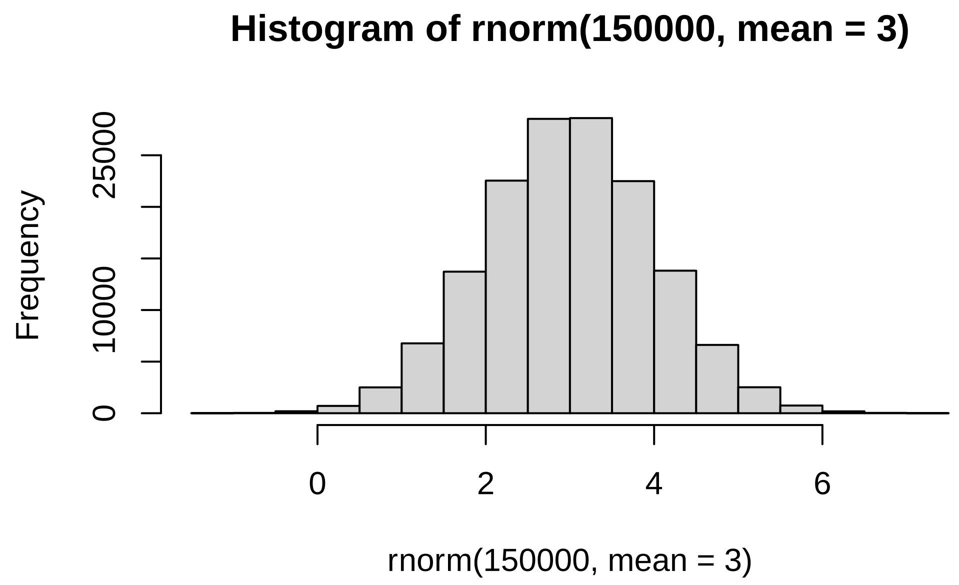
Class 07: Machine Learning 1

Vanesa Fernandez

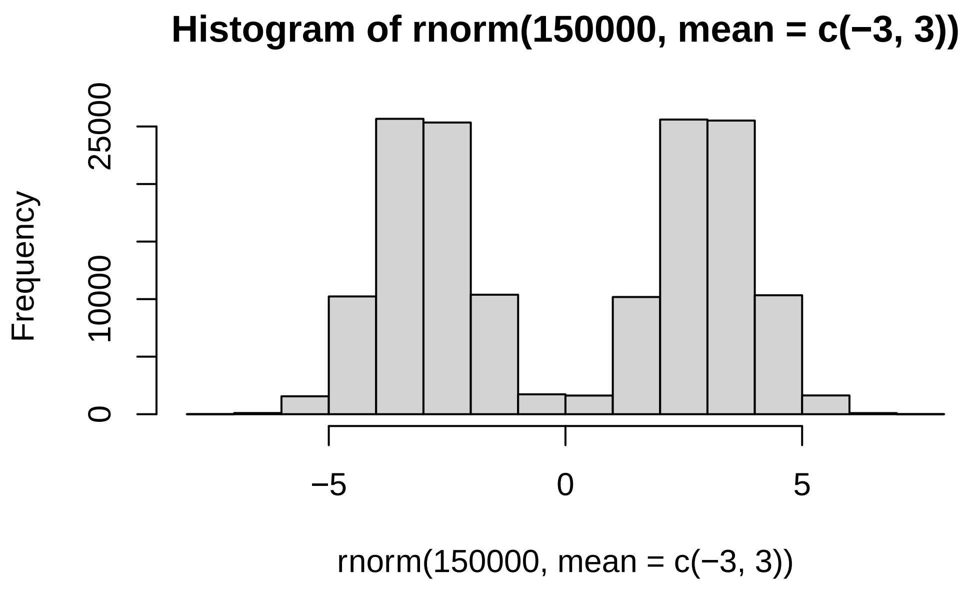
Before we get into clustering methods let's make some sample data to cluster where we know what the answer should be.

To help with this I will use the 'rnorm()' fuction.

