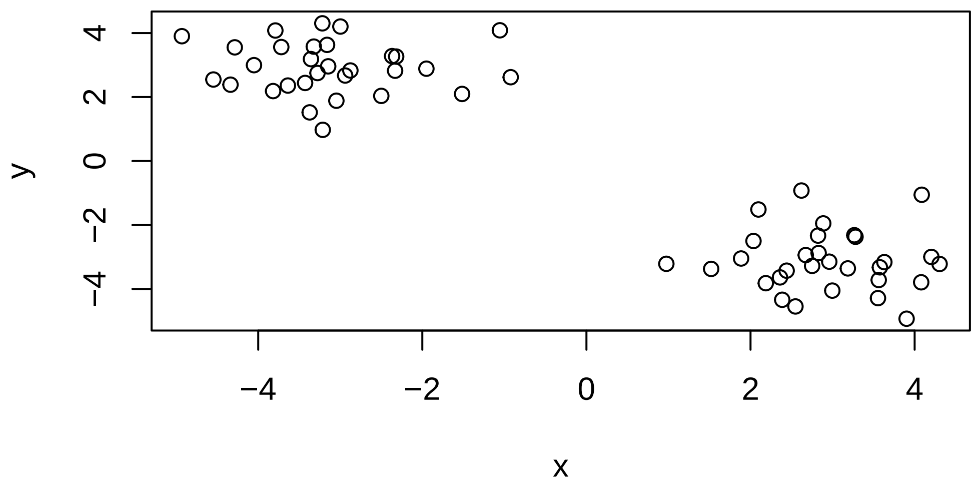


Make data with two "clusters"

```r
x <- c( rnorm(30, mean=-3),
        rnorm(30, mean=+3) )
z <- cbind(x=x, y=rev(x))
head(z)
```

```
              x        y
[1,] -2.8767481 2.830009
[2,] -3.7909754 4.080100
[3,] -3.6385218 2.358911
[4,] -0.9231745 2.620554
[5,] -3.3730264 1.520274
[6,] -3.7190381 3.562471
```

```r
plot(z)
```



How big is ‘z’

```r
nrow(z)
```

```
[1] 60
```

```r
ncol(z)
```

```
[1] 2
```

##K-means clustering

The main function in “base” R for K-means clustering is called `kmeans()`

```r
k <- kmeans(z, centers=2)
k
```

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

Cluster means:
          x         y
1 -3.114809  2.920081
2  2.920081 -3.114809

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] 45.54605 45.54605
 (between_SS / total_SS =  92.3 %)

Available components:

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

```r
attributes(k)
```

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

$class
[1] "kmeans"
```

> Q. How many points lie in each cluster?

```r
k$size
```

```
[1] 30 30
```

> Q. What component of our results tells us about the cluster membership (i.e which point lies in which cluster)?

```r
k$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
```

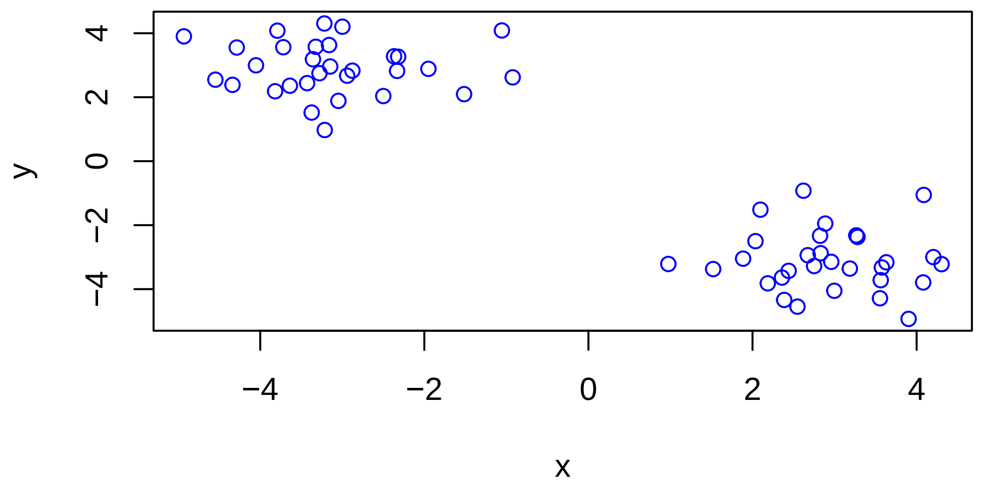
> Q. Center of each cluster?

