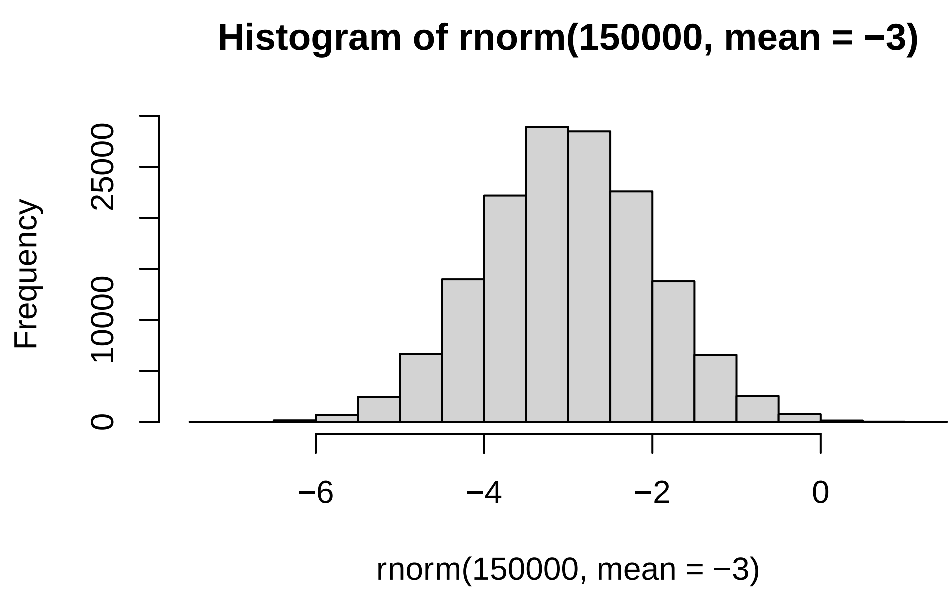
Class 07

Josie (A11433761)

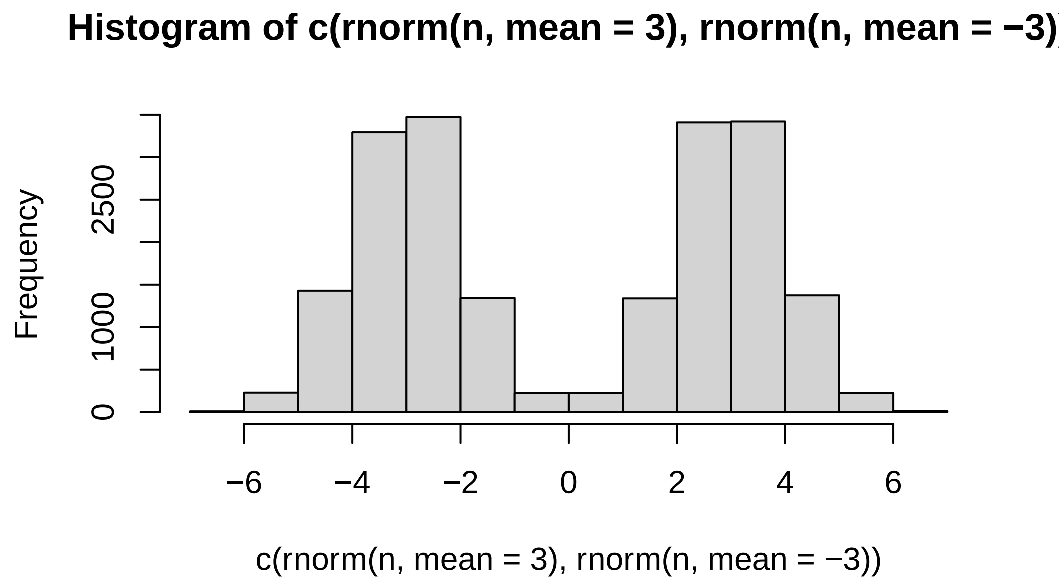
Before we get into clustering methods let's make some sample data to cluster where we know what the answer should be.

