Class 07 : Machine Learning 1

Vidisha Marwaha (PID: A16677246)

Today we will start our multi-part exploration of some key machine learning methods. We will begin with clustering - finding groupings in data, and then dimensional reduction.
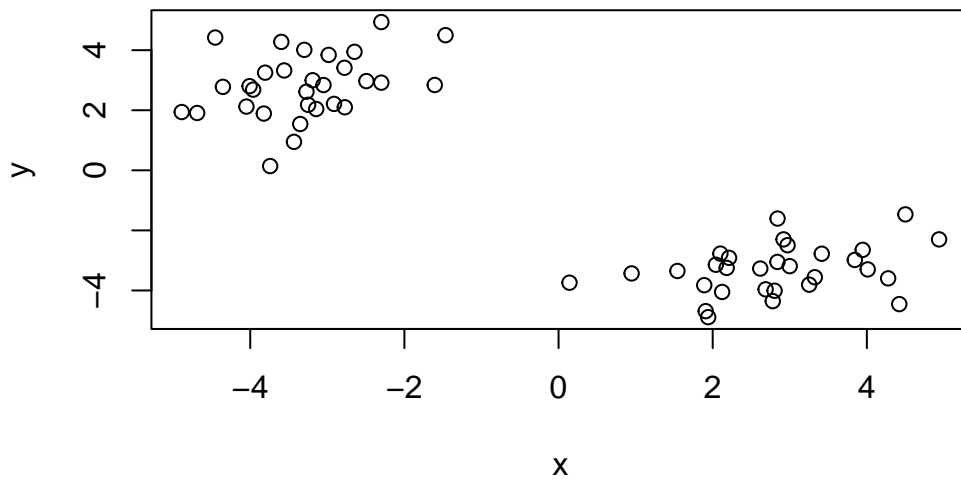
Clustering

Let's start with "k-means" clustering. The main function in base R for this `kmeans()`

```
# Make up some data
hist(rnorm(100000, mean=3))
```

Histogram of rnorm(1e+05, mean = 3)

rnorm(1e+05, mean = 3)
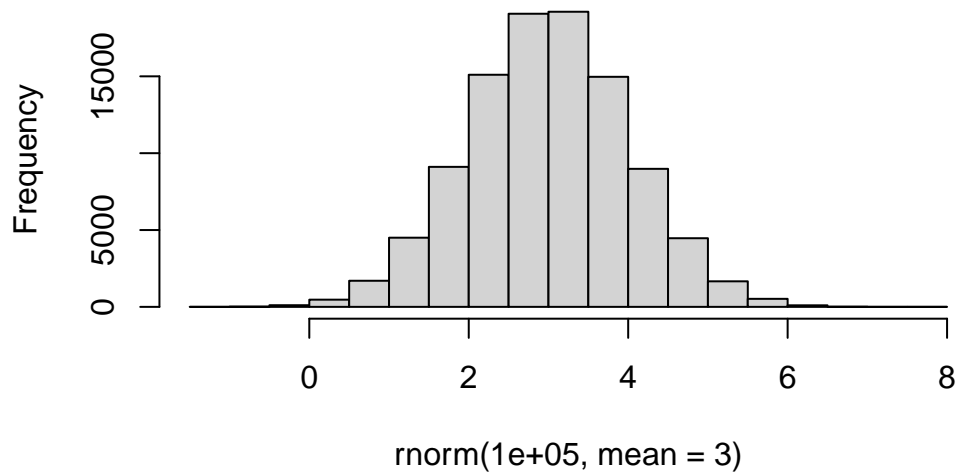
```
tmp <- c(rnorm(30, -3), rnorm(30, +3))
x <- cbind(x=tmp, y=rev(tmp))
plot(x)
```


Now lets try out kmeans()

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

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

Cluster means:
```
          x          y
1 -3.306016  2.813668
2  2.813668 -3.306016
```

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] 53.32278 53.32278
 (between_SS / total_SS = 91.3 %)

Available components:

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

```r
attributes(km)
```

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

$class
[1] "kmeans"
```

Q. How many points in each cluster?

