class07: Machine Learning 1

Richard Gao (PID: A16490010)

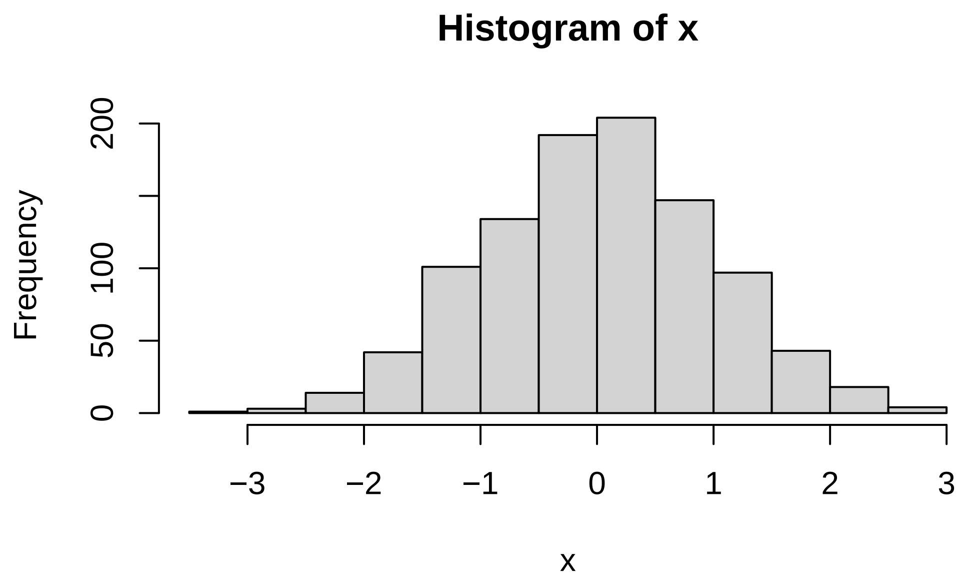
Clustering Methods

The broad goal here is to find grouping (clusters) in your input data.

Kmeans

First, let's make up some data to cluster.

```r
x <- rnorm(1000)
hist(x)
```

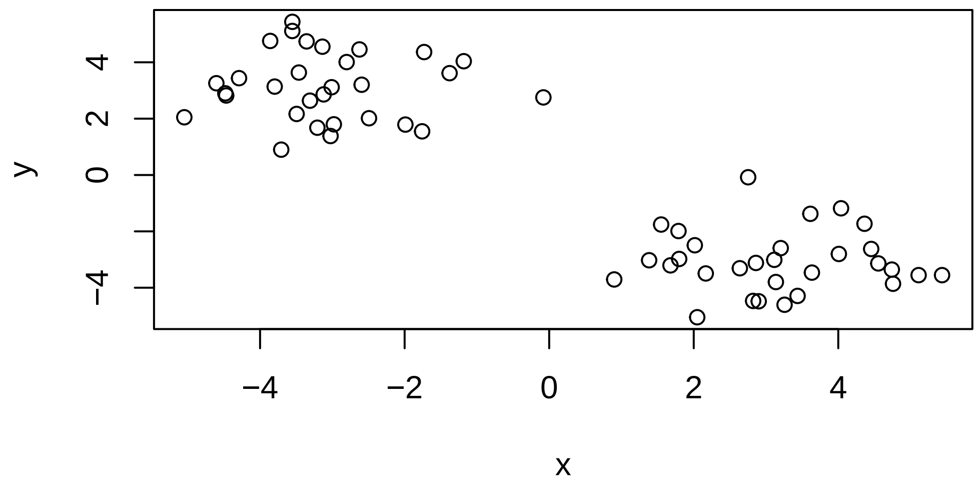


Make a vector of length 60 with 30 points centered at -3 and 30 points centered at +3

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

I will now make a wee x and y dataset with 2 groups of points.