```r
k$centers
```

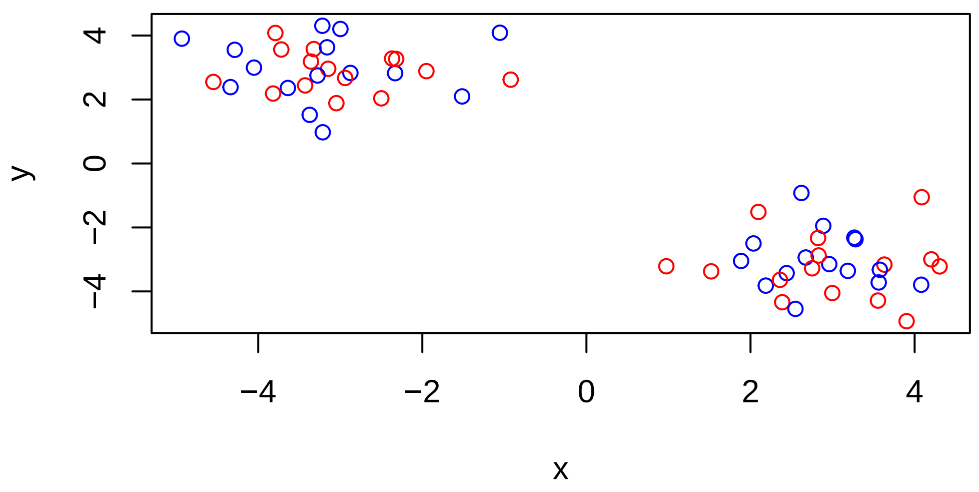
```
          x         y
1 -3.114809  2.920081
2  2.920081 -3.114809
```

> Q. Put these results information together and make a "base R" plot of the clustering results with the cluster center points.

```r
plot(z, col="blue")
```

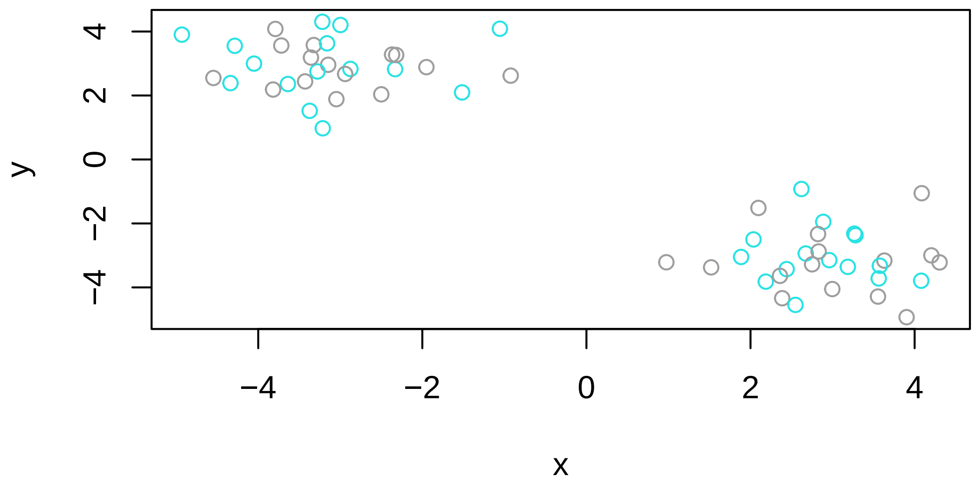


```
plot(z, col=c("blue", "red"))
```



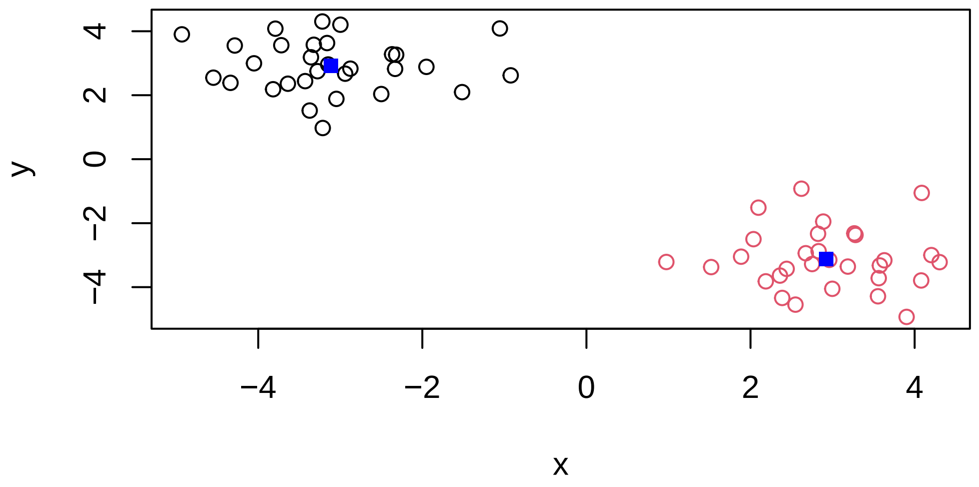
You can color by number.