```r
km$size
```

```
[1] 30 30
```

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

```r
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 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 2
```

Q. What are centers/mean values of each cluster?

```r
km$centers
```

```
          x          y
1 -3.306016   2.813668
2  2.813668 -3.306016
```

Q. Make a plot of your data showing your clustering results (groupings/clusters and cluster centers).

```r
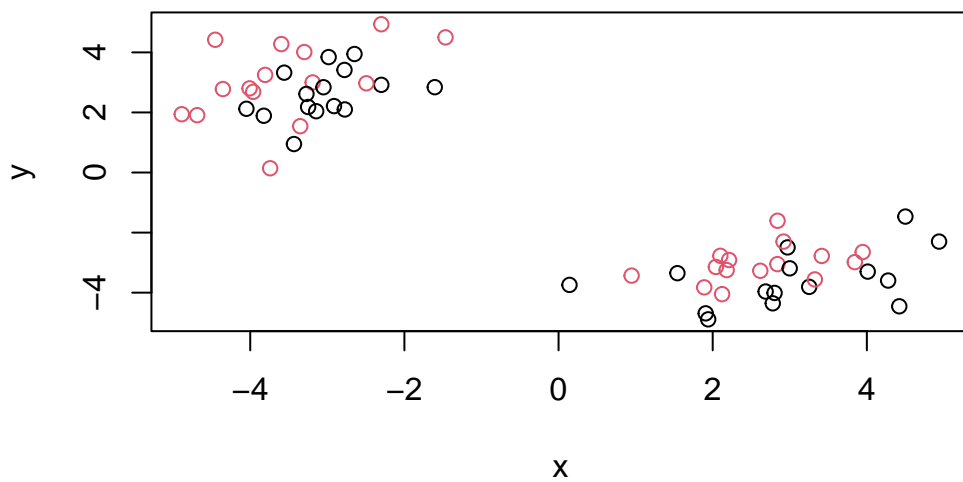plot(x, col=c("red", "blue"))
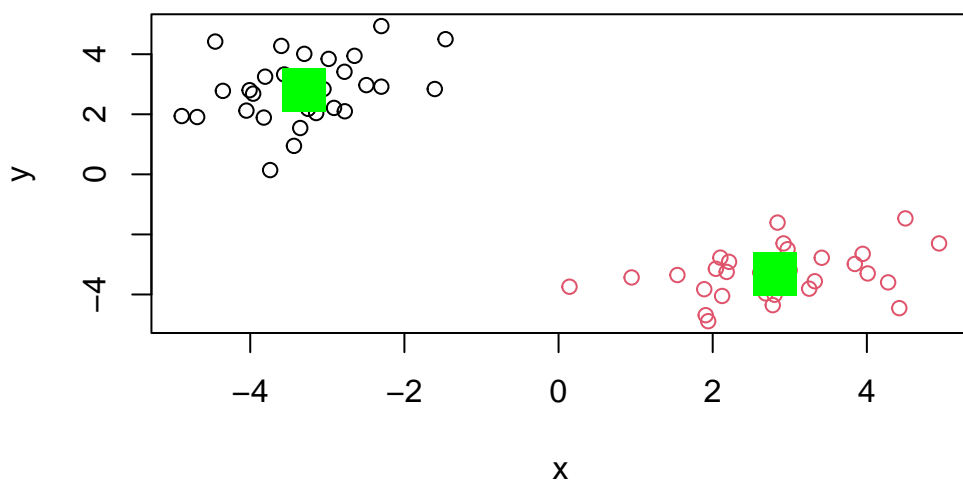```
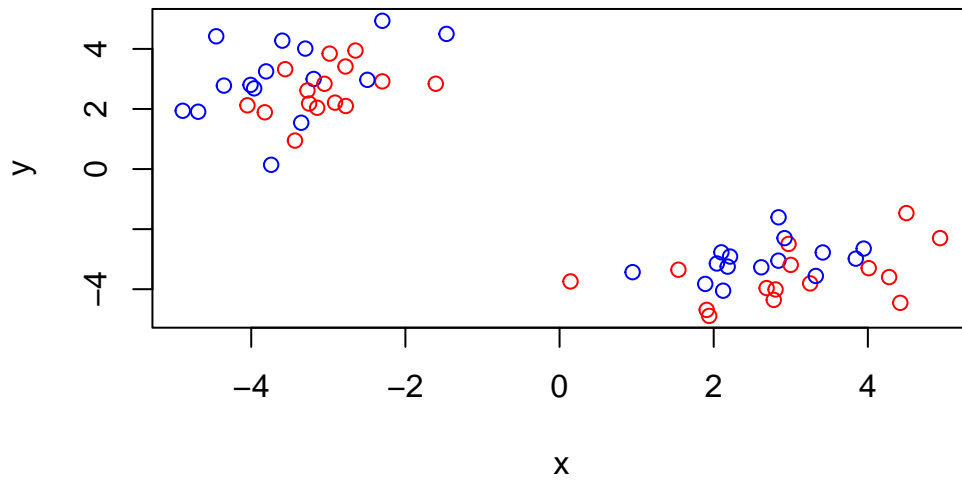


```r
c(1:5)+c(100,1)
```

