class07

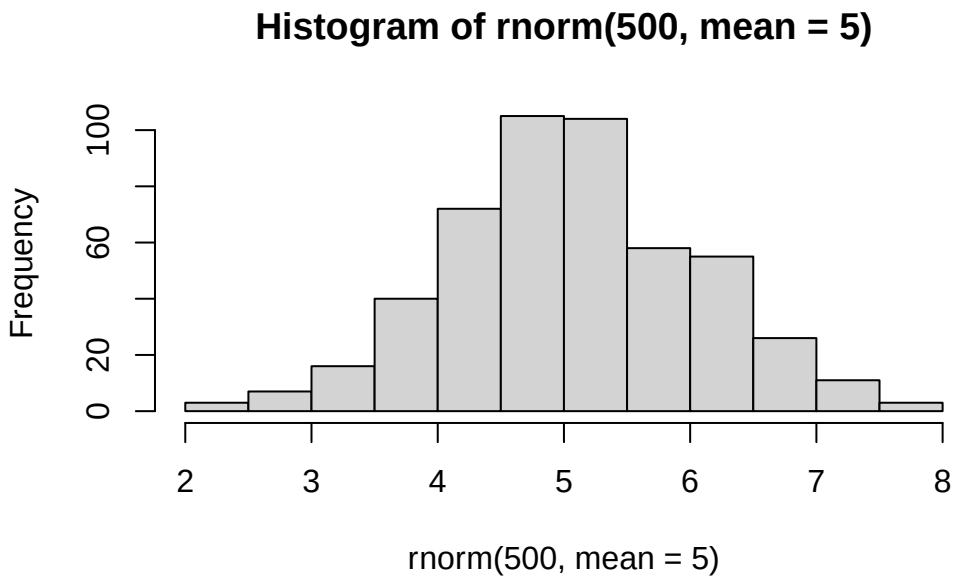
Joshua Martin (PID: A18545389)

Today we will explore some fundamental machine learning methods including clustering and dimensionality reduction.

K-means clustering

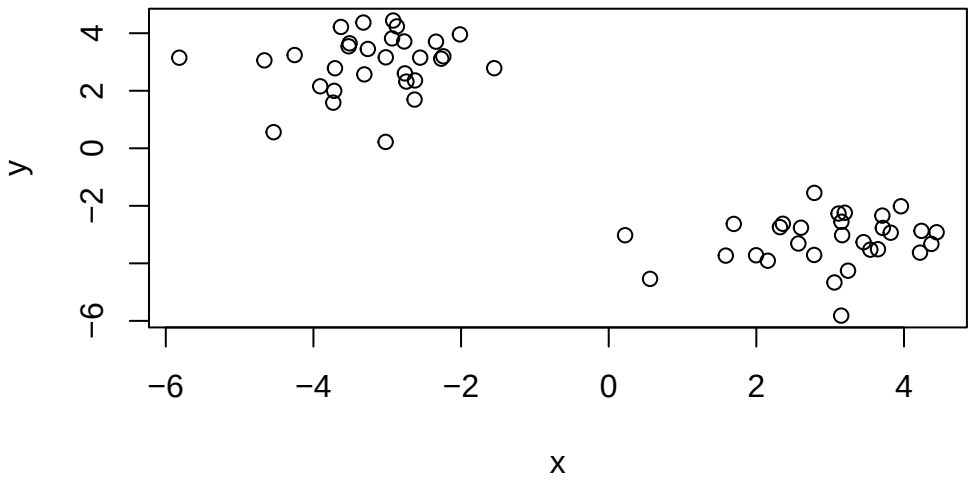
To see how this works let's first makeup some data to cluster where we know what the answer should be. We can use the `rnorm()` function to help here:

```
hist( rnorm(500, mean=5))
```



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

```r
x <- cbind(x,y)
plot(x)
```