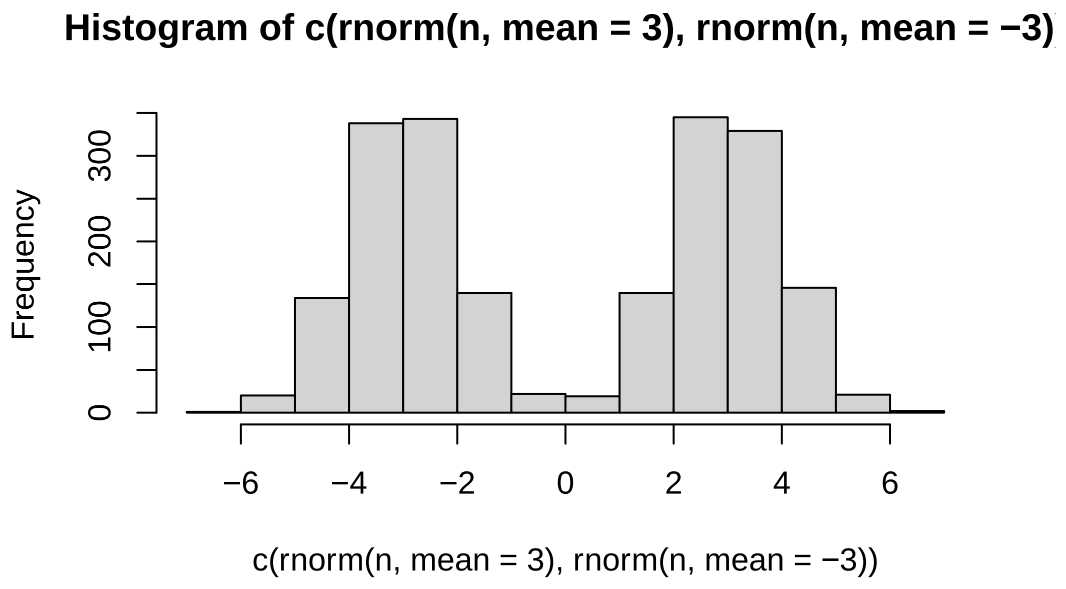
```r
hist( rnorm(150000, mean= 3))
```



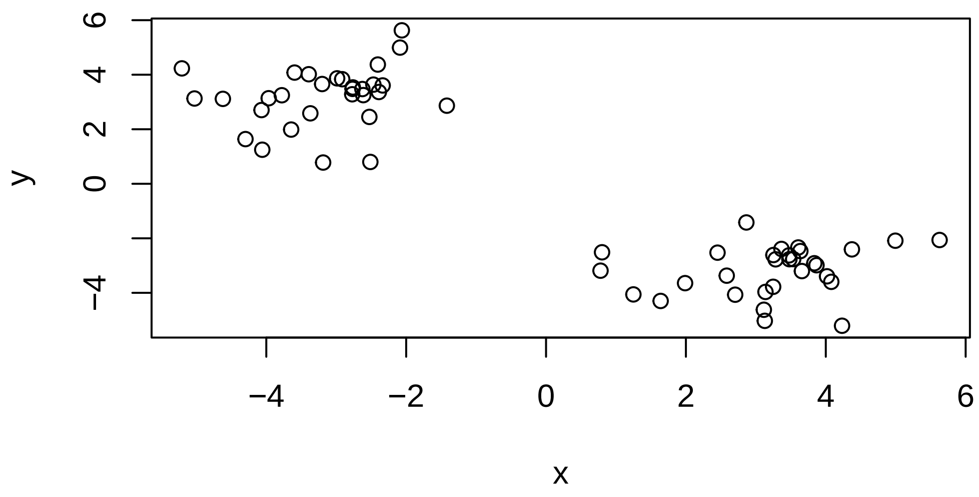
```r
hist( rnorm(150000, mean=c(-3,3 )))
```



```r
n=1000
hist( c(rnorm(n, mean=3), rnorm(n, mean=-3) ))
```



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

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



K-means clustering

works well with big datasets

The function in base R for k-means clustering is called `kmeans()`.