Warning in c(1:5) + c(100, 1): longer object length is not a multiple of
shorter object length

```
[1] 101   3 103   5 105
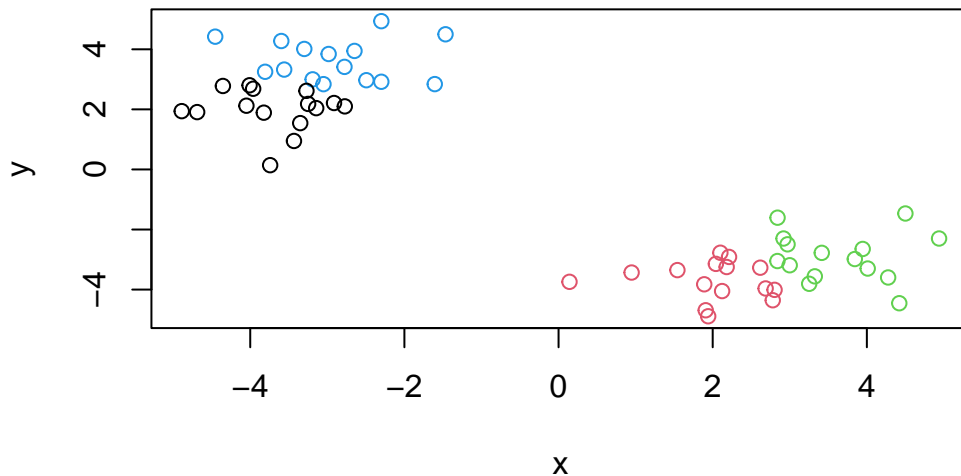```

```r
plot(x, col=c(1,2))
```

```
plot(x, col=km$cluster)
points(km$centers, col="green", pch=15, cex=3)
```


5

Q. Run `kmeans()` again and cluster in 4 groups and plot the results.