The function for K-means clustering in "base" R is `kmeans()`

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

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

Cluster means:
          x         y
1 -3.205319  2.960707
2  2.960707 -3.205319

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] 53.30216 53.30216
 (between_SS / total_SS =  91.5 %)

Available components:

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

To get at the results of the returned list object we can use the dollar `$` syntax

Q. How many points are in each cluster?

```
k$size
```

```
[1] 30 30
```

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

```
head(k$cluster)
```

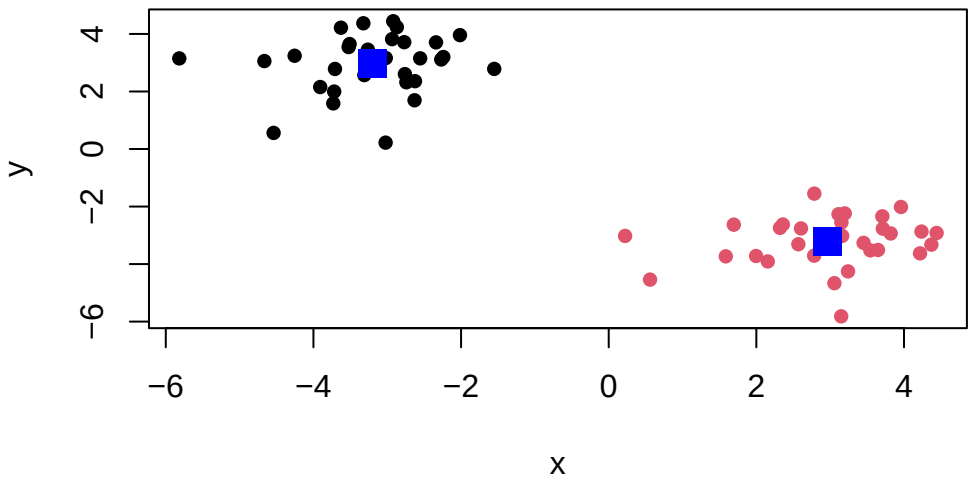
```
[1] 1 1 1 1 1 1
```

```
k$centers
```

```
          x         y
1 -3.205319  2.960707
2  2.960707 -3.205319
```

Q. Make a clustering results figure of the data colored by cluster membership.

```
plot(x, col=k$cluster, pch=16)
points(k$centers, col="blue", pch=15, cex=2)
```



K-means clustering is very popular as it is very fast and relatively straight forward: it takes numeric data input and returns the cluster membership vector etc.

The "issue" is we tell `kmeans()` how many clusters we want!

> Q. Run kmeans again and cluster into 4 grps/clusters and plot the results like we did above?

```
k4 <- kmeans(x, centers = 4)
k4
```

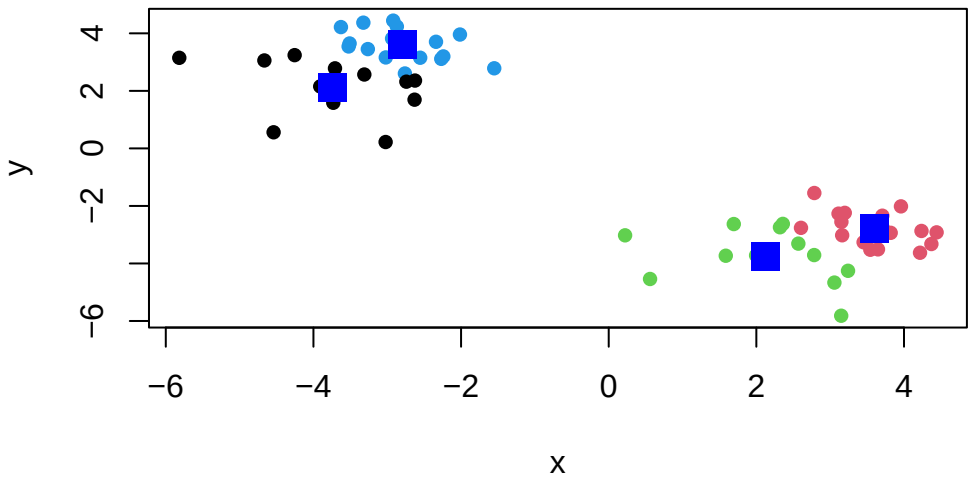
```
K-means clustering with 4 clusters of sizes 13, 17, 13, 17

Cluster means:
          x         y
1 -3.743862  2.130260
2  3.595756 -2.793493
3  2.130260 -3.743862
4 -2.793493  3.595756

Clustering vector:
 [1] 4 1 4 4 4 4 4 4 1 1 1 1 1 4 4 1 4 1 1 4 1 4 4 1 4 4 1 4 1 4 2 3 2 3 2 2 3 2
[39] 2 3 2 3 3 2 3 2 2 3 3 3 3 3 2 2 2 2 2 2 3 2
```

```
Within cluster sum of squares by cluster:
[1] 20.73095 10.09639 20.73095 10.09639
 (between_SS / total_SS =  95.1 %)

Available components:

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