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



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

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

Cluster means:
          x         y
1 -3.069859  3.138758
2  3.138758 -3.069859

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] 77.25998 77.25998
 (between_SS / total_SS =  88.2 %)

Available components:

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

Q. From your result object `k` how many points are in each cluster?

```r
k$size
```

```
[1] 30 30
```

Q. What “component” of your result object details the cluster membership?

```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
```

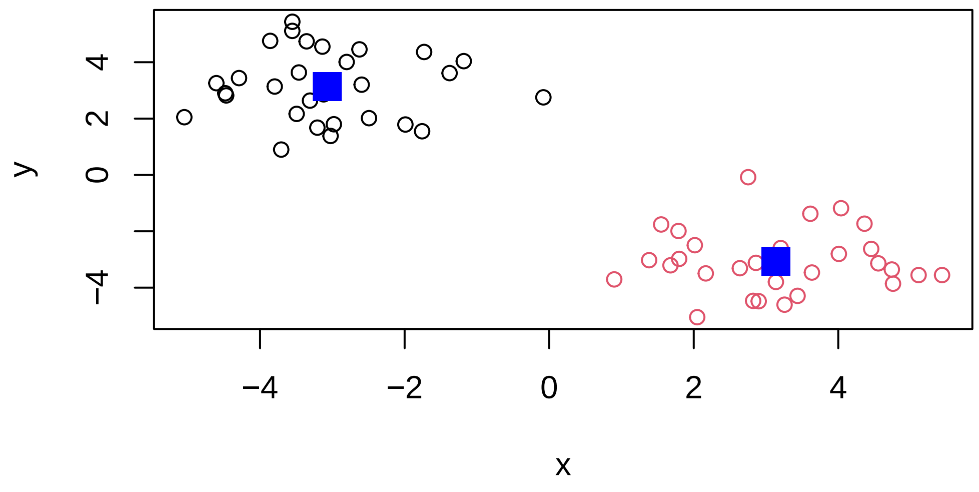
Q. Cluster centers?

```r
k$centers
```

```
          x         y
1 -3.069859  3.138758
2  3.138758 -3.069859
```

Q Plot of our cluster results?

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



We can cluster into 4 groups

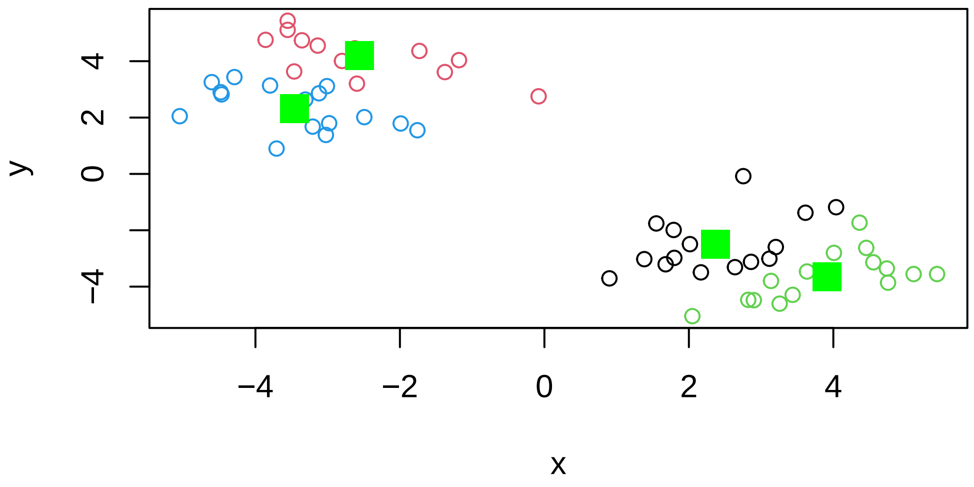
```
k_4 <- kmeans(x, centers=4)
k_4
```

```
K-means clustering with 4 clusters of sizes 15, 13, 15, 17

Cluster means:
          x         y
1  2.366830 -2.488355
2 -2.562647  4.205293
3  3.910687 -3.651363
4 -3.457727  2.323173

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

Within cluster sum of squares by cluster:
[1] 25.57827 22.63622 23.66111 22.62632
 (between_SS / total_SS =  92.8 %)
```

```
Available components:

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

Plot of Results with 4 Centers