To help with this I will use the `rnorm()` function.

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




```
n=10000
hist(c(rnorm(n,mean=3),rnorm (n,mean=-3)))
```



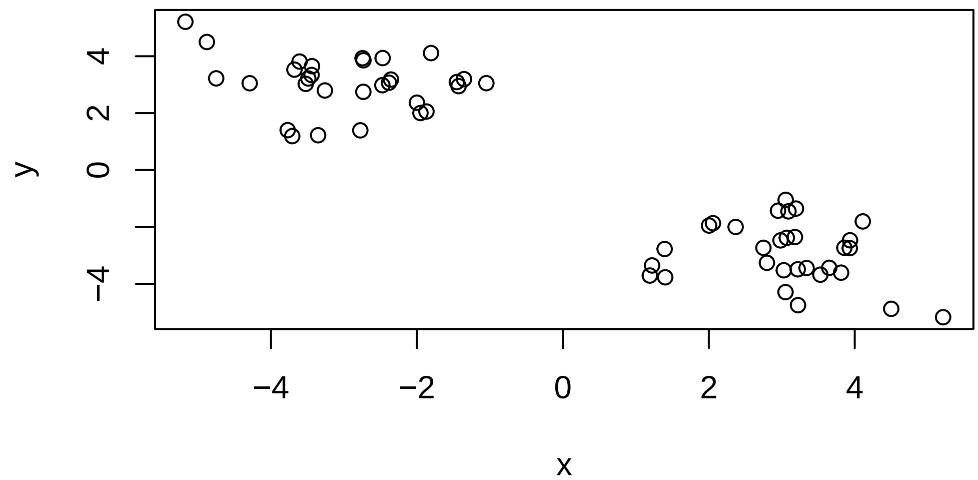

```r
n=30
x<-c(rnorm(n,mean=3),rnorm (n,mean=-3))
y<-rev(x)
z<-cbind(x,y)
z
```

```
             x         y
 [1,] 4.110670 -1.806963
 [2,] 3.813138 -3.608678
 [3,] 3.218754 -3.489699
 [4,] 1.222338 -3.354657
 [5,] 1.191884 -3.709332
 [6,] 2.748526 -2.735017
 [7,] 3.053610 -1.048753
 [8,] 3.180411 -2.356207
 [9,] 3.935262 -2.470225
[10,] 3.650484 -3.438469
[11,] 4.500134 -4.879594
[12,] 2.982635 -2.474210
[13,] 1.402237 -3.771985
[14,] 2.057058 -1.869827
[15,] 3.528990 -3.679315
[16,] 2.796476 -3.261870
```

```
[17,]  1.394751 -2.776697
[18,]  5.211226 -5.173876
[19,]  3.857948 -2.733254
[20,]  3.069064 -2.383431
[21,]  3.339916 -3.444614
[22,]  3.932075 -2.745028
[23,]  2.948115 -1.429715
[24,]  2.367746 -2.000807
[25,]  3.051122 -4.292615
[26,]  3.027701 -3.524213
[27,]  3.222355 -4.751351
[28,]  3.194708 -1.355170
[29,]  2.003491 -1.951833
[30,]  3.091731 -1.453643
[31,] -1.453643  3.091731
[32,] -1.951833  2.003491
[33,] -1.355170  3.194708
[34,] -4.751351  3.222355
[35,] -3.524213  3.027701
[36,] -4.292615  3.051122
[37,] -2.000807  2.367746
[38,] -1.429715  2.948115
[39,] -2.745028  3.932075
[40,] -3.444614  3.339916
[41,] -2.383431  3.069064
[42,] -2.733254  3.857948
[43,] -5.173876  5.211226
[44,] -2.776697  1.394751
[45,] -3.261870  2.796476
[46,] -3.679315  3.528990
[47,] -1.869827  2.057058
[48,] -3.771985  1.402237
[49,] -2.474210  2.982635
[50,] -4.879594  4.500134
[51,] -3.438469  3.650484
[52,] -2.470225  3.935262
[53,] -2.356207  3.180411
[54,] -1.048753  3.053610
[55,] -2.735017  2.748526
[56,] -3.709332  1.191884
[57,] -3.354657  1.222338
[58,] -3.489699  3.218754
[59,] -3.608678  3.813138
```

```
[60,] -1.806963  4.110670
```

```r
plot(z)
```