```
plot(z, col=c(5,8))
```



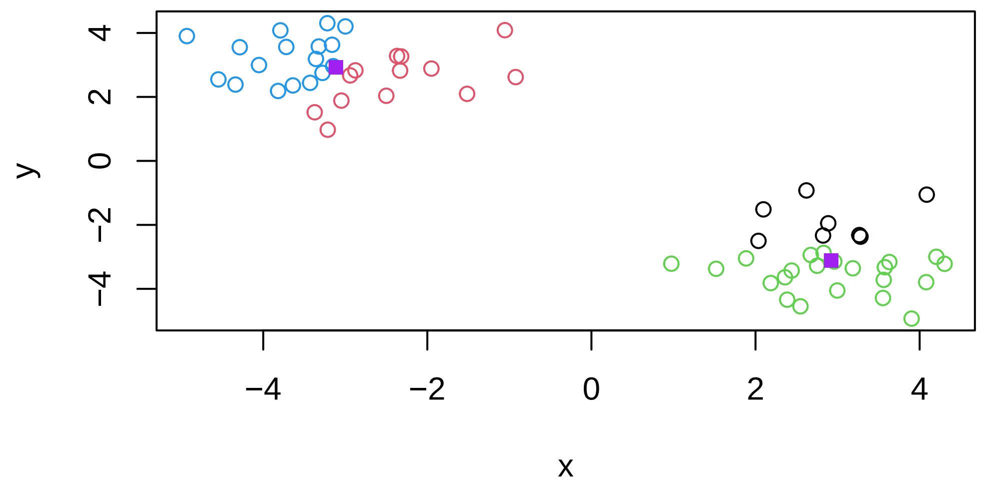
Plot colored by cluster membership:

```
plot(z, col=k$cluster)
points(k$centers, col="blue", pch=15)
```



Q. Run kmeans on our input **z** and define 4 clusters making the same result visualization plot as above (plot of z colored by cluster membership).

```
k4 <- kmeans(z, centers=4)
plot(z, col=k4$cluster)
points(k$centers, col="purple", pch=15)
```



Hierarchical Clustering

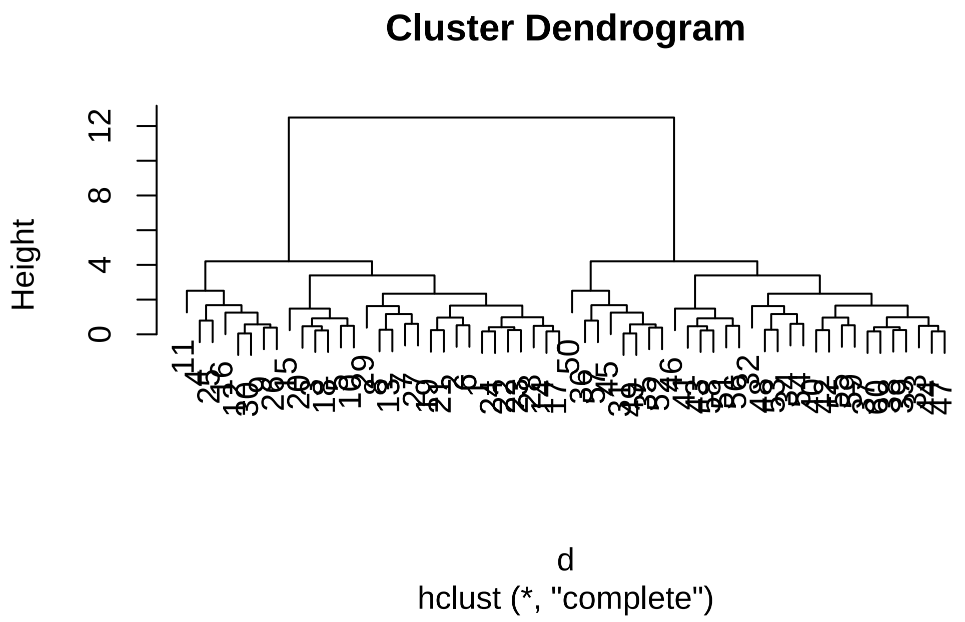
The main function in base R for this called `hclust()` it will take an input of a distance matrix (key point is that you can't just give your "raw" data as input - you have to first calculate a distance matrix from your data).