```
k4$size
```

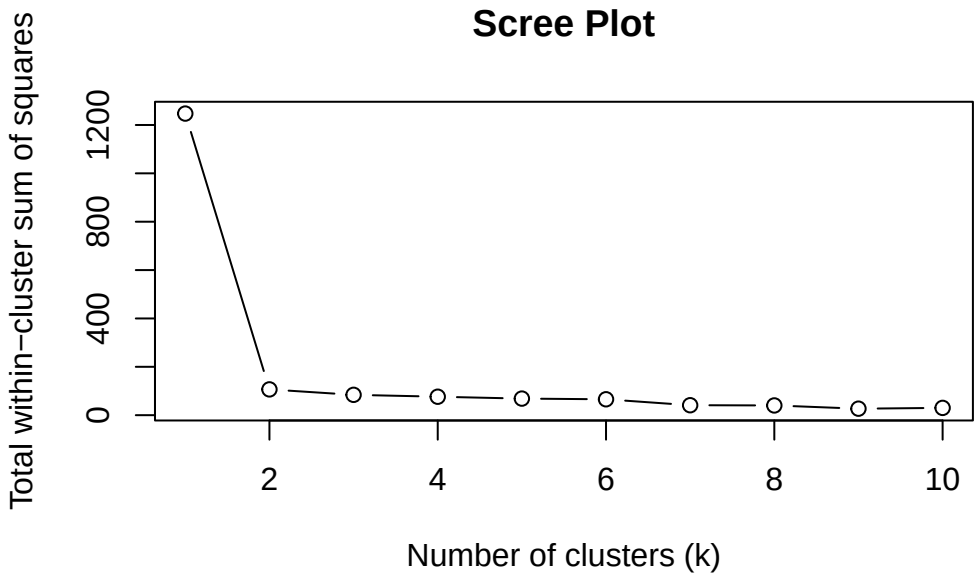
```
[1] 13 17 13 17
```

```
plot(x, col = k4$cluster, pch = 16)

points(k4$centers, col="blue", pch = 15, cex = 2)
```



Q. Generate a scree plot.

```r
# Basic scree plot
wss <- numeric(10)