K-means clustering

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

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

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

Cluster means:
          x         y
1 -2.932368  3.036819
2  3.036819 -2.932368

Clustering vector:
 [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
```

```
Within cluster sum of squares by cluster:
[1] 60.09283 60.09283
 (between_SS / total_SS =  89.9 %)

Available components:

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

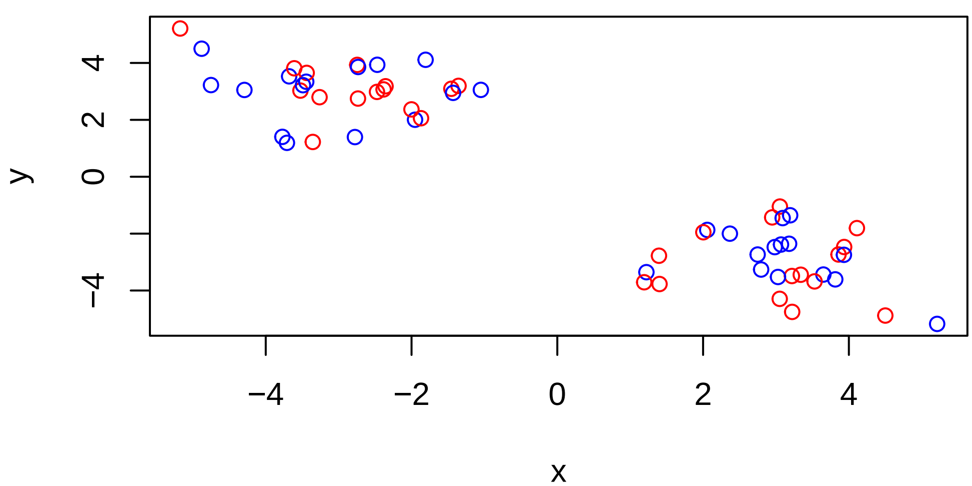
```
km$centers
```

```
          x         y
1 -2.932368  3.036819
2  3.036819 -2.932368
```

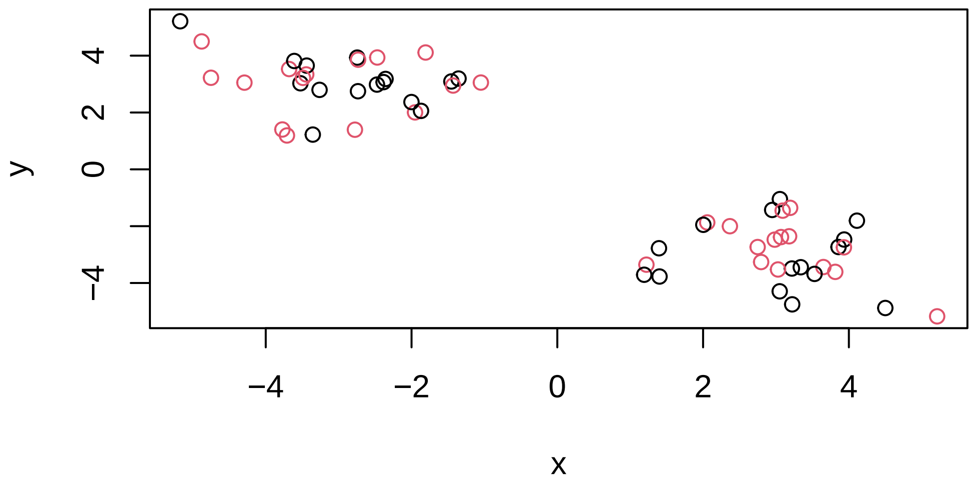
```
km$cluster
```

```
 [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
```

