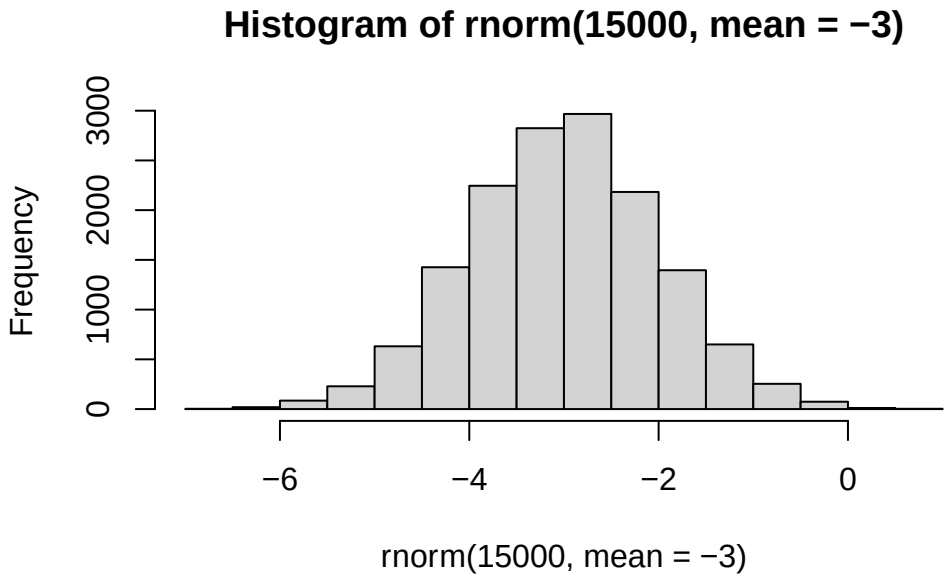
Class 7: Machine Learning 1

Emily Hendrickson (PID: A69034780)

First, make a control dataset to run tests for clustering methods.

To do this, I will use the *rnorm()* function.

```
hist(rnorm(15000, mean = -3))
```

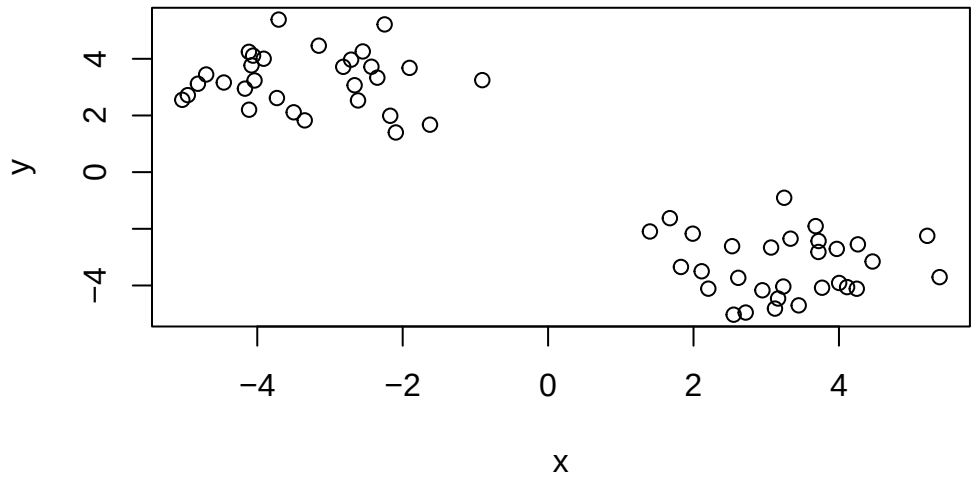



```
n=30
x <- c(rnorm(n, mean = 3), rnorm(n, mean = -3))
y <- rev(x)

z <- cbind(x,y)
z
```

```
              x          y
 [1,]  1.8266057 -3.3471949
 [2,]  3.2346561 -4.0376206
 [3,]  4.2599938 -2.5492661
 [4,]  2.7176280 -4.9554233
 [5,]  3.9701214 -2.7095457
 [6,]  3.6797354 -1.9054722
 [7,]  2.1120916 -3.4988553
 [8,]  3.4464018 -4.7027957
 [9,]  4.0025660 -3.9119321
[10,]  3.3343175 -2.3487802
[11,]  1.6745866 -1.6245805
[12,]  2.5521230 -5.0313193
[13,]  3.1629314 -4.4602467
[14,]  3.2469560 -0.9051491
[15,]  2.6152825 -3.7298416
[16,]  3.7680025 -4.0793705
[17,]  5.3846422 -3.7057275
[18,]  3.7196431 -2.4301705
[19,]  5.2144135 -2.2481711
[20,]  3.0667192 -2.6611745
[21,]  4.4625702 -3.1542336
[22,]  3.1199212 -4.8161400
[23,]  2.2032628 -4.1124318
[24,]  4.1118572 -4.0577939
[25,]  1.4000201 -2.0946414
[26,]  2.5306913 -2.6142891
[27,]  4.2451616 -4.1152087
[28,]  3.7178573 -2.8175446
[29,]  1.9890756 -2.1712261
[30,]  2.9470494 -4.1695630
[31,] -4.1695630  2.9470494
[32,] -2.1712261  1.9890756
[33,] -2.8175446  3.7178573
[34,] -4.1152087  4.2451616
[35,] -2.6142891  2.5306913
[36,] -2.0946414  1.4000201
[37,] -4.0577939  4.1118572
[38,] -4.1124318  2.2032628
[39,] -4.8161400  3.1199212
[40,] -3.1542336  4.4625702
[41,] -2.6611745  3.0667192
[42,] -2.2481711  5.2144135
```

```
[43,] -2.4301705  3.7196431
[44,] -3.7057275  5.3846422
[45,] -4.0793705  3.7680025
[46,] -3.7298416  2.6152825
[47,] -0.9051491  3.2469560
[48,] -4.4602467  3.1629314
[49,] -5.0313193  2.5521230
[50,] -1.6245805  1.6745866
[51,] -2.3487802  3.3343175
[52,] -3.9119321  4.0025660
[53,] -4.7027957  3.4464018
[54,] -3.4988553  2.1120916
[55,] -1.9054722  3.6797354
[56,] -2.7095457  3.9701214
[57,] -4.9554233  2.7176280
[58,] -2.5492661  4.2599938
[59,] -4.0376206  3.2346561
[60,] -3.3471949  1.8266057
```

```
plot(z)
```