```
plot(x, col=k_4$cluster)
points(k_4$centers, col="green", pch=15, cex=2)
```



A big limitation of kmeans is that it does what you ask even if you ask for silly clusters

Hierarchical Clustering

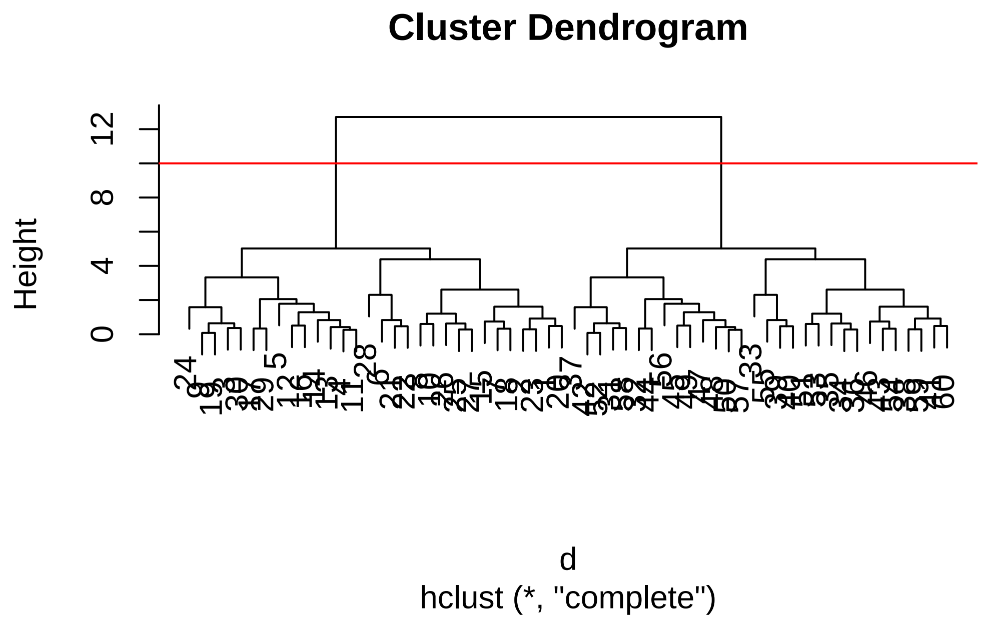
The main base R function for HIerarchical Clustering is `hclust()`. Unlike `kmeans()` you can not just pass it your data as input. You first need to calculate a distance matrix.

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

```
Call:
hclust(d = d)

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

Use `plot()` to view results

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



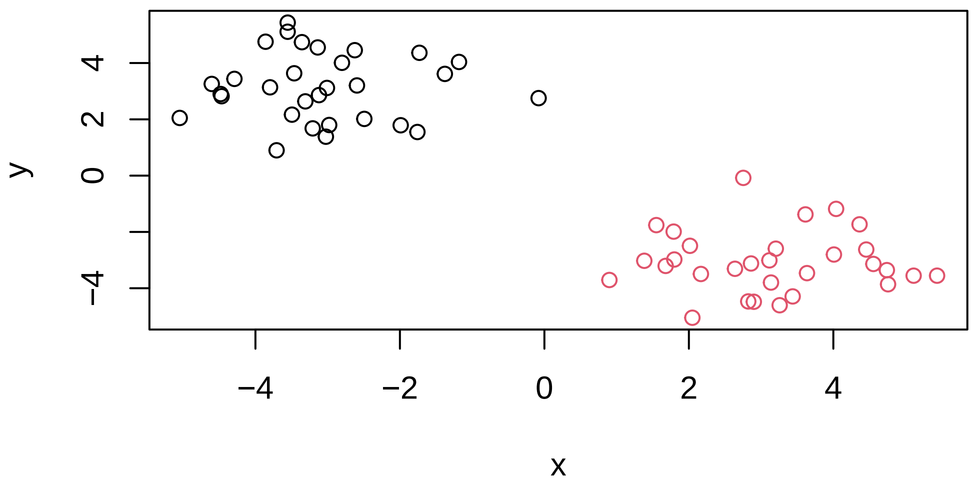
To make the "cut" and get our cluster membership vector we can use the `cutree()` function.