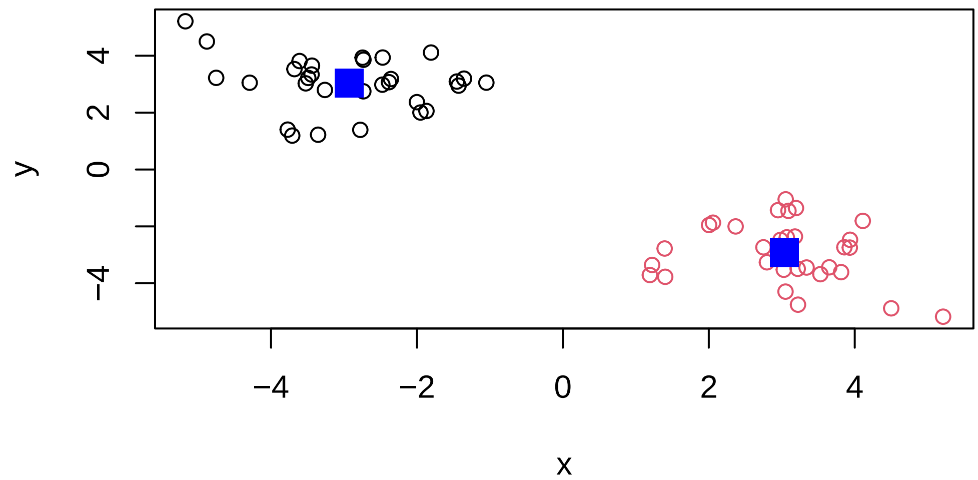
```
plot(z,col=c("red","blue"))
```



```
plot(z, col=c(1,2))
```

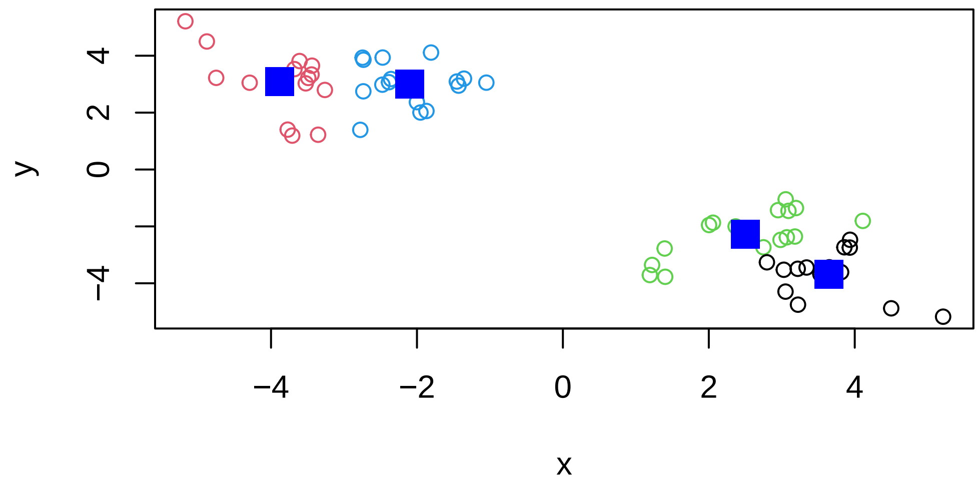



```
plot(z, col=km$cluster) #plot with clustering results and cluster centers
points(km$centers, col="blue",pch=15,cex=2) #color the center and make it a square center (p
```