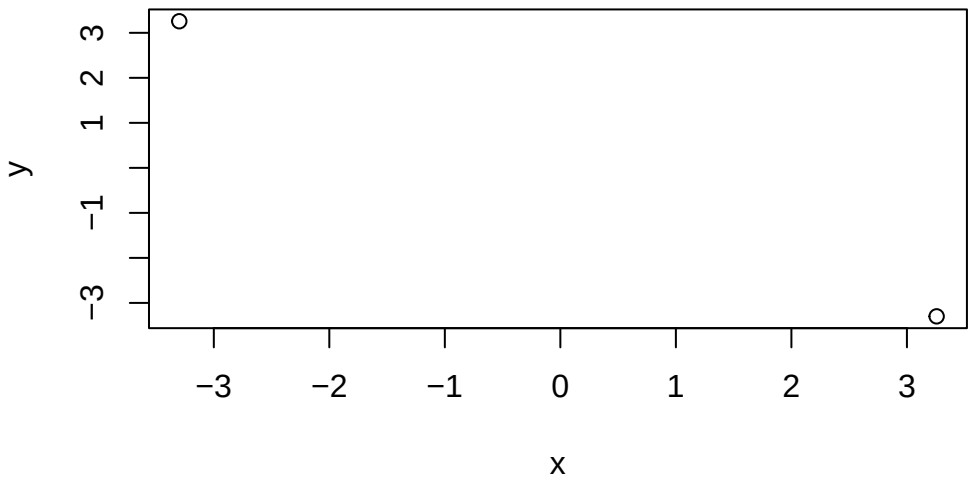
K-means Clustering

To perform k-means clustering, use the function *kmeans()*

```r
km <- kmeans(z, 2)
km
```

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

Cluster means:
          x         y
1  3.257229 -3.298857
2 -3.298857  3.257229

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 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] 62.36139 62.36139
 (between_SS / total_SS =  91.2 %)

Available components:

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

```r
plot(km$centers)
```

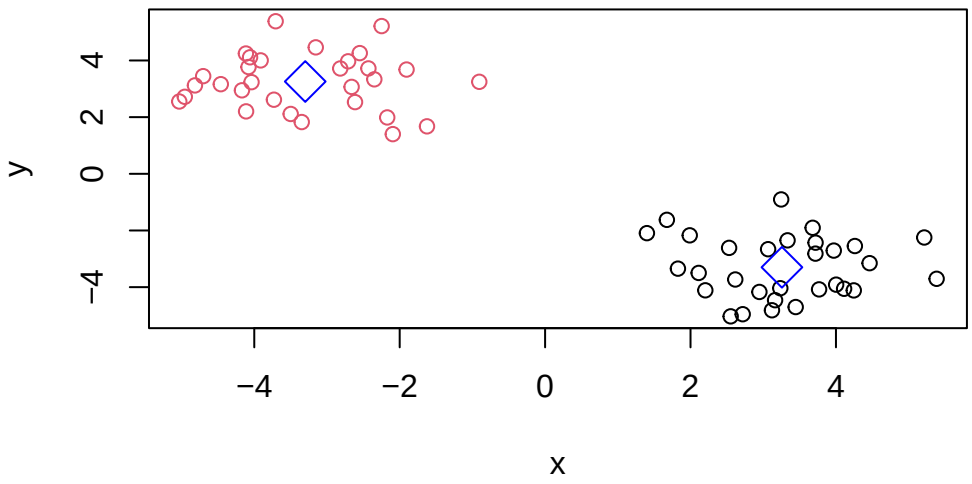


Print the cluster vector and add cluster centers

```
km$cluster
```

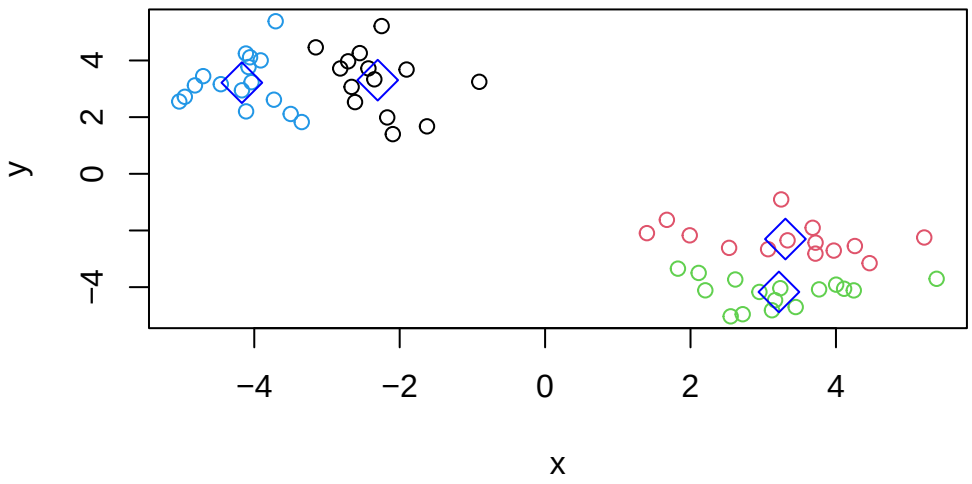
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
plot(z, col = km$cluster)
points(km$centers, cex = 2, col = "blue", pch = 5)
```



Can you cluster our data in z into four clusters please?

```
km4 <- kmeans(z, centers = 4)
plot(z, col = km4$cluster)
points(km4$centers, cex = 2, col = "blue", pch = 5)
```

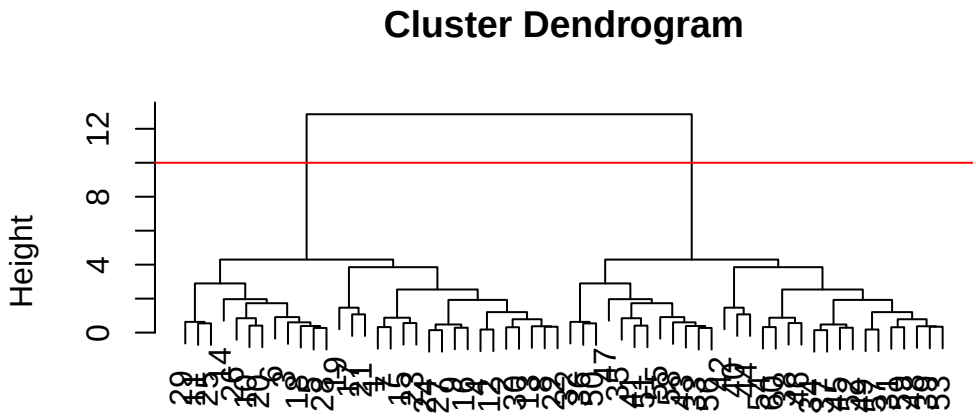


Hierarchical Clustering

The main function in base R for heirarchical clustering is *hclust()*

Unlike k-means, I can't just lazily pass in my data. I have to do some work and make a distance matrix of my data.