```
d <- dist(z)
hc <- hclust(d=d)
hc
```

```
Call:
hclust(d = d)
```

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

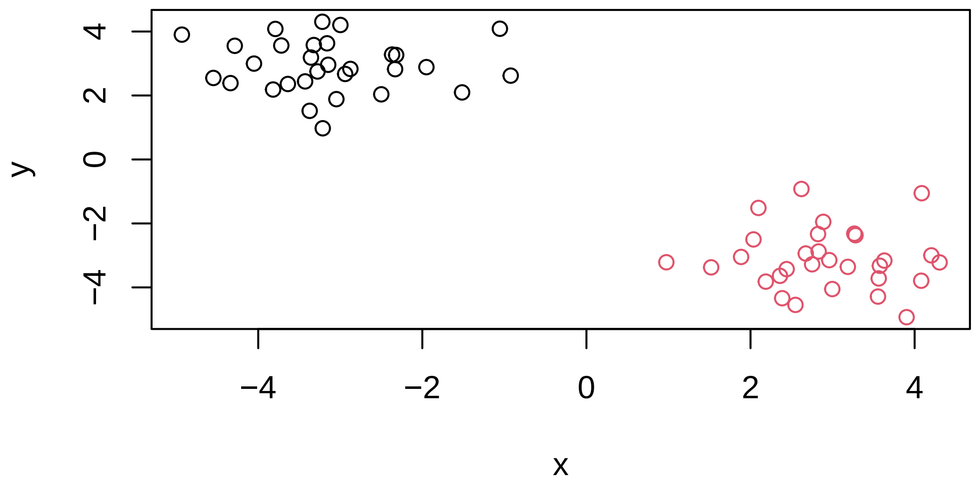
```r
plot(hc)
```



Once I inspect the "tree"/dendrogram, I can "cut" the tree to yield my groupings or clusters. The function to do this is called `cutree()`.

```r
grps <- cutree(hc, h=10)
```

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



Hands on with Principal Component Analysis (PCA)

Let's examine some silly 17-dimensional data detailing food consumption in the UK (England, Scotland, Wales, and Northern Ireland). Are these countries eating habits different or similar and if so how?

Data Import

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

> Q1. How many rows and columns are in your new data set framed x? What R functions could you se to answer this question?

```r
nrow(x)
```

```
[1] 17
```

```r
ncol(x)
```

```
[1] 4
```

```r
dim(x)
```

```
[1] 17  4
```

Checking the Data

To preview the first 6 rows.

```r
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
```

Here, the row names are set as the first column of the `x` data frame rather than set as proper row names. So, we need to remove the first column.

```r
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
```

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

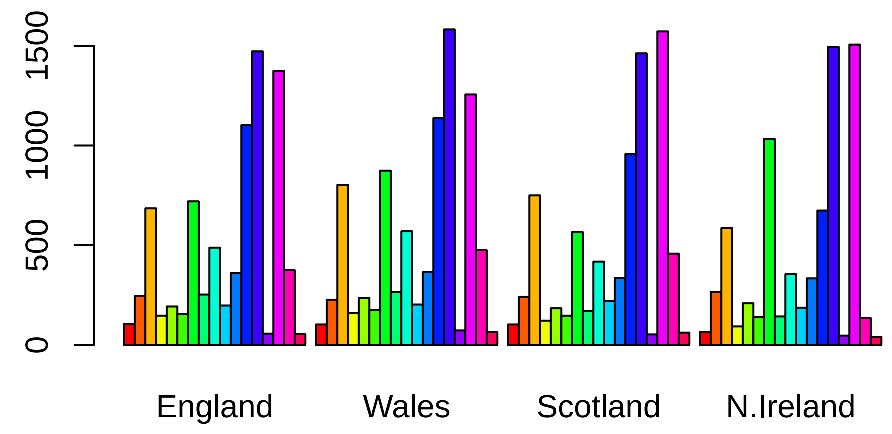
```
               England Wales Scotland N.Ireland
Cheese             105   103      103        66
Carcass_meat       245   227      242       267
Other_meat         685   803      750       586
Fish               147   160      122        93
Fats_and_oils      193   235      184       209
Sugars  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