```
km4 <- kmeans(x, centers = 4)
plot(x, col=km4$cluster)
```


Heirarchical Clustering

This form of clustering aims to reveal the structure in your data by progressively grouping points into a ever smaller number of clusters.

The main function in base R for this is called `hclust()`. This function does not take our input data directly but wants a "distance matrix" that details how (dis)similar all our input points are to each other.

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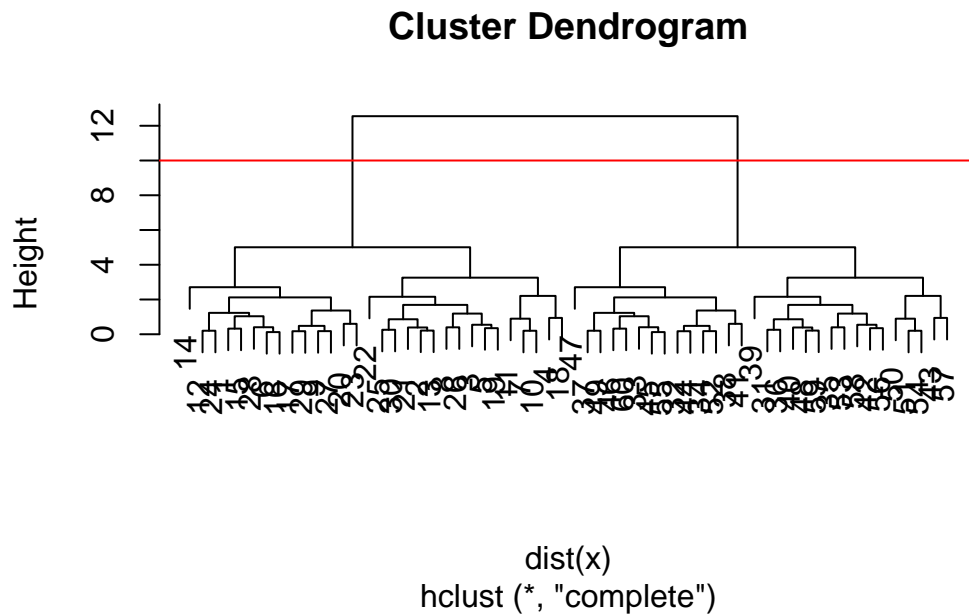hc <- hclust(dist(x))
hc
```

```
Call:
hclust(d = dist(x))
```

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

The print out is not very useful (unlike that from kmeans) but there is a useful `plot()` method.

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

Cluster Dendrogram

dist(x)
hclust (*, "complete")

To get my main result (my cluster membership vector) I need to "cut" my tree using the function `cutree()`

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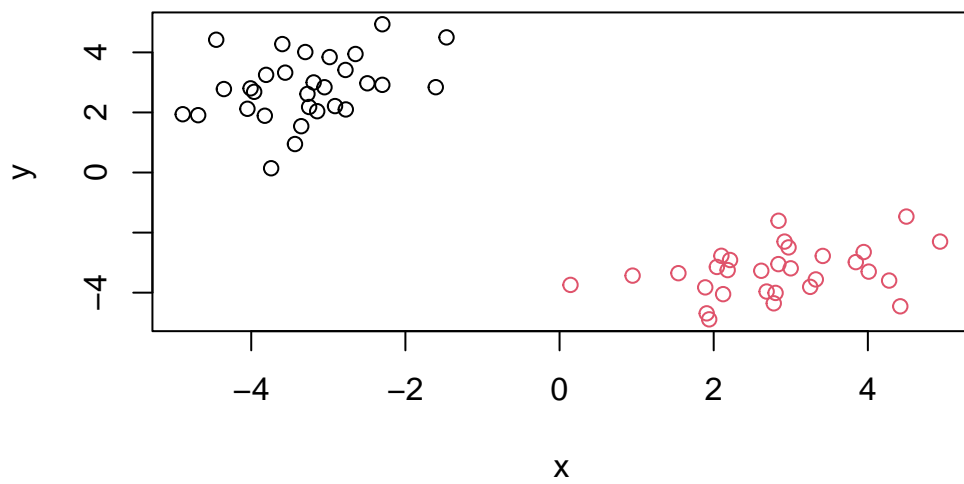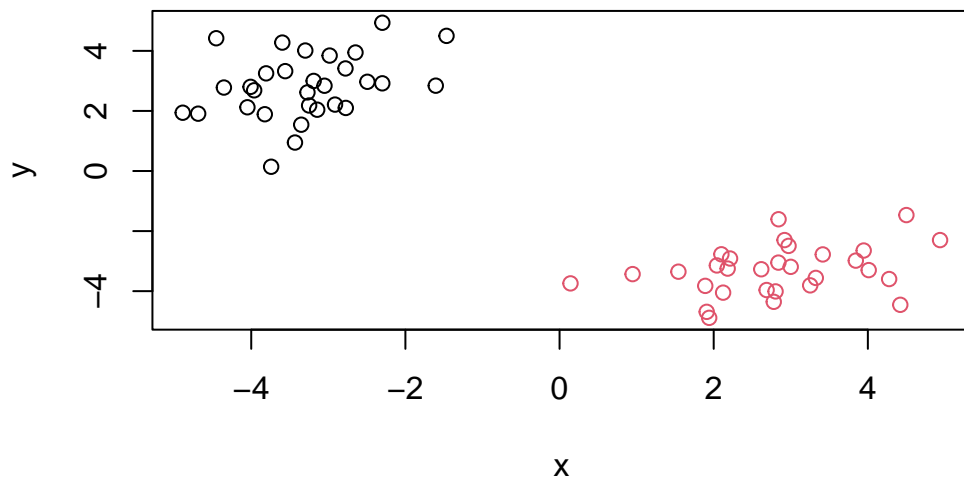grps <- cutree(hc, h=10)
grps
```