```
z.dist <- dist(z)
z.hc <- hclust(z.dist)
plot(z.hc)
abline(h=10, col = "red")
```

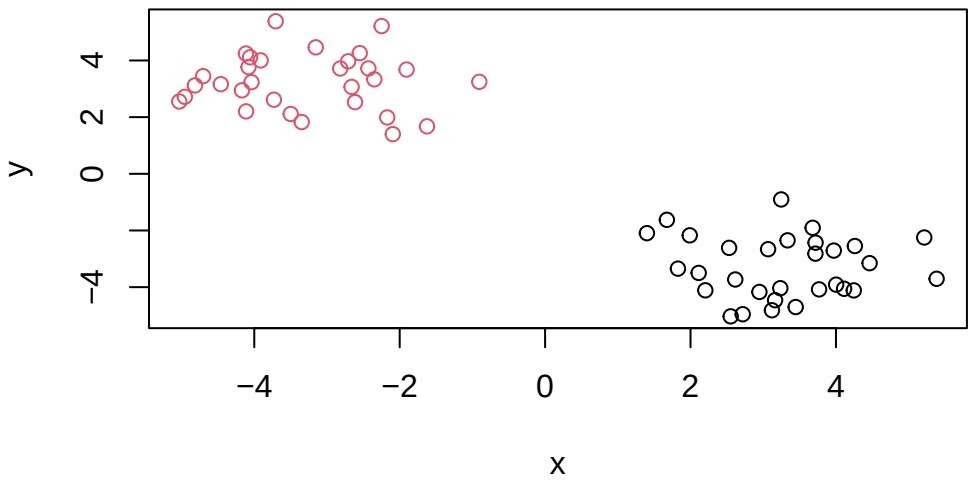


To get my main clustering restult, I can cut my tree at a given height.

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



Principal Component Analysis - Class 7 Lab

Q1

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

```
[1] 17  5
```

Preview the first 6 rows

```
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
```

Note how the minus indexing works

```
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
```

Alternative approach:

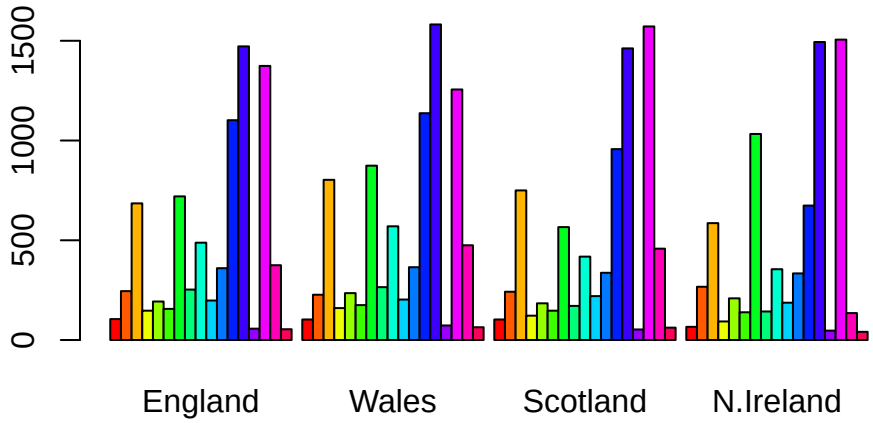
```
x<- read.csv(url, row.names=1)
head(x)
```

```
               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: I prefer the secon