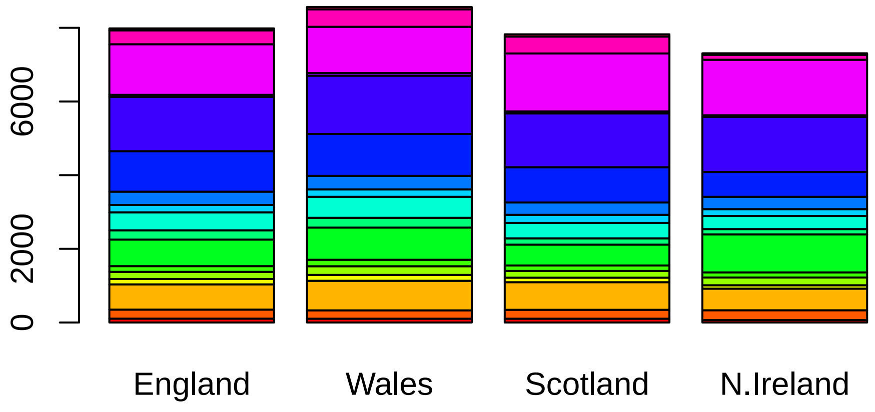
The second method is more preferable because if the first method with `x <- x[,-1]` is used multiple times it will keep removing a column of data each time and we will end up with an empty data set. The second method also requires less code which makes it easier to use.

Spotting major differences and trends

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



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

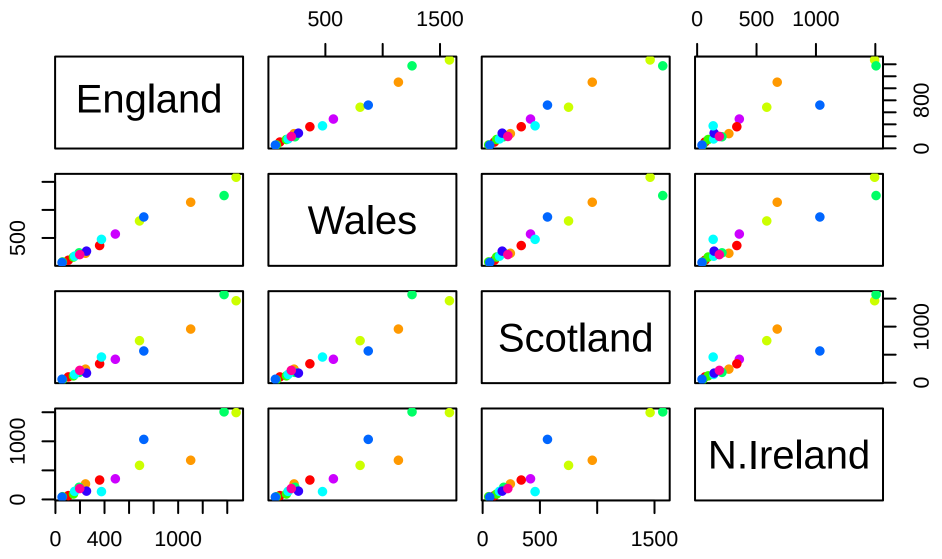


> Q3. Changing what optional argument in the first barplot() function resulted in the second plot?

Changing the `beside` argument from True to False changed the barplot from bars adjacent to each other to bars stacked upon each other in the second plot.

> Q. 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?

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



The plot pairs each country together to compare their eating habits. If a point lies on the diagonal when comparing two countries, they have similar eating habits for the specified food group. If a point does not lie on the diagonal for a give plot it means that the corresponding country does not have similar eating habits to the comparison country for the food group of interest.

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

Northern Ireland has one food group indicated by a dark blue point in the plots in which their food consumption of this group greatly varies from the other three countries.

Looking at these types of "pairwise plots" can be helpful but it does not scale well and kind of sucks. There must be a better way…

PCA to the rescue!

The main function for PCA in base R is called `prcomp()`. This function wants the transpose of our input data - i.e. the important foods in as columns and the countries as rows.

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

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 2.921e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

For cumulative proportion, a plot of PC1 and PC2 will capture 96.50% of the data.