for (k in 1:10) {
  wss[k] <- kmeans(x, centers = k)$tot.withinss
}

plot(1:10, wss, type = "b",
     xlab = "Number of clusters (k)",
     ylab = "Total within-cluster sum of squares",
     main = "Scree Plot")
```



Hierarchical Clustering

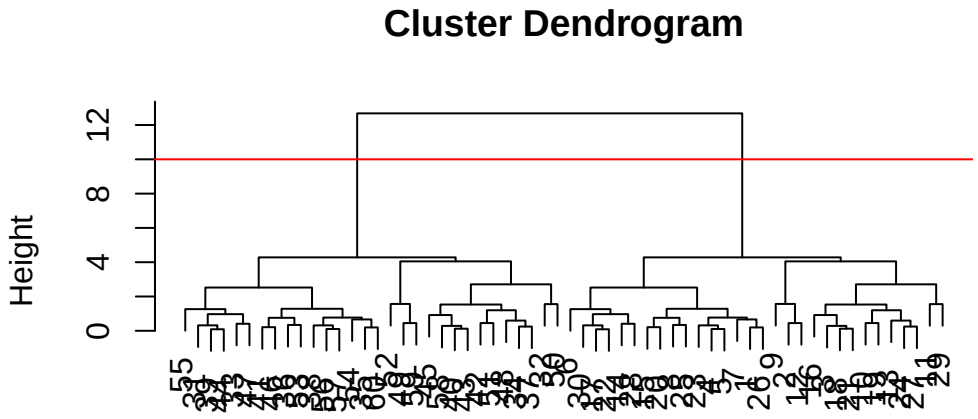
The main "base" R function for Hierarchical Clustering is called `hclust()`. Here we can't just input our data we need to first calculate a distance matrix (e.g., `dist()`) for our data and use this as input to `hclust()`

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

```
Call:
hclust(d = d)

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

There is a plot method for hclust results lets try it

```
plot(hc)
abline(h=10, col="red")
```



To get our cluster "membership" vector (i.e. our main clustering result) we can "cut" the tree at a given height or at a height that yields a given "k" groups.

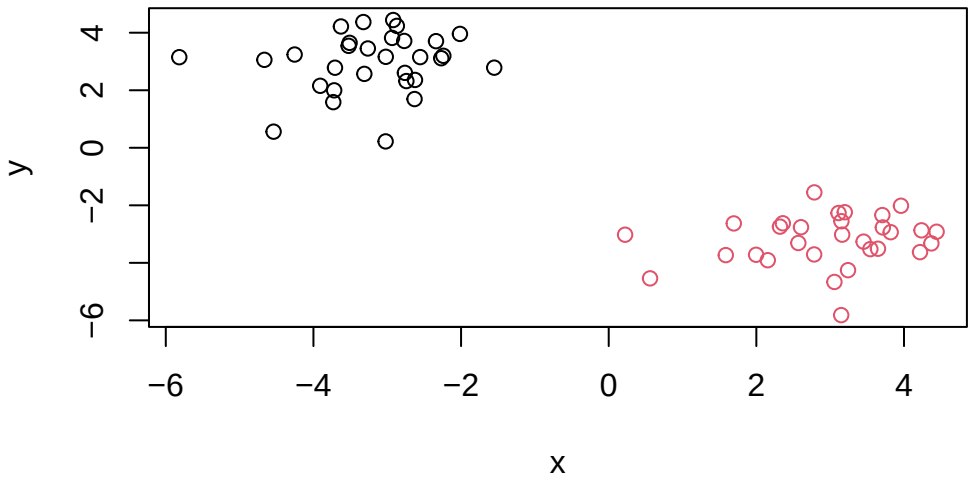
```
cutree(hc, h=10)
```

```
 [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
```

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

Q. Plot the data with our hclust result coloring

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



Principal Component Analysis (PCA)

PCA of UK food data

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

```
              X England Wales Scotland N.Ireland
1        Cheese     105   103      103        66
2  Carcass_meat     245   227      242       267
3    Other_meat     685   803      750       586
4          Fish     147   160      122        93
```

```
5 Fats_and_oils      193   235      184       209
6         Sugars     156   175      147       139
```

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

```
x <- read.csv("UK_foods.csv")
```

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

```
head(x)
```

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

```
dim(x)
```

```
[1] 17  5
```

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

```
dim(x)
```

```
[1] 17  4
```