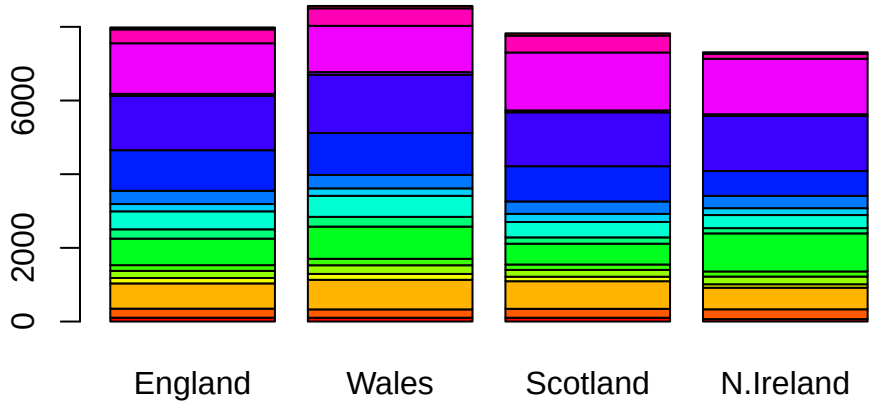
Making a barplot

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



Changing the argument beside = T to the default beside = F makes the following plot

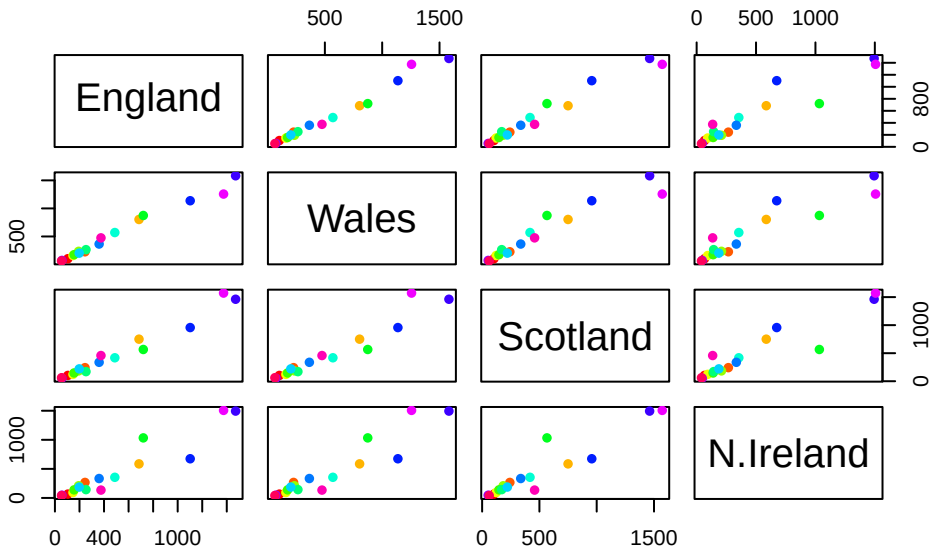
```
barplot(as.matrix(x), beside = F, col = rainbow(nrow(x)))
```



Making pairs plot.

Q5 The code is generating plots of pairwise comparisons between each food category for each country. X axis is the country in that row, and the Y axis is the country in that column. If a dot falls on the diagonal, it means the value for that dot is the same for each country.

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



Q6

The main function to do PCA in base R is called *prcomp()* To analyze our data with PCA, we need to transpose it.

```
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
```

