Class 7: Machine learning 1

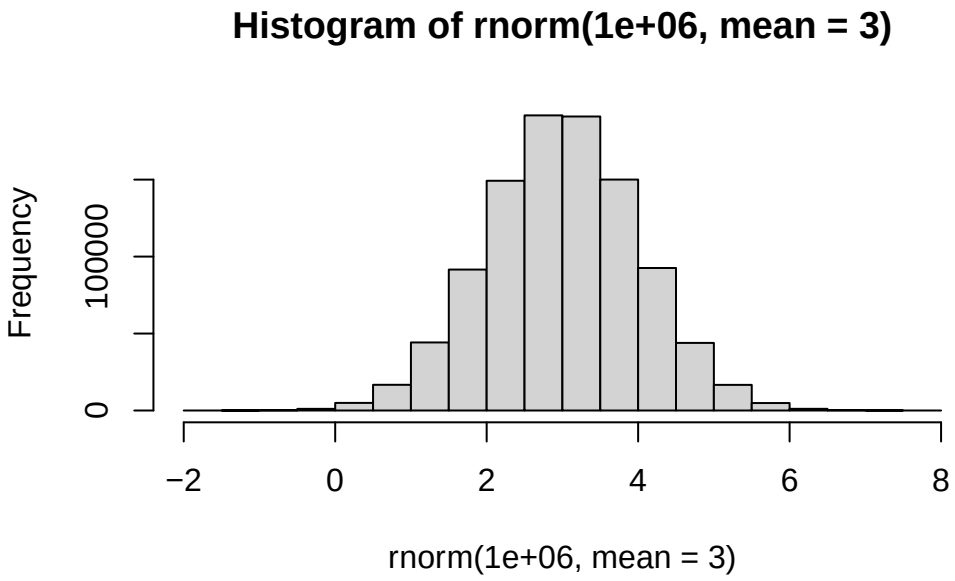
Justin Lu (A16318305)

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 dimensionality reduction.

##Clustering

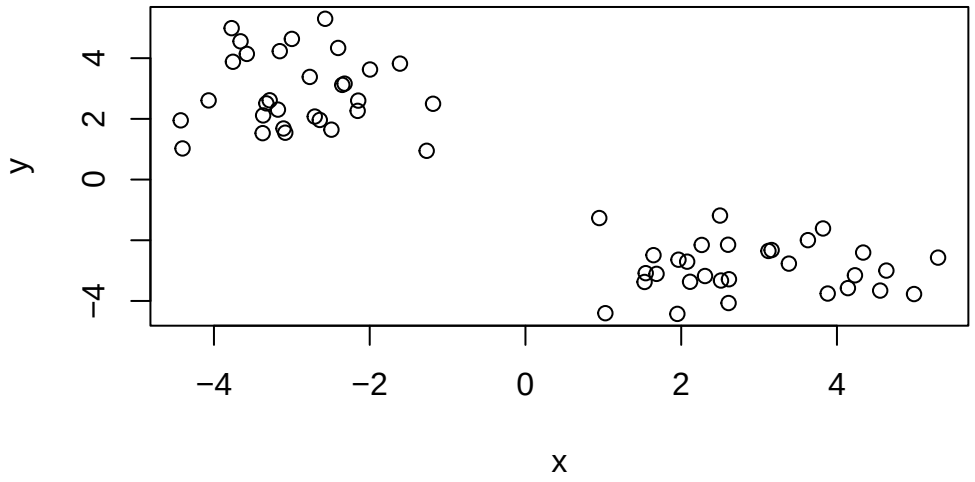
Let's start with "k-means" clustering. The main function in base R for this `kmeans()`

```
#make up some rnadom data
hist(rnorm(1000000, mean = 3))
```




```
tmp <- c(rnorm(30,-3), rnorm(30,+3))
x <- cbind (x=tmp, y=rev(tmp))
```

```r
plot(x)
```



Now let's try out `kmeans()`

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

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

Cluster means:
          x         y
1  2.901968 -2.905577
2 -2.905577  2.901968

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 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] 62.38694 62.38694
 (between_SS / total_SS =  89.0 %)
```

```
Available components:

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

Q. How many points in each cluser

```
km$size
```

```
[1] 30 30
```

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

```
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 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
```