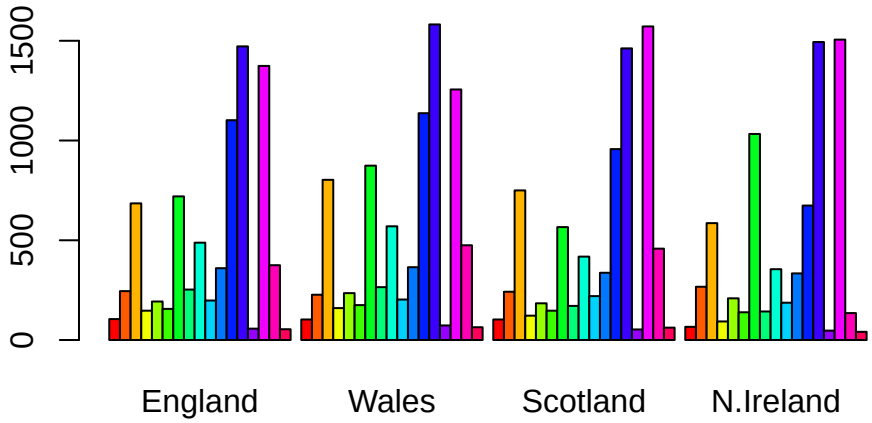
```
head(x)
```

```
              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?

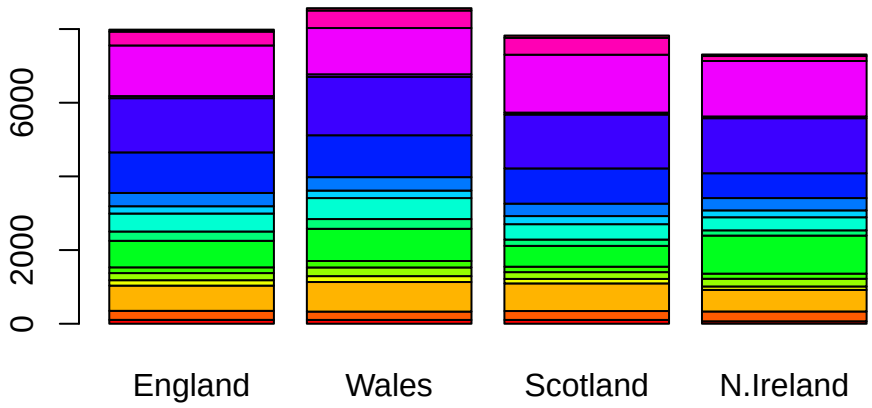
I prefer using row.names=1 argument in read.csv() when importing the data. More robust, cleaner.

```
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? bedside=FALSE

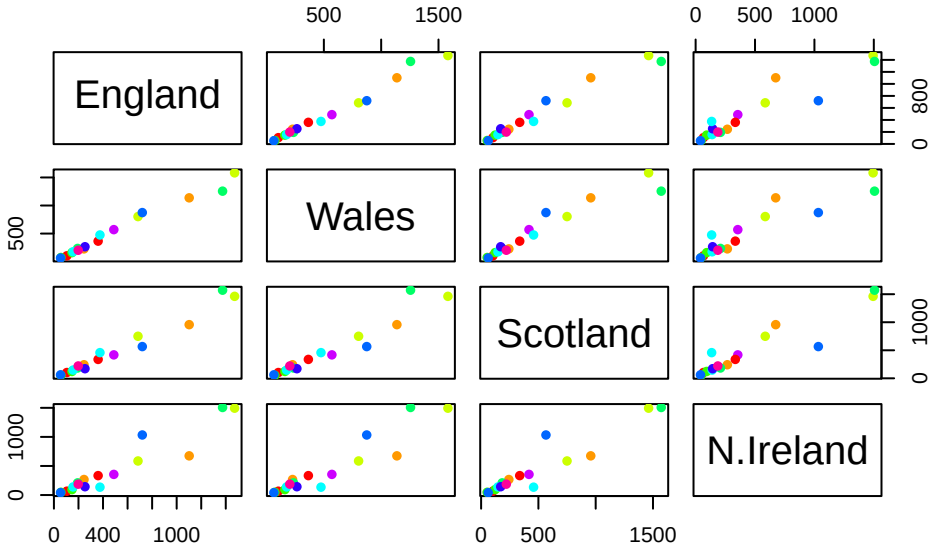
```
barplot(as.matrix(x), beside = FALSE, col = rainbow(nrow(x)))
```



Q4. 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?