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

Make a plot of our data colored by hclust results

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



Principal Component ANalysis (PCA)

Here we will do Principal Component Analysis (PCA) on some food data from the UK.

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names=1) # sets the first column to be "row numbers" (actually name
View(x)
```

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
```

Preview the first 6 rows

```
head(x)
```

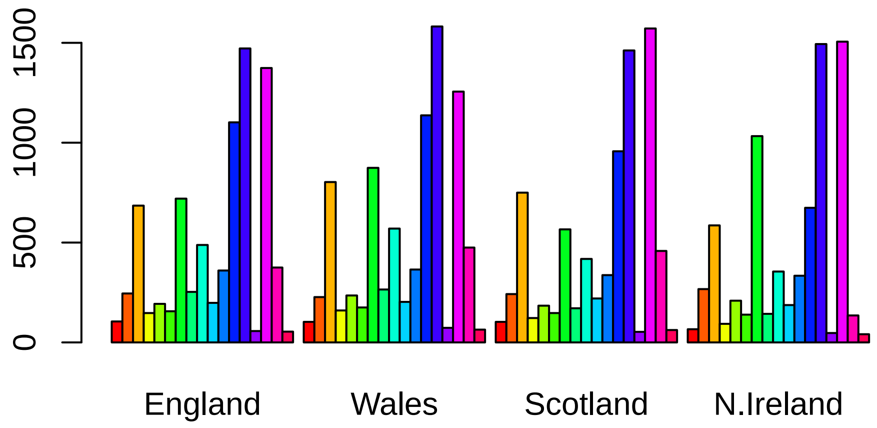
```
               England Wales Scotland N.Ireland
Cheese             105   103      103        66
Carcass_meat       245   227      242       267
Other_meat         685   803      750       586
Fish               147   160      122        93
Fats_and_oils      193   235      184       209
Sugars  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

Note how the minus indexing works - Set the first column to be row numbers.

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

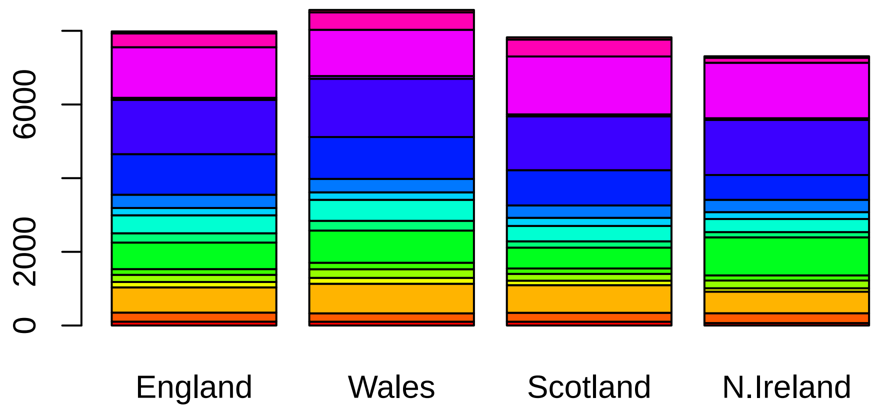
Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances? I prefer the editting of the read.csv function since the other method is iterative and will begin deleting columns when it's run repeatedly.

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



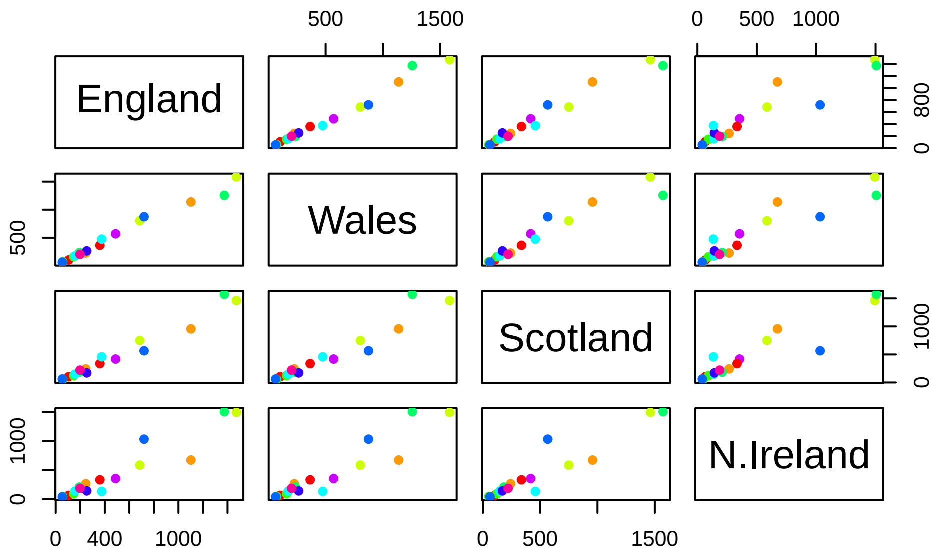
Q3: Changing what optional argument in the above barplot() function results in the following plot?