```
 [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 1 1 1 1 1 1 1 1 2 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(x, col=grps)
```

```
plot(x, col=cutree(hc, h=6))
```

Principal Component Analysis (PCA)

Read data about eating trends in UK ..

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

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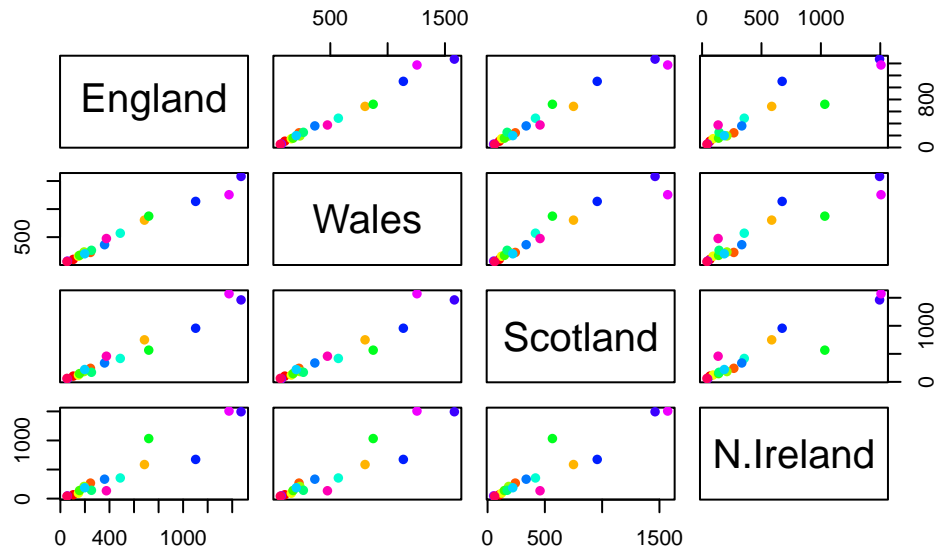barplot(as.matrix(x), col=rainbow(nrow(x)))
```
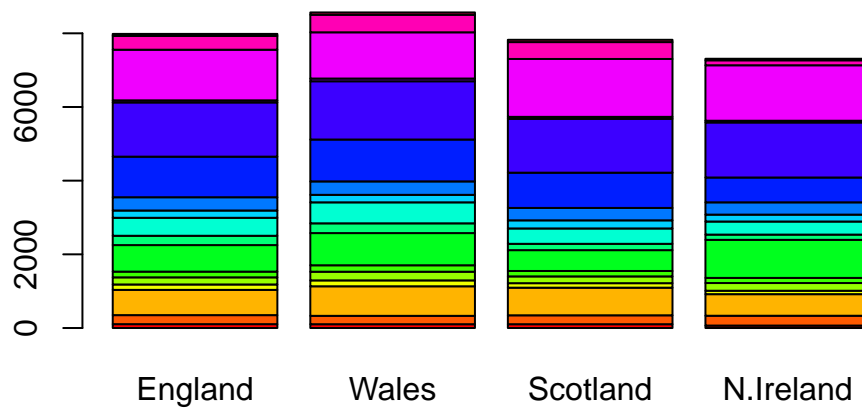
```r
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```


10
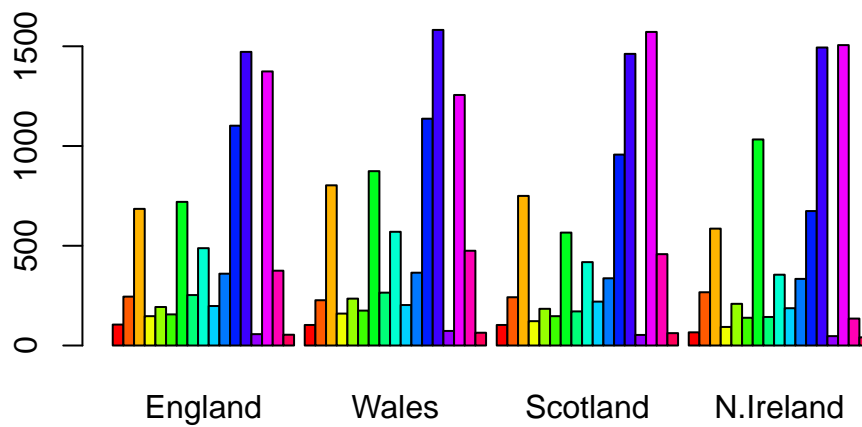
The so-called "pairs" plot can be useful for small datasets.

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