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

```r
km$centers
```

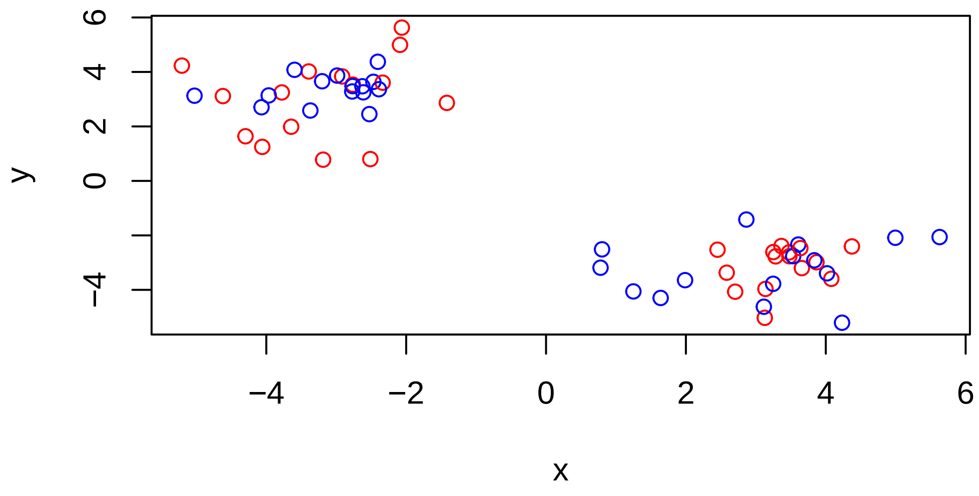
```
          x         y
1 -3.168839  3.200134
2  3.200134 -3.168839
```

Q. Print out the cluster membership vector (i.e. our main answer)

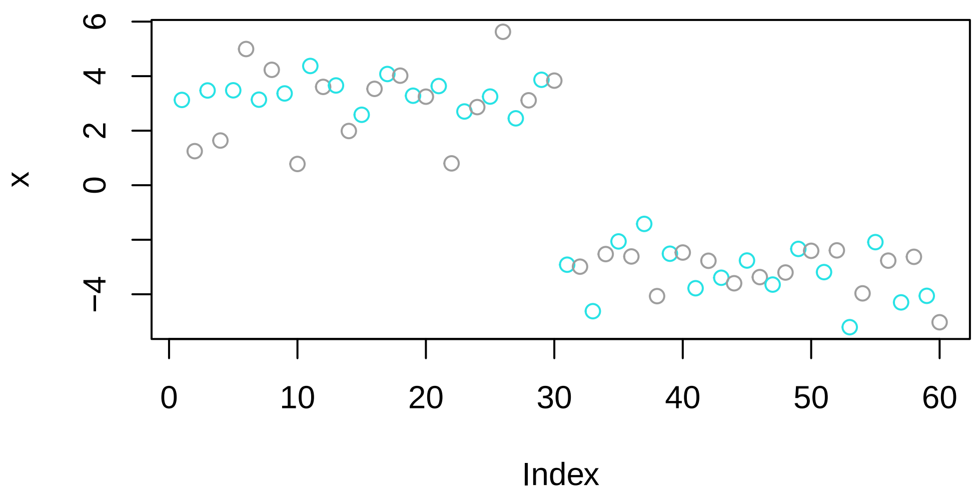
```
km$cluster
```

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

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

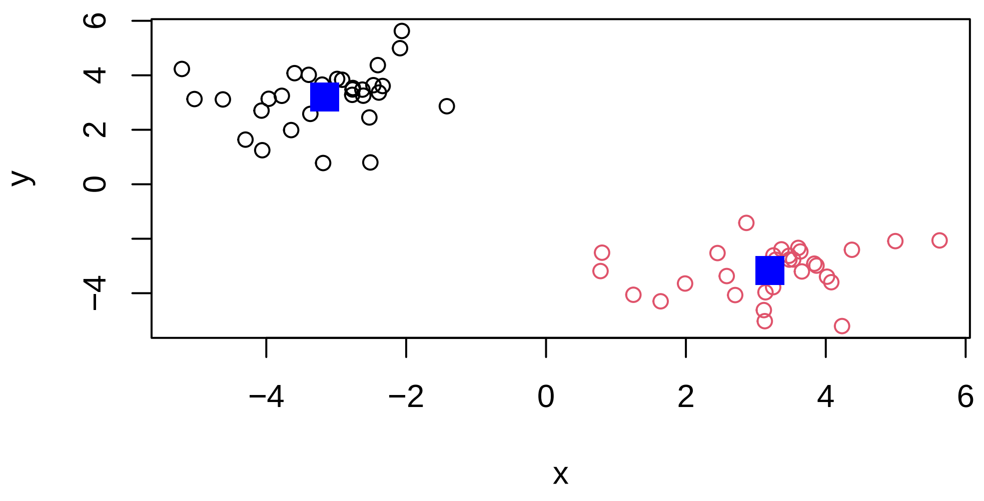


```
plot(x, col=c(125,200) )
```



Plot with clustering result and add cluster centers:

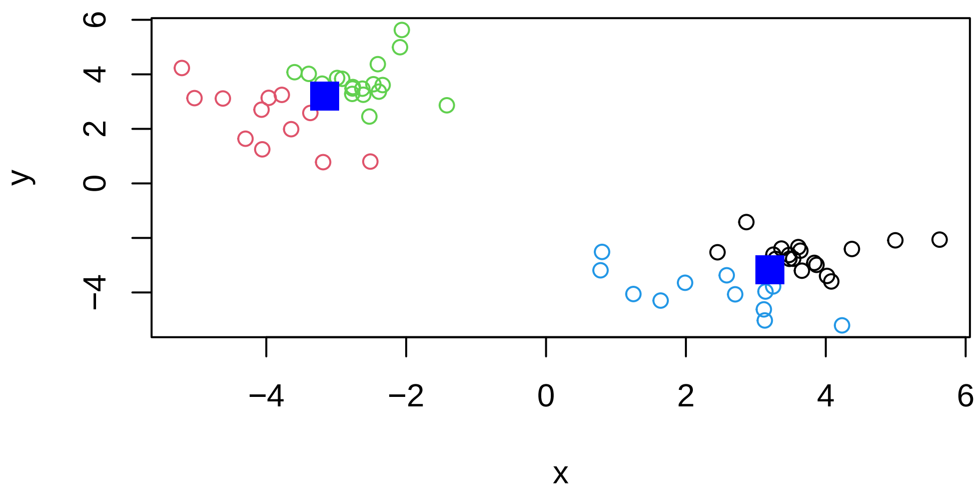
```
plot(z, col=km$cluster)
points(km$centers, col="blue", pch=15, cex=2)
```



Q. Can you cluster our data in `z` into four clusters?

Kmeans would fit the data but not the accurate plot

```
km4 <- kmeans(z, centers = 4)
plot(z, col=km4$cluster)
points(km$centers, col="blue", pch=15, cex=2)
```



Hierarchical Clustering

b-up or b-down clusters

The main function for hierarchical clustering in base R is called `hclust()`

Unlike `kmeans()` I can not just pass in my data as input I first need a distance matrix from my data.

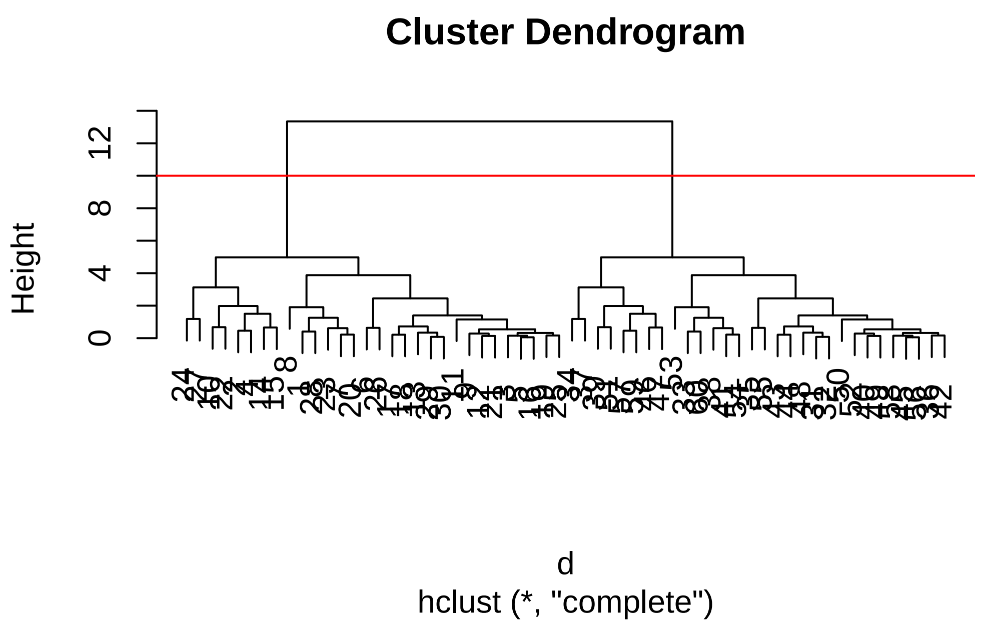
```
d <- dist(z)
hc <- hclust(d)
hc
```

```
Call:
hclust(d = d)

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

There is a specific hclust plot() method…

##the horizontal line = how close/similar the clusters are. ## the red line = height

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



To get my clustering result (i.e. the membership vector) I can "cut" my tree at a given height. To do this I will use the `cutree()`