If a point lies on the diagonal, it means that food is consumed equally in both countries and their values are similar or the same.

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



Main point: It can be difficult to spot major trends and patters even in relatively small multivariate datasets (here we only have 17 dimensions, typically we have 1000s)

PCA to the rescue

The main function in "base" R for PCA is called `prcomp()`

I will take the transpose of our data so the "foods" are in the columns:

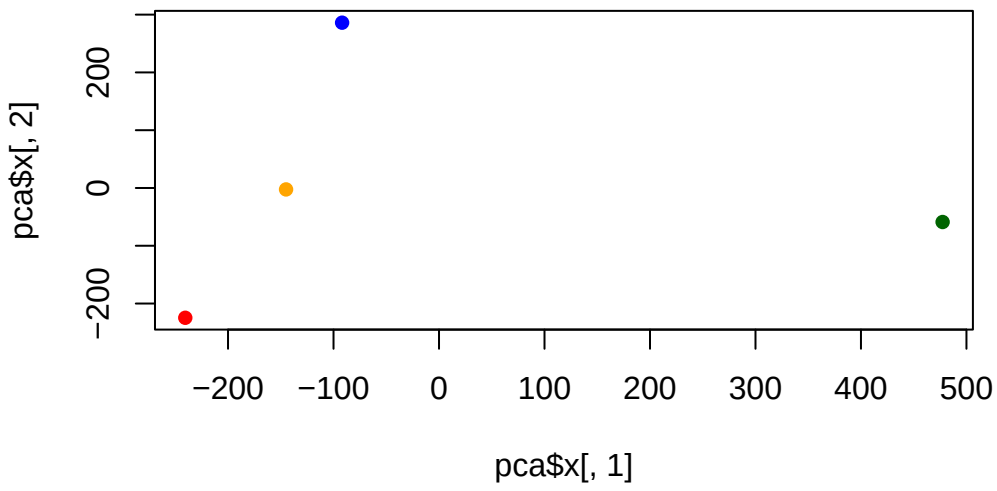
```
pca <- prcomp( t(x) )
summary(pca)
```

```
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
```

```
pca$x
```