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

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



Hierarchical Clustering

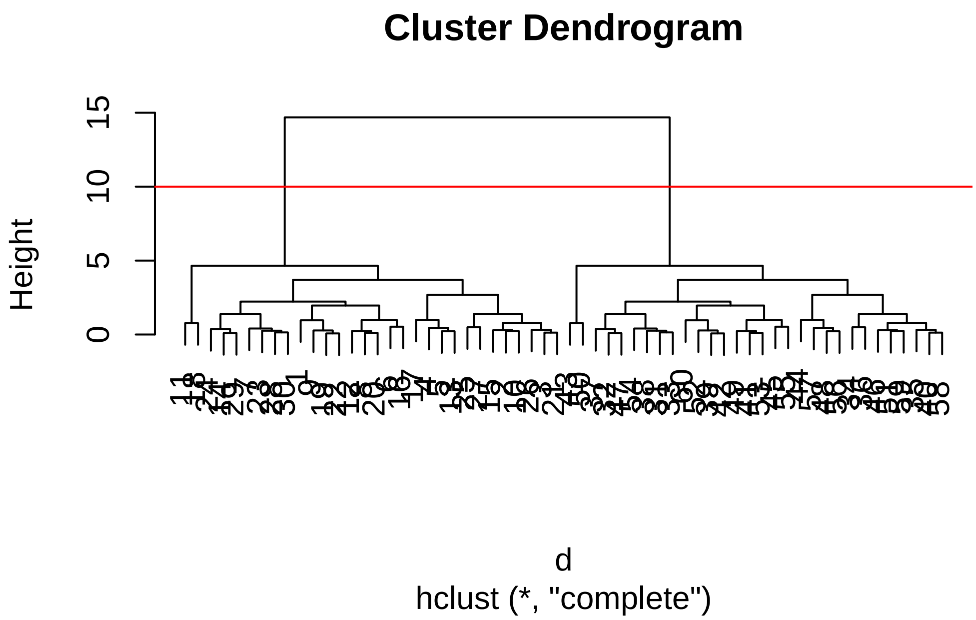
The main function for hierarchical clustering is base R is called `hclust()`. Unlike `kmeans()` I cannot just pass in my data as input. I first need a distance matrix from my data.

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

```
Call:
hclust(d = d)

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

hclust plot method

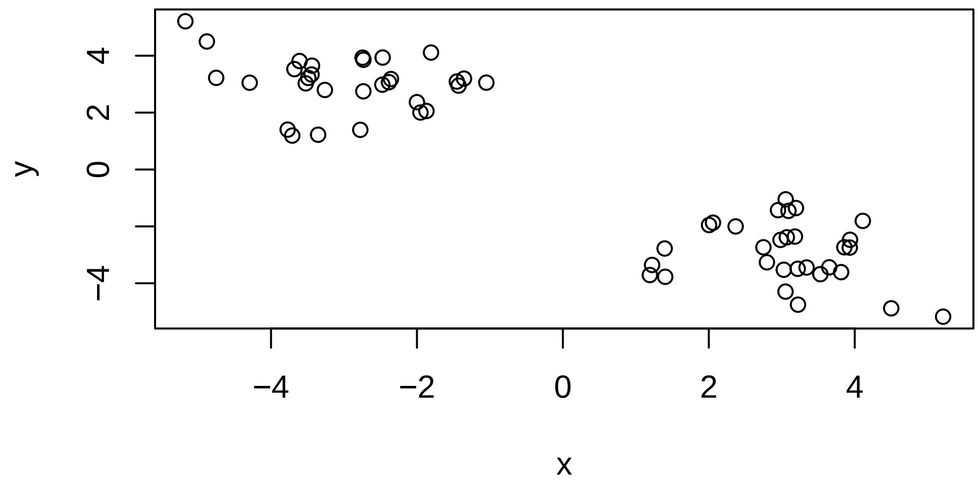
```
plot(hc)
abline(h=10, col="red")
```



To get my clustering results, I can "cut" my tree at a given height. To do this, I will use the `cutree`.

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