Let's see what is in our PCA result object `pca`

```r
attributes(pca)
```

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

$class
[1] "prcomp"
```

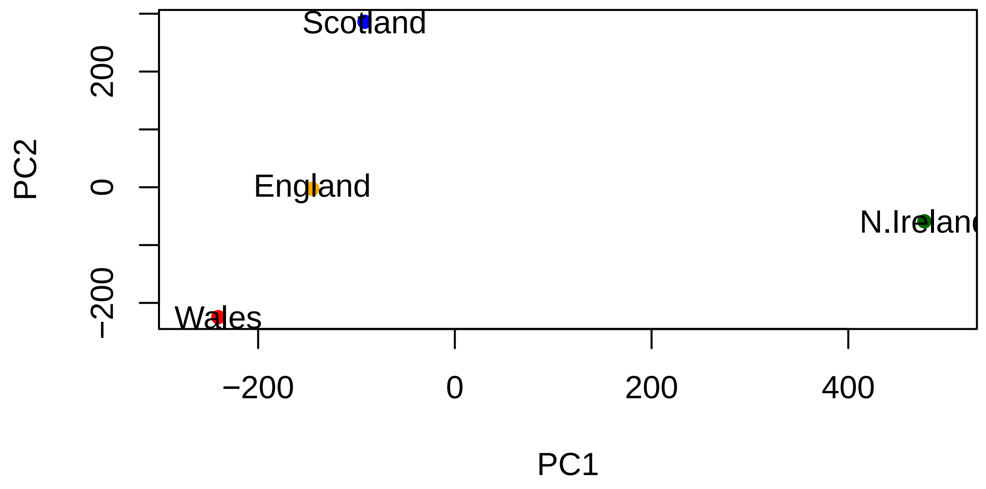
The `pca$x` result object is where we will focus first as this details how the countries are related to each other in terms of our new "axis" (aka. "PCs", "eigenvectors", etc.).

```r
head(pca$x)
```

```
                 PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -9.152022e-15
Wales     -240.52915 -224.646925 -56.475555  5.560040e-13
Scotland   -91.86934  286.081786 -44.415495 -6.638419e-13
N.Ireland  477.39164  -58.901862  -4.877895  1.329771e-13
```

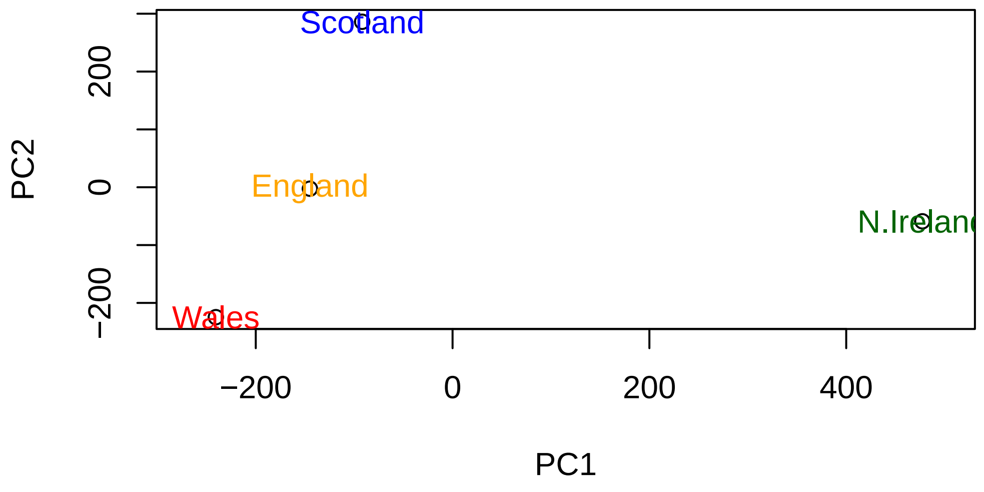
> Q7. Complete the code below to generate a plot of PC1 vs. PC2. The second line adds text labels over the data points.

```r
plot (pca$x[,1], pca$x[,2], pch=16,
      col=c("orange", "red", "blue", "darkgreen"),
      xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x))
```



Q8. Customize the plot so that the colors of the country names match the colors in our UK and Ireland map and table at the start of this document.

```
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"))
```



We can look at the so-called PC "loadings" result object to see how the original foods contribute to our new PC (i.e. how the original variables contribute to our new better variables). The Eigenvectors contain information about the contributions of each principal component to the total variance of the coordiantes.

To calculate how much variation in the original data each PC accounts for, use the square of `pca$sdev`.