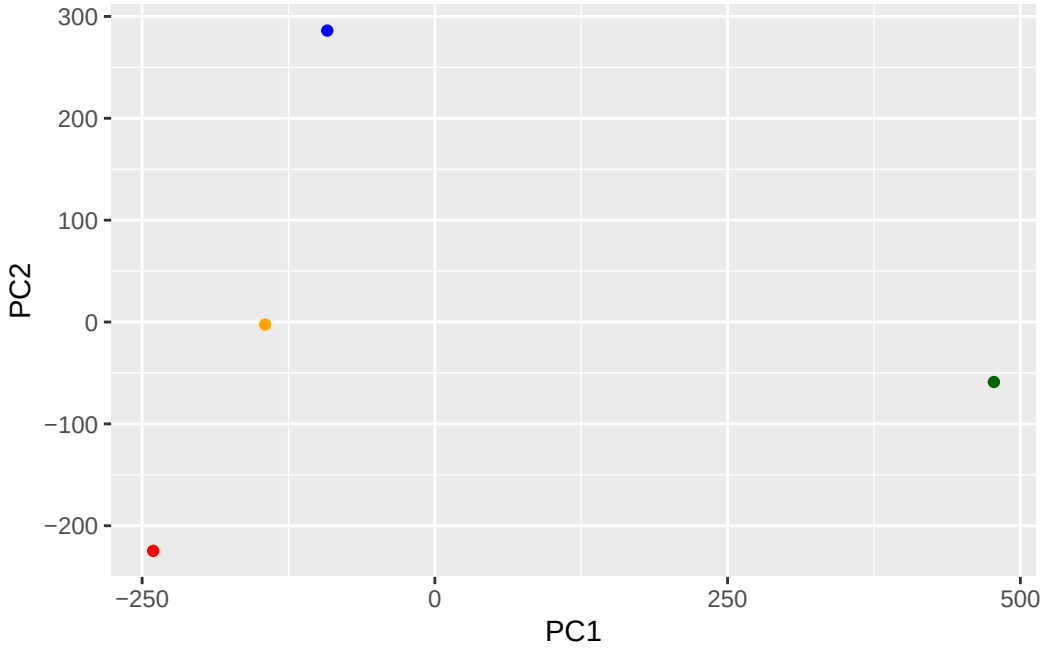
```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -9.152022e-15
Wales     -240.52915 -224.646925 -56.475555  5.560040e-13
Scotland   -91.86934  286.081786 -44.415495 -6.638419e-13
N.Ireland  477.39164  -58.901862  -4.877895  1.329771e-13
```

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

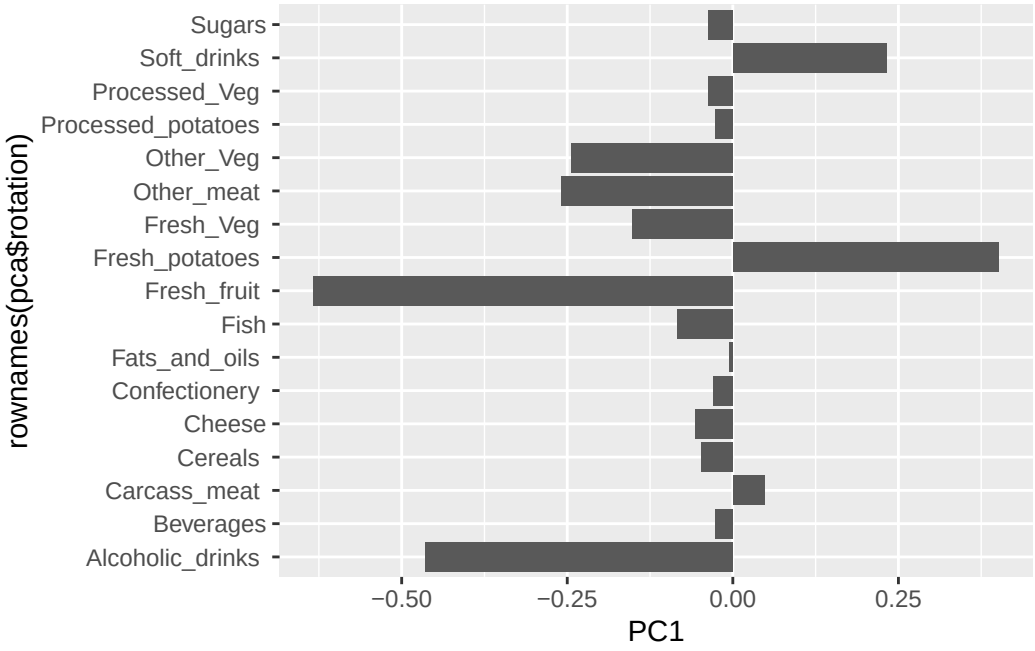



```
library(ggplot2)
```

```
ggplot(pca$x) +
  aes(PC1, PC2) +
  geom_point(col=cols)
```




```
ggplot(pca$rotation) +
  aes(PC1, rownames(pca$rotation)) +
  geom_col()
```



PCA looks super useful and will come back to describe this further next day :-)