```
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? - Each individual box is the food information of one country plotted against another country. If a given point lies on the diagonal for a given plot it means that each country has the same value.

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



PCA to the rescue

The main "base" R function for PCA is called `prcomp()`. Here we need to take the transpose of our input as we want the countries in the rows and food as the columns.

Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set? N. Ireland is more different from the other countries of the UK in the consumption of foods, since the graphs involving N. Ireland deviate more from the diagonal than the graphs not involving them. Particularly, the blue and orange points are most different from that of other countries but we are unable to identify what foods they correspond to.

```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
```

Q. How much variance is captured in 2 PCs

96.5%

To make our main "PC score plot" or "PC1 vs PC2 plot", or "PC plot" or "ordienation plot".

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

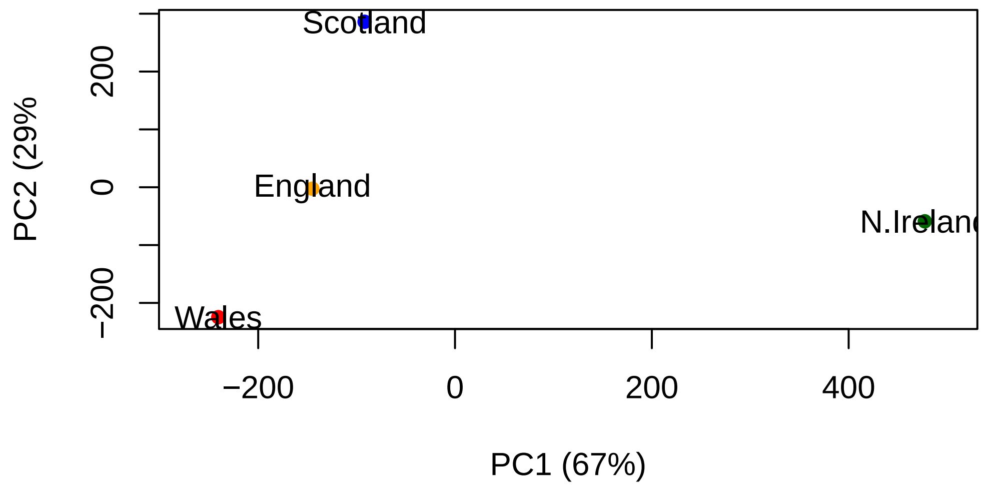
We are after the `pca$x` result component to make our main PCA plot.

```
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
```