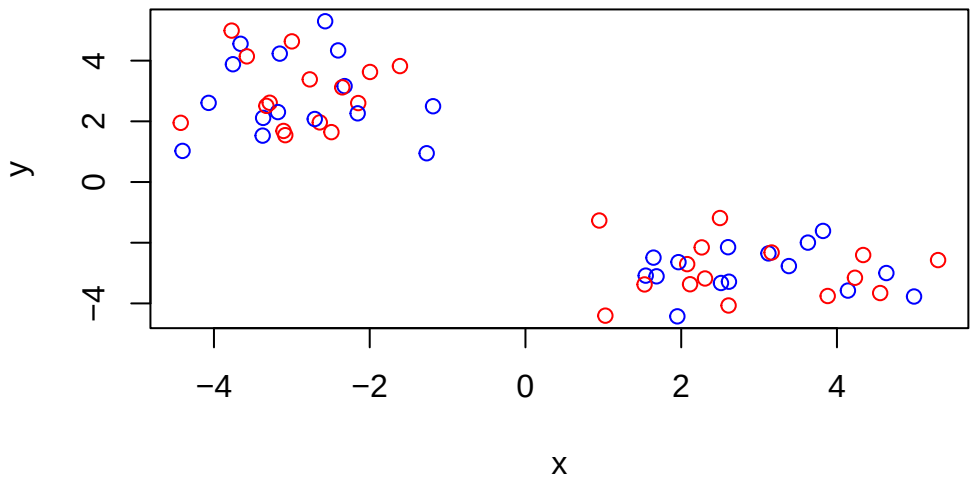
Q. What are the centers/mean values of each cluster

```
km$centers
```

```
          x         y
1  2.901968 -2.905577
2 -2.905577  2.901968
```

Q. Make a plot of your data showing your clustering results

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

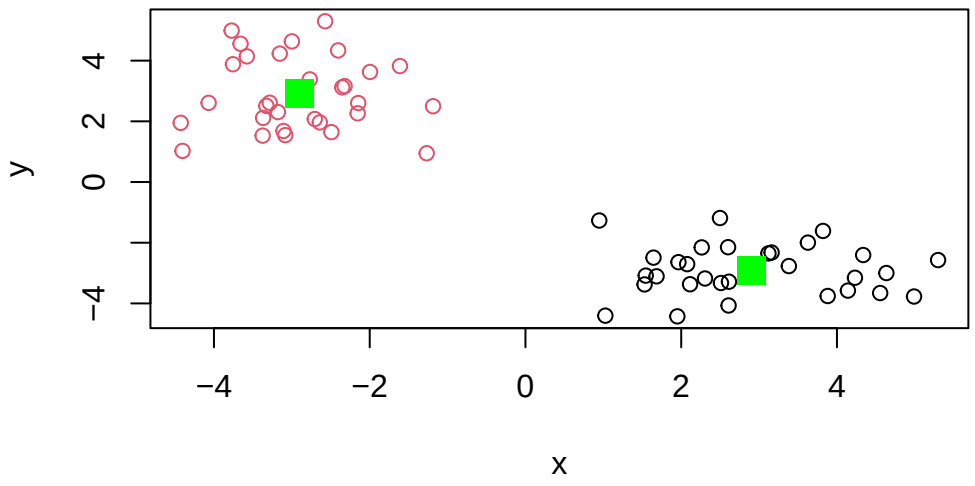



```
c(1:5) + c(100,1)
```

```
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
```