```
plot(z,hc$grps)
```



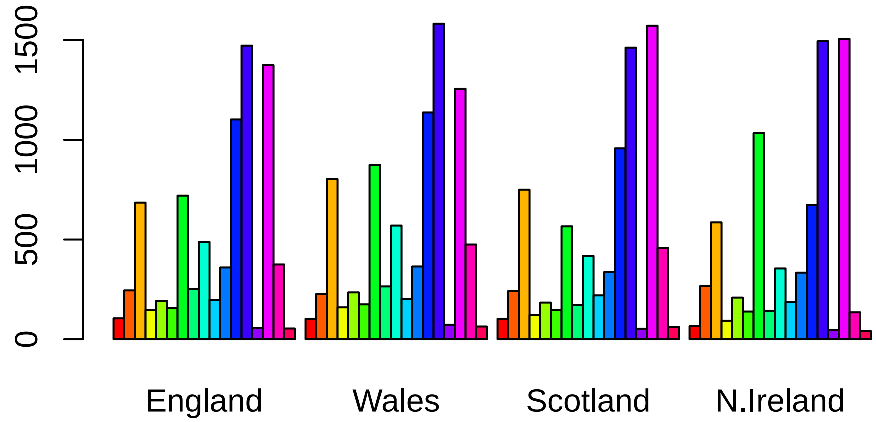
Principle Component Analysis

PCA of UK food data

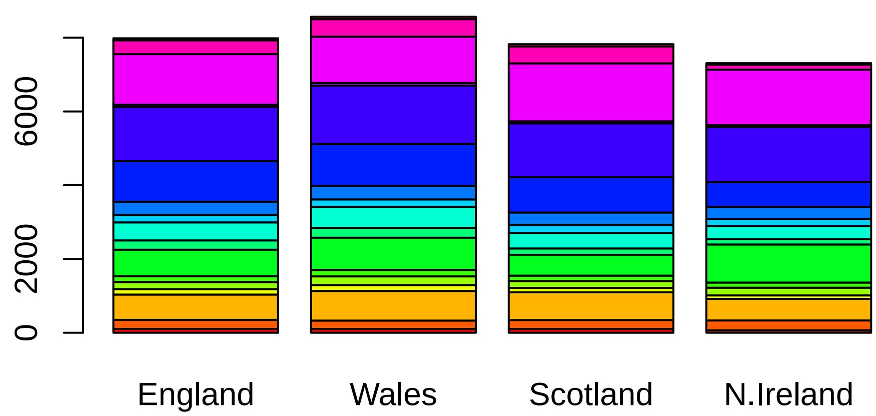
```
url<-"http://tinyurl.com/UK-foods"
x<-read.csv(url,row.names=1)
head(x)
```

```
               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
```

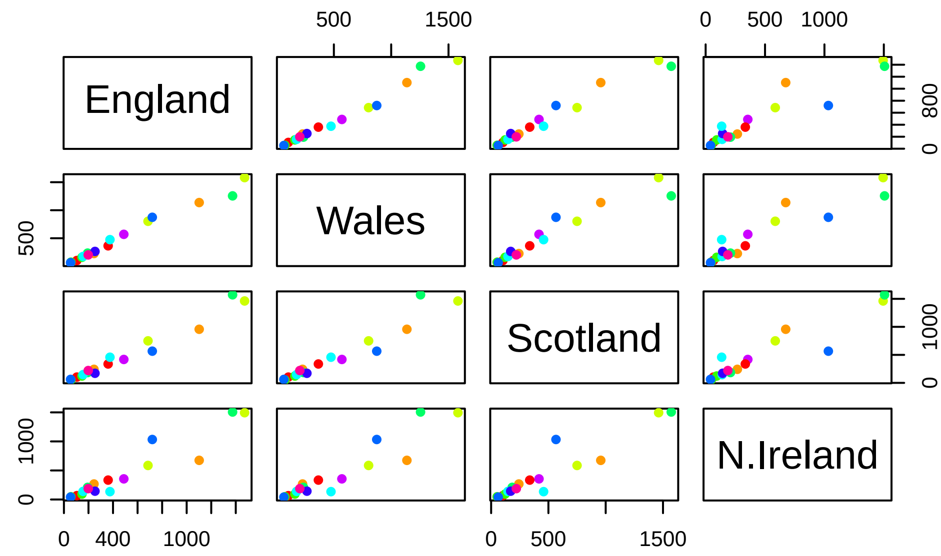
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```




```
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()`

```
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 our result object `pca`