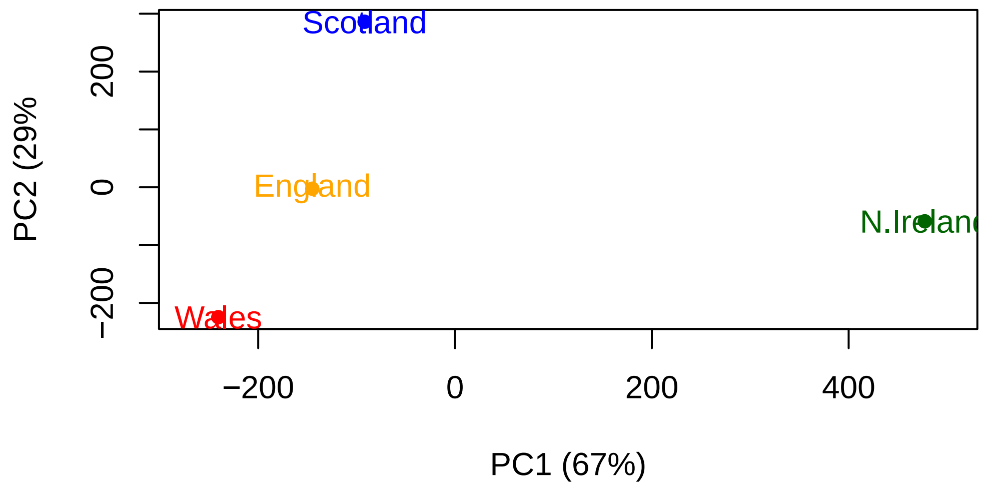
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
mycols <- c("orange", "red", "blue", "darkgreen")
plot(pca$x[,1], pca$x[,2], col=mycols, pch=16, xlab="PC1 (67%)", ylab="PC2 (29%", xlim=c(-
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.

```
mycols <- c("orange", "red", "blue", "darkgreen")
plot(pca$x[,1], pca$x[,2], col=mycols, pch=16, xlab="PC1 (67%)", ylab="PC2 (29%", xlim=c(-
text(pca$x[,1], pca$x[,2], colnames(x), col = mycols)
```



Another important result from PCA is how the original variables (in this case the foods) contribute to the PCs.

This is contained in the `pca$rotation` object - folks often call this the "loadings" or "contributions" to the PCs

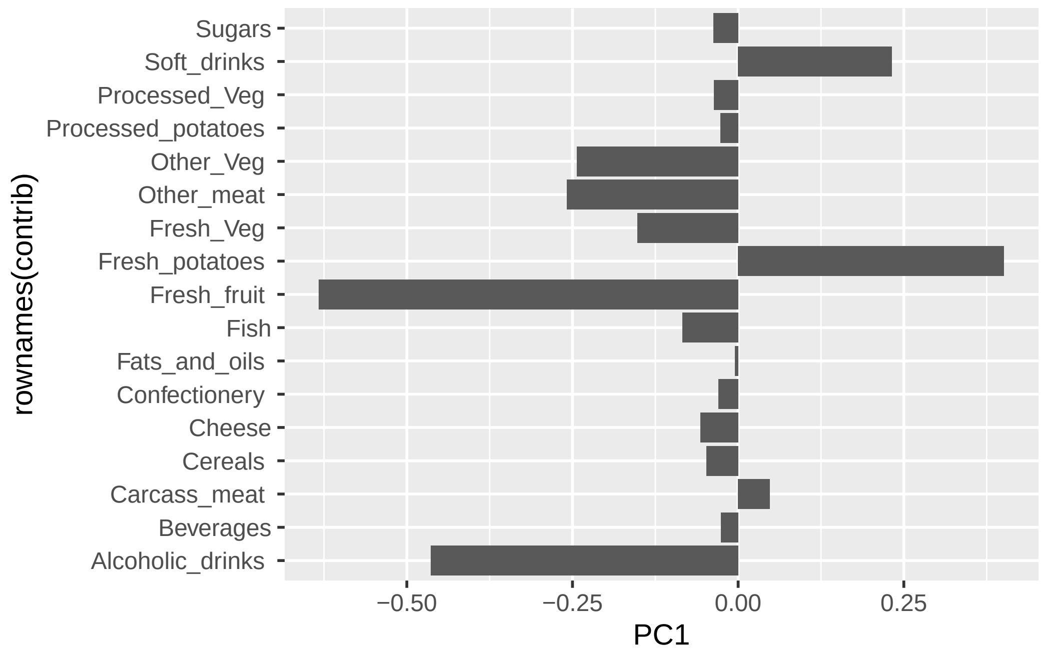
```
pca$rotation
```

```
                           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
```