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



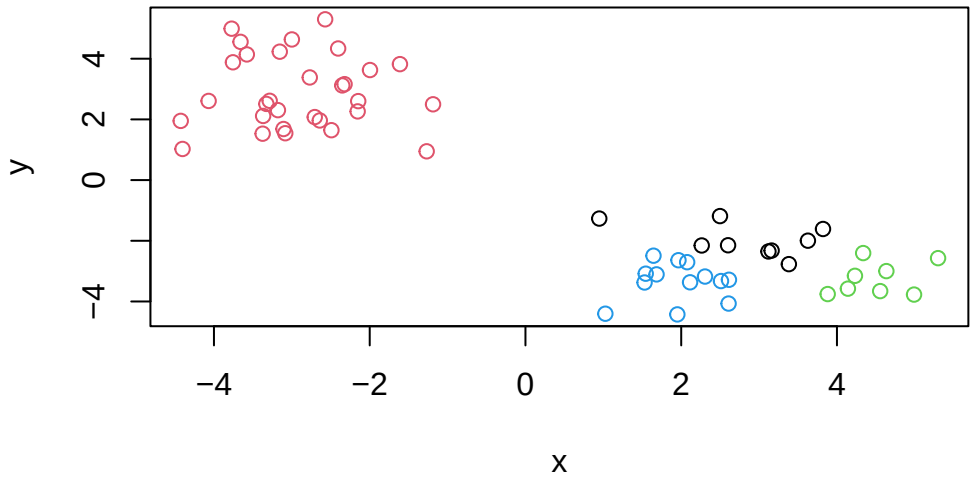
Q. Run `kmeans()` again and cluster in 4 groups and plot the results

```
km4 <- kmeans(x, centers = 4)
km4
```

```
K-means clustering with 4 clusters of sizes 9, 30, 8, 13

Cluster means:
          x         y
1  2.824688 -1.978871
2 -2.905577  2.901968
3  4.508380 -3.237094
4  1.966907 -3.343132

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 1 4 4 1 4 1 4 3
[39] 4 4 3 1 3 3 1 4 3 1 3 4 1 1 1 4 4 3 4 4 3 4

Within cluster sum of squares by cluster:
[1]  8.351393 62.386939  3.556259  7.317457
 (between_SS / total_SS =  92.8 %)
```

```
Available components:

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

```
plot(x, col = km4$cluster)
```



##Hierarchial Clustering

This form of clustering aims to reveal the structure in your data by progressively grouping points into an 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.

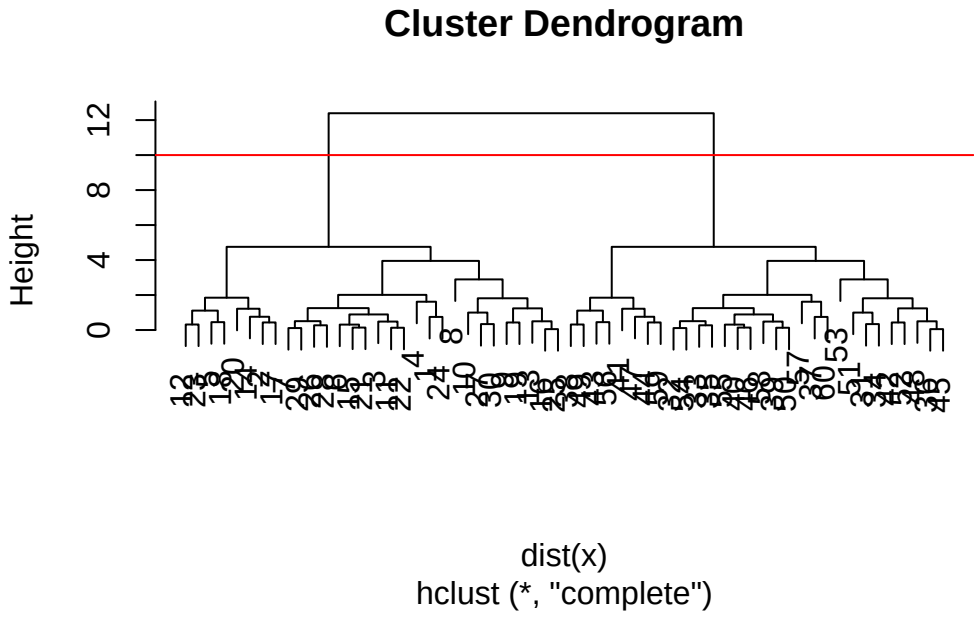
```
hc <- hclust(dist(x))
hc
```

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

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

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

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

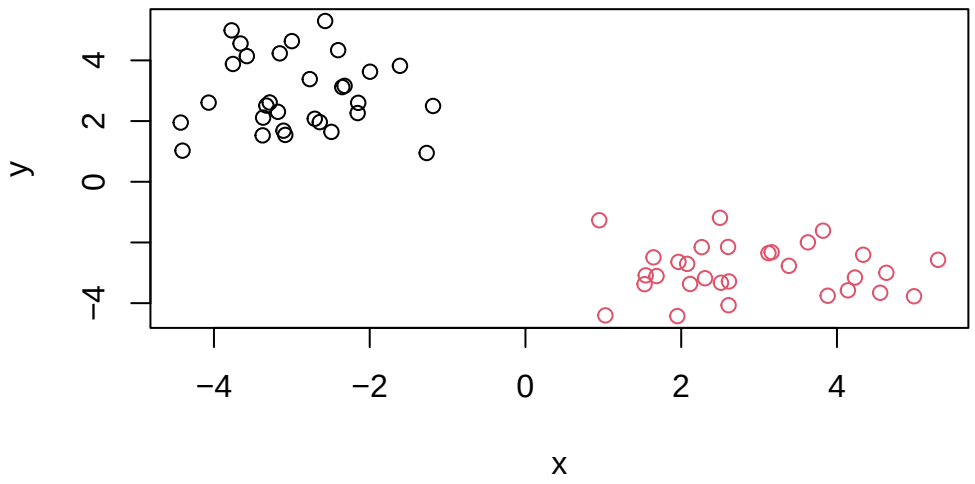


To get my main result (my cluster membership vector), I need to cut my tree using the function `cutree()`

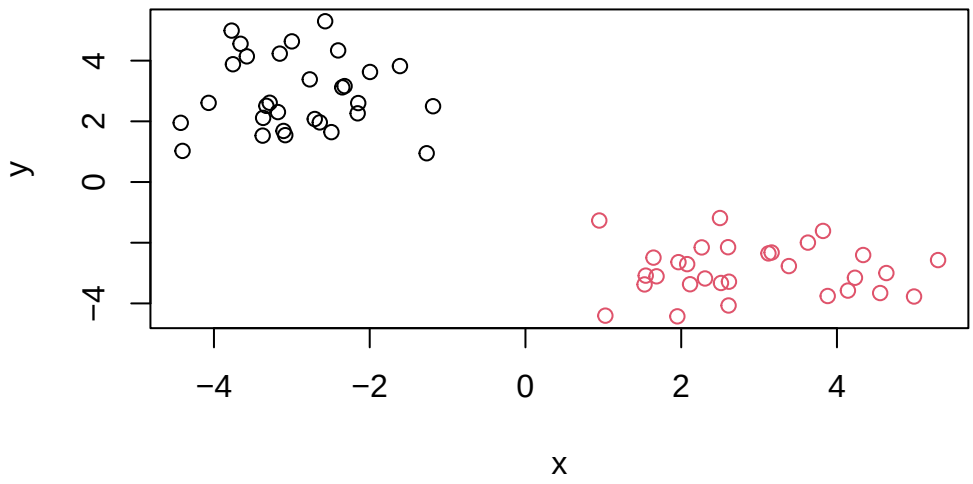
```
grps <- cutree(hc, h = 10)
grps
```