```
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
```

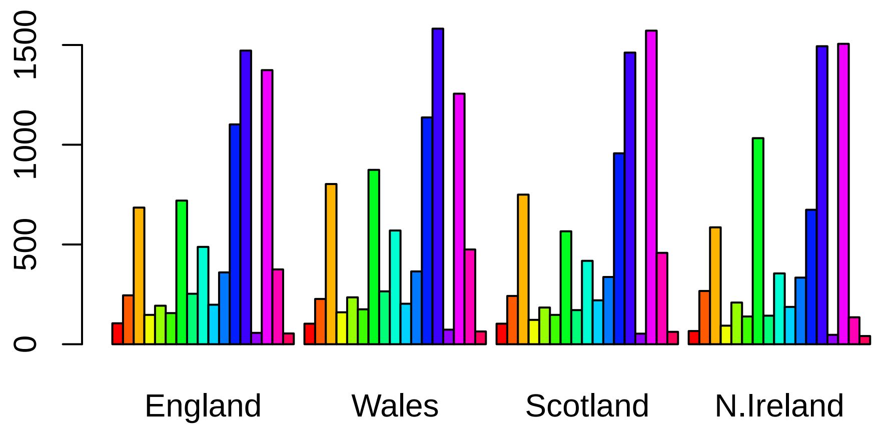
Principal Component Analysis

Principal component analysis (PCA) is a well established “multivariate statistical technique” used to reduce the dimensionality of a complex data set to a more manageable number (typically 2D or 3D). This method is particularly useful for highlighting strong patterns and relationships in large datasets (i.e. revealing major similarities and differences) that are otherwise hard to visualize.

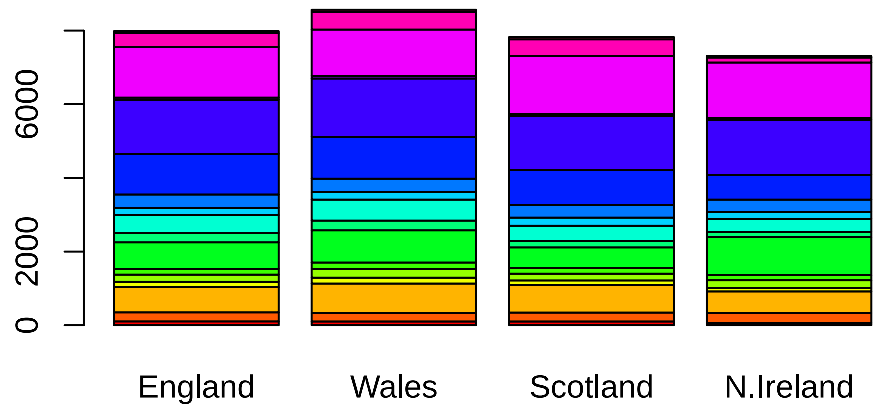
##PCA of UK food data

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

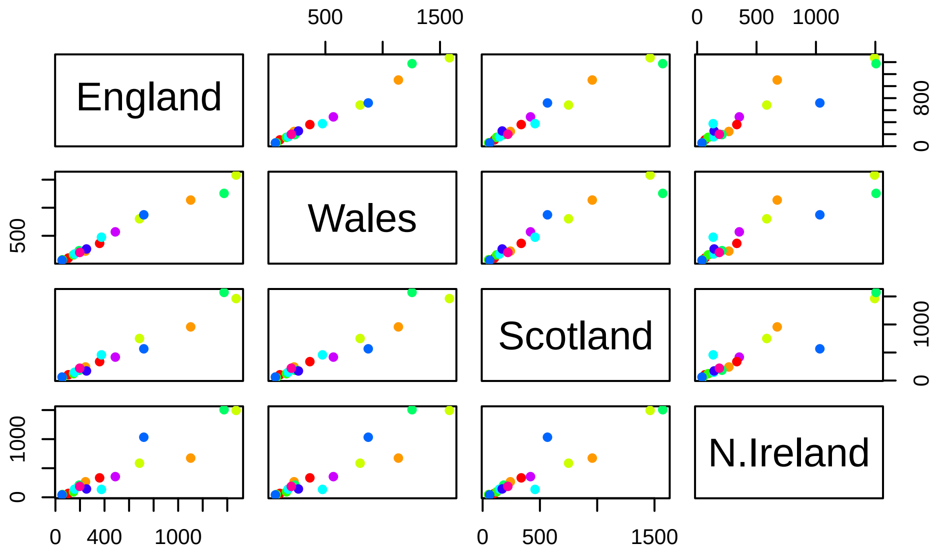
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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



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



PCA to the rescue

The main function to do PCA in base R is called `prcomp()`. `center` and `scale` important arguments

Note that I need to take the transpose of this particular data as that is what the `prcomp()` help page was asking for

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

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 3.176e-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 is inside our result object `pca` that we just calculated:

```r
attributes(pca)
```

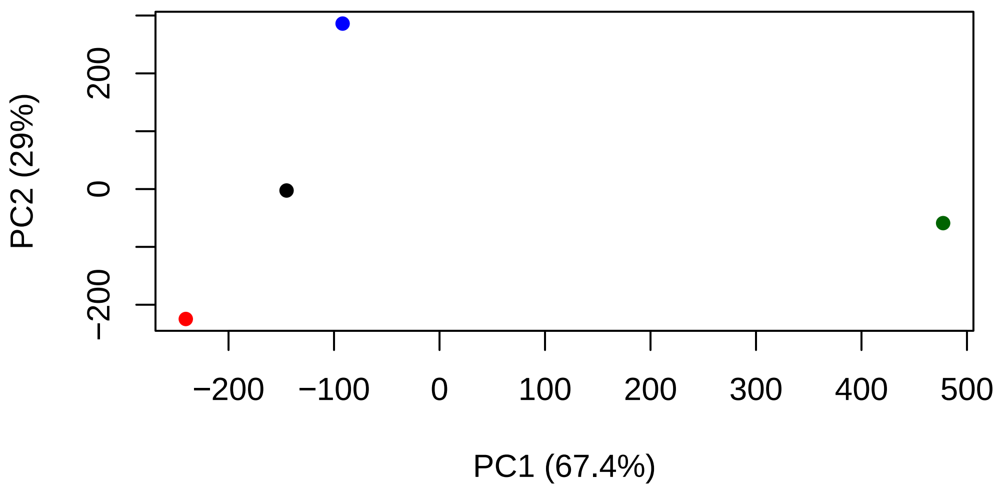
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