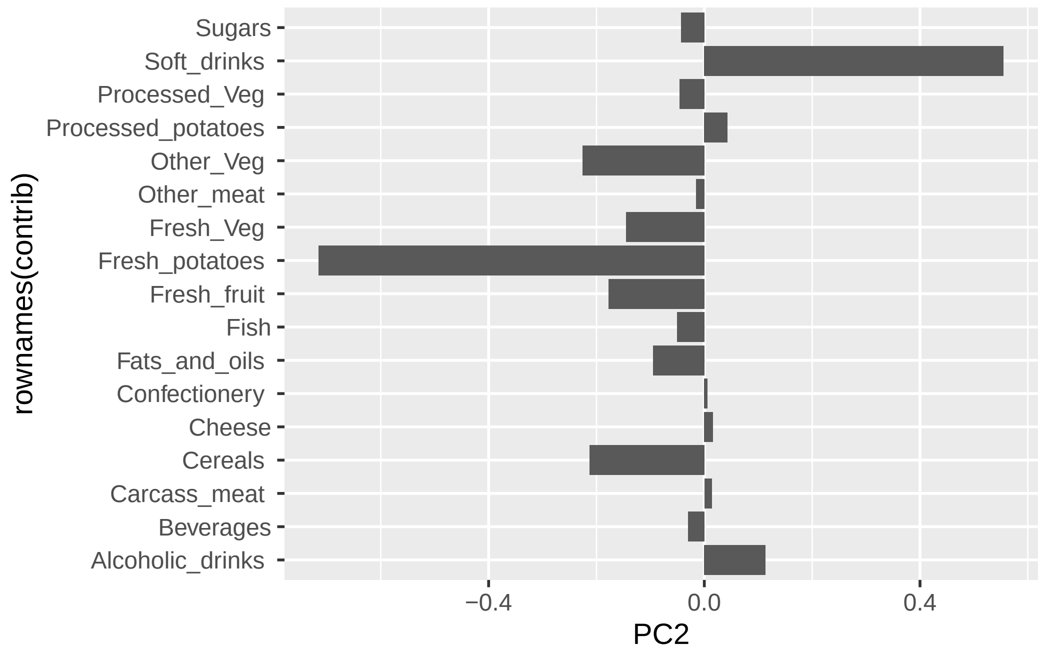
We can make a plot along PC1.

```
library(ggplot2)

contrib <- as.data.frame(pca$rotation)

ggplot(contrib) +
  aes(PC1, rownames(contrib)) +
  geom_col()
```



Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about? - Fresh potatoes and soft drinks. - PC2 mainly tells us that the secondary variance is driven mainly by the difference in production of fresh potatoes and soft drinks.

```r
ggplot(contrib) +
  aes(PC2, rownames(contrib)) +
  geom_col()
```


Sugars
Soft_drinks
Processed_Veg
Processed_potatoes
Other_Veg
Other_meat
Fresh_Veg
Fresh_potatoes
Fresh_fruit
Fish
Fats_and_oils
Confectionery
Cheese
Cereals
Carcass_meat
Beverages
Alcoholic_drinks
rownames(contrib)
PC2