```
attributes(pca)
```

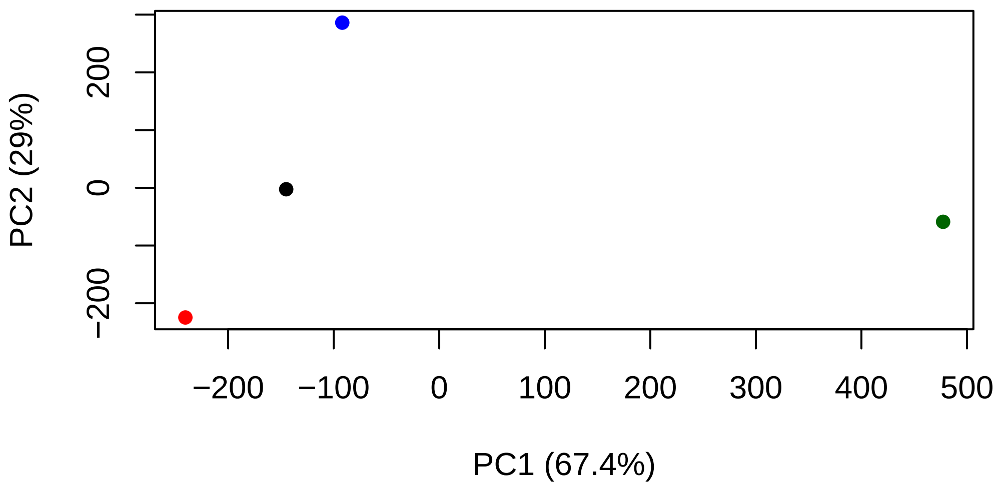
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

```
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
```

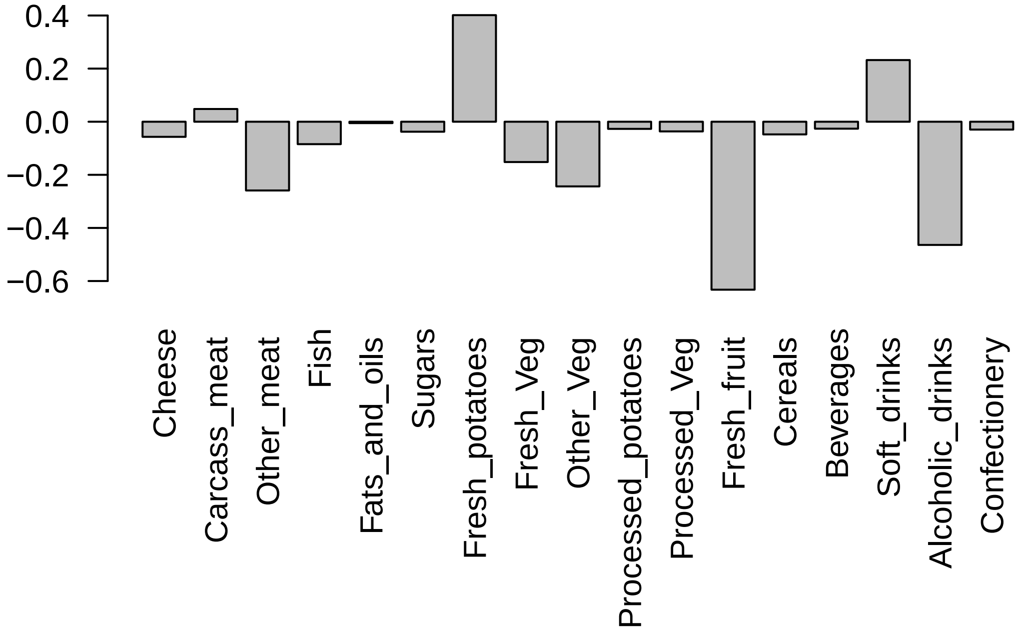
To make our main result figure, called a PC plot (or score plot, ordination plot, PC1 vs.PC2):

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



Variable Loadings Plot: 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 )
```




```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
```