```
v <- round (pca$sdev^2/sum(pca$sdev^2)*100)
v
```

```
[1] 67 29  4  0
```

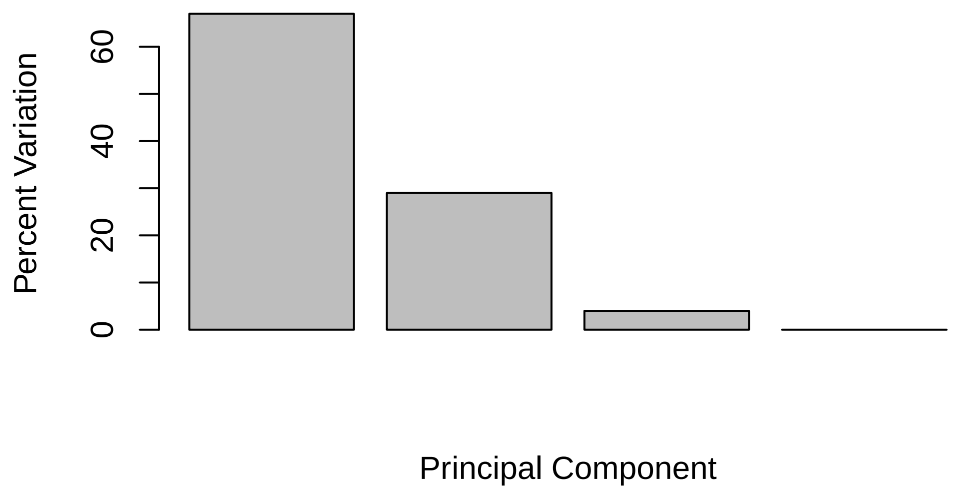
The variance information is also provided in the second row in the summary of the dataset.

```
z <- summary(pca)
z$importance
```

```
                             PC1       PC2      PC3          PC4
Standard deviation     324.15019 212.74780 73.87622 2.921348e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

This information can be summarized in a plot of of the variance (eigenvalues) with respect to the principal component number (eigenvector number), which is given below.

```r
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```



Rotation helps to improve the interpretibility of the principal components by simplifying the structures of the loadings. It slightly moves the PCA axes relative to the original variable axes, while still maintaining the orthogonality (or "uncorrelatedness") of the components.

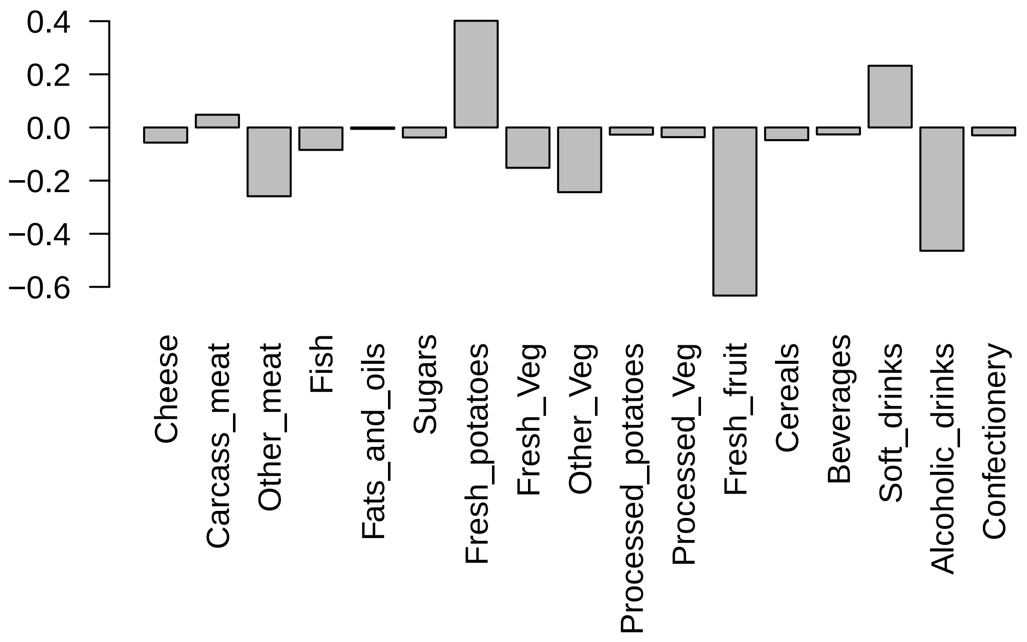
```r
pca$rotation
```