Let's see what's inside

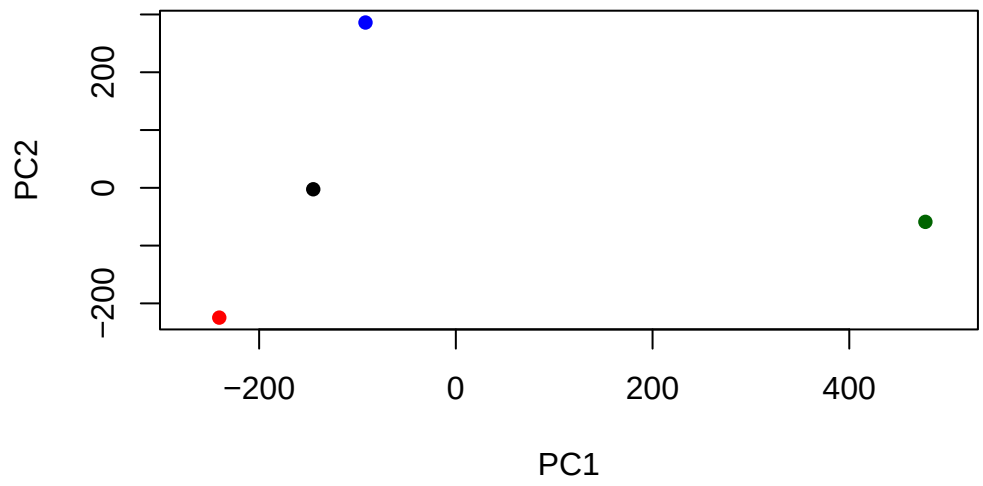
```
attributes(pca)
```

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

$class
[1] "prcomp"
```

Q7

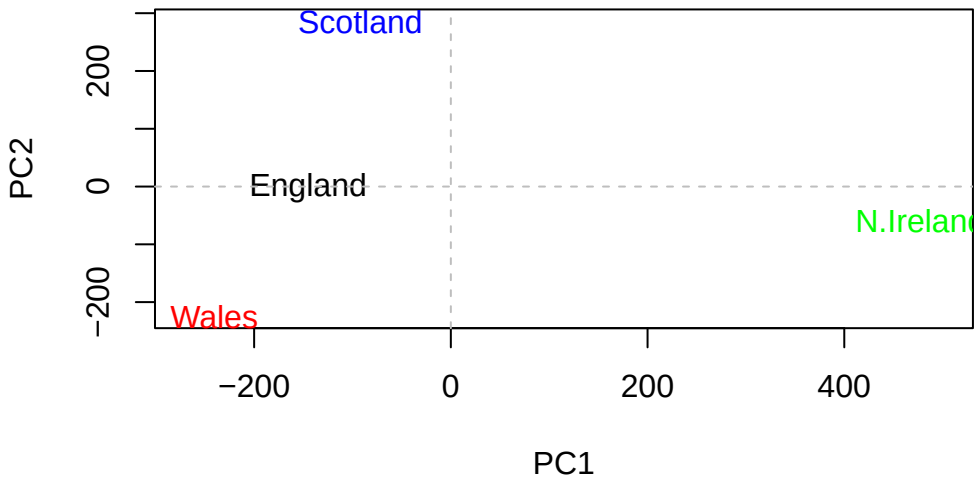
Our main result figure by plotting PC1 vs PC2.

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500), col = c("black", "red",
```



With color

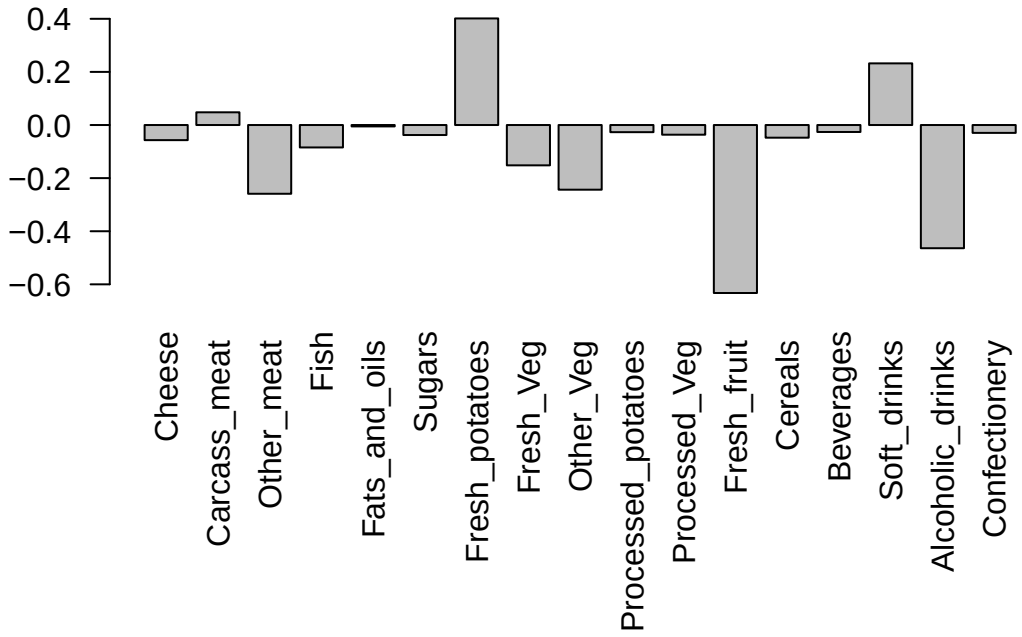
```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500), cex = 0)
text(pca$x[,1], pca$x[,2], colnames(x), col = c("black", "red", "blue", "green"))
abline(v=0, col = "gray", lty = 2)
abline(h=0, col = "gray", lty = 2)
```



Percent Variation

Variable loadings with PC1 because it captures most of the variance (67%)

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




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