```r
pca$x
```

```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -4.894696e-14
Wales     -240.52915 -224.646925 -56.475555  5.700024e-13
Scotland   -91.86934  286.081786 -44.415495 -7.460785e-13
N.Ireland  477.39164  -58.901862  -4.877895  2.321303e-13
```

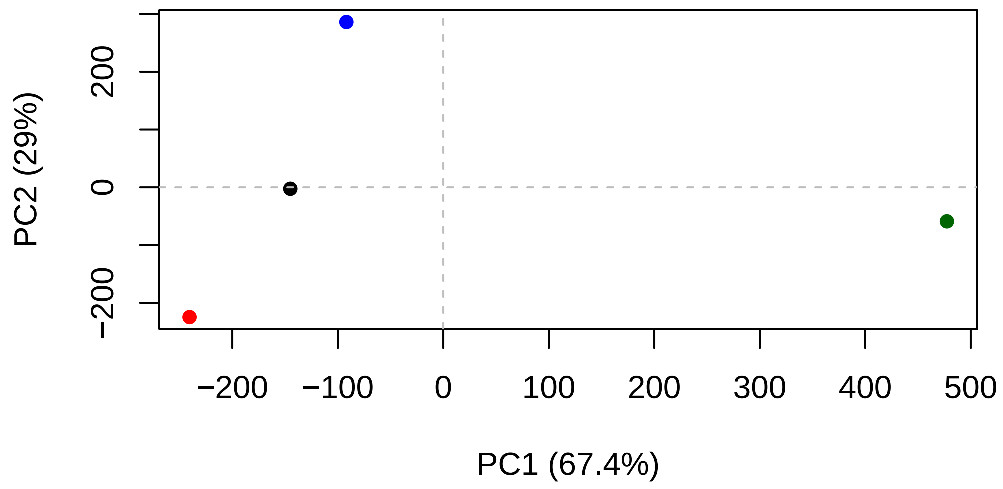
To make our main result figure, called a "PC plot" (or "score plot", "ordination plot" or "PC1 VS PC2 plot").

```r
plot(pca$x[,1], pca$x[,2], col=c("black", "red", "blue", "darkgreen"),
pch=16, xlab="PC1 (67.4%)", ylab="PC2 (29%)")
```



To make our main result figure, called

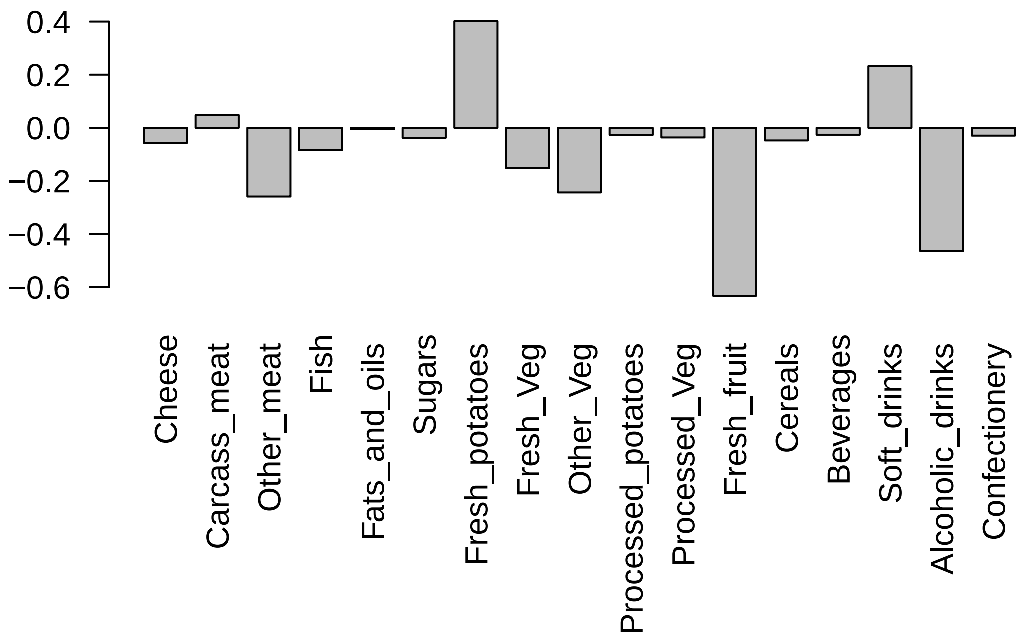
```r
plot(pca$x[,1], pca$x[,2],
col=c("black", "red", "blue", "darkgreen"), pch=16, xlab= "PC1 (67.4%)", ylab="PC2 (29%)")

abline(h=0, col="gray", lty=2)
abline(v=0, col="gray", lty=2)
```



Variable Loadings plot

Can give us insight on how the original variables (in this case the foods) contribute to our new PC axis.

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



0.4
0.2
0.0
−0.2
−0.4
−0.6
Cheese
Carcass_meat
Other_meat
Fish
Fats_and_oils
Sugars
Fresh_potatoes
Fresh_Veg
Other_Veg
Processed_potatoes
Processed_Veg
Fresh_fruit
Cereals
Beverages
Soft_drinks
Alcoholic_drinks
Confectionery