```
                            PC1          PC2         PC3          PC4
Cheese             -0.056955380  0.016012850  0.02394295 -0.409382587
Carcass_meat        0.047927628  0.013915823  0.06367111  0.729481922
Other_meat         -0.258916658 -0.015331138 -0.55384854  0.331001134
Fish               -0.084414983 -0.050754947  0.03906481  0.022375878
Fats_and_oils      -0.005193623 -0.095388656 -0.12522257  0.034512161
Sugars             -0.037620983 -0.043021699 -0.03605745  0.024943337
Fresh_potatoes      0.401402060 -0.715017078 -0.20668248  0.021396007
Fresh_Veg          -0.151849942 -0.144900268  0.21382237  0.001606882
Other_Veg          -0.243593729 -0.225450923 -0.05332841  0.031153231
Processed_potatoes -0.026886233  0.042850761 -0.07364902 -0.017379680
```

```
Processed_Veg       -0.036488269 -0.045451802  0.05289191  0.021250980
Fresh_fruit         -0.632640898 -0.177740743  0.40012865  0.227657348
Cereals             -0.047702858 -0.212599678 -0.35884921  0.100043319
Beverages           -0.026187756 -0.030560542 -0.04135860 -0.018382072
Soft_drinks          0.232244140  0.555124311 -0.16942648  0.222319484
Alcoholic_drinks    -0.463968168  0.113536523 -0.49858320 -0.273126013
Confectionery       -0.029650201  0.005949921 -0.05232164  0.001890737
```

Lets focus on PC1 as it accounts for $>$ 90% of variance

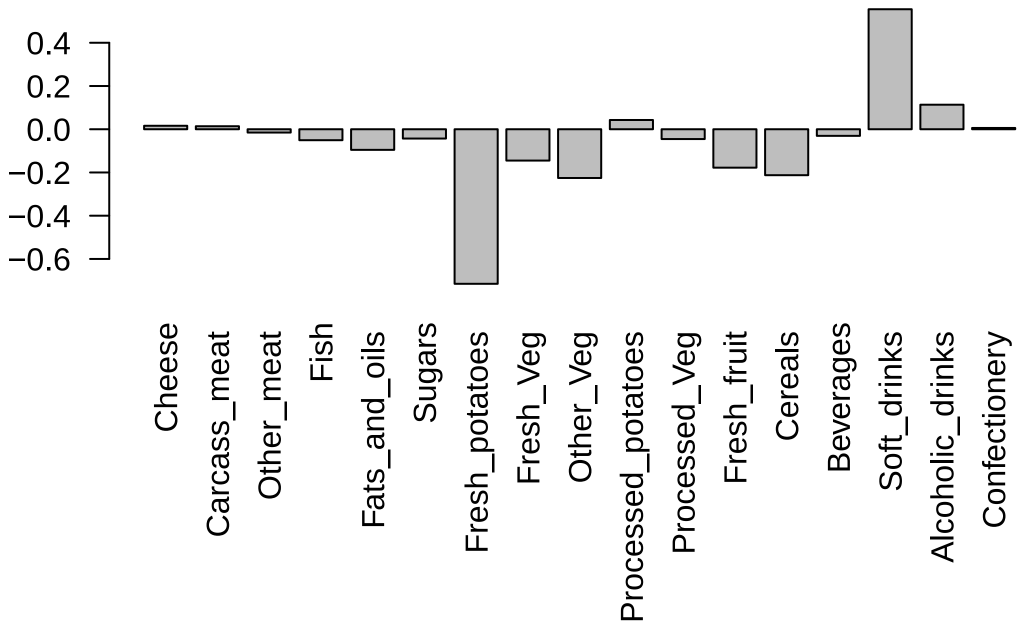
```r
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



The plot shows that food groups with the largest positive loading scores, which are fresh potatoes and soft drinks, “push” N. Ireland to the right side of the plot. In other words, N. Ireland greatly varies from the other countries in PC1 due to their consumption habits of fresh potatoes and soft drinks. On the other hand, foods with the highest negative scores contribute to the countries that are mainly present on the left side of the plot. So, the consumption of fresh fruit and alcoholic drinks contribute to the similar patterns observed for Wales, England, and Scotland.

Q9. Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about?

```r
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```



This plot for PCA2, which considers the second most variation in the entire dataset, shows that soft drinks is the largest food group with a high positive loading score. This means that soft drinks is the main contributor that pushes Scotland upwards, and therefore leading to its distinction, in the plot comparing the food consumption of the four countries. On the other hand, fresh potatoes has the highest negative loading score which means that it is the food group that contributes most to the patterns between the countries towards the bottom of the graph, which are England, N. Ireland, and Wales.