```
#rainbow (nrow(x))
```

So the pairs is useful for small data sets but it can be lots of work to interpret and gets intractable for larger data sets

So PCA to the rescue...

The main function to do PCA in base R is called `prcomp()`. This function wants the transpose of our data in this case.

```
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
```

11

```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 -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
```

A major PCA result viz is called a "PCA plot" (aka a score plot, biplot, PC1 vs PC2 plot, ordination plot)

```r
mycols <- c("orange", "red", "blue","darkgreen")
plot(pca$x[,1], pca$x[,2], col=mycols, pch=16,
     xlab="PC2", ylab="PC1")
abline(h=0, col="gray")
abline(v=0, col="gray")
```
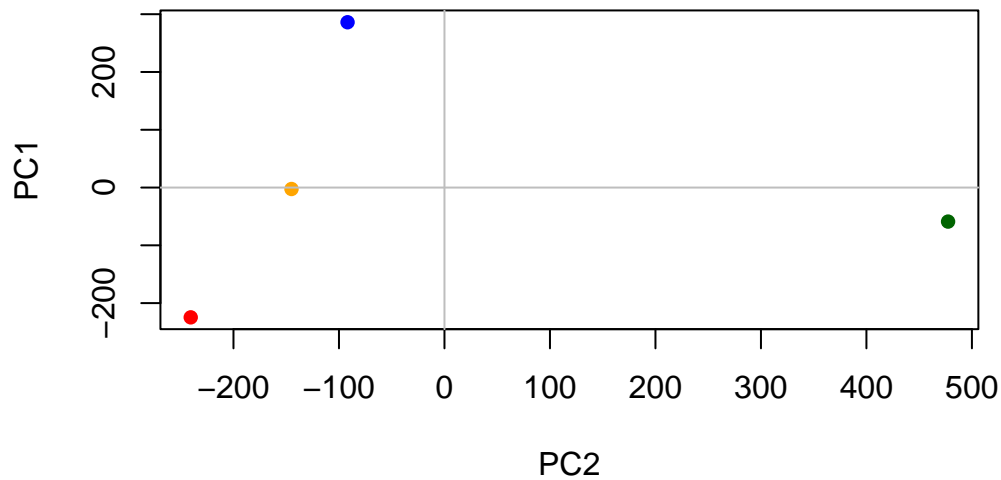
Another important output from PCA is called the "loadings" vector of the "rotation" component - this tells us how much the original variables (the foods in this case) contribute to the new PCs.

```
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

13

```
Alcoholic_drinks    -0.463968168  0.113536523 -0.49858320 -0.273126013
Confectionery       -0.029650201  0.005949921 -0.05232164  0.001890737
```

PCA is a super useful method for gaining insight into high dimensional data that is difficult to examine in other ways.

Questions

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

```
nrow(x)
```

```
[1] 17
```

```
ncol(x)
```

```
[1] 4
```

```
## preview first 6 rows
head(x,6)
```

```
              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
```

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

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

```
  dim(x)
```

```
[1] 17  3
```

```
  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. 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?