```
 [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 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
```

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




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



#Principal Component Analysis (PCA)

The goal of PCA is to reduce the dimensionality of. adataset down to some smaller subset of new vairables (called PCs) that are a useful basis for further analysis, like visualization, clustering, etc.

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

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

```
[1] 5
```

```
nrow(x)
```

```
[1] 17
```

```
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
```

Using the nrow() and ncol() functions, I found that there were 17 rows and 5 columns.

```
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
```

#Another method of removing the first column

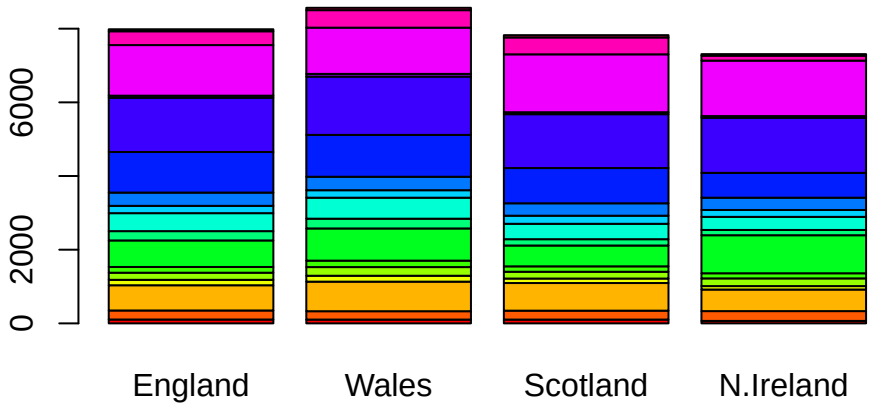
```
y <- read.csv(url, row.names=1)
head(y)
```

```
              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?

I preferred the second one more because it required less steps and code. I could just remove the column right from the beginning, which seems much more convenient compared to having to remove it after reading the URL file and having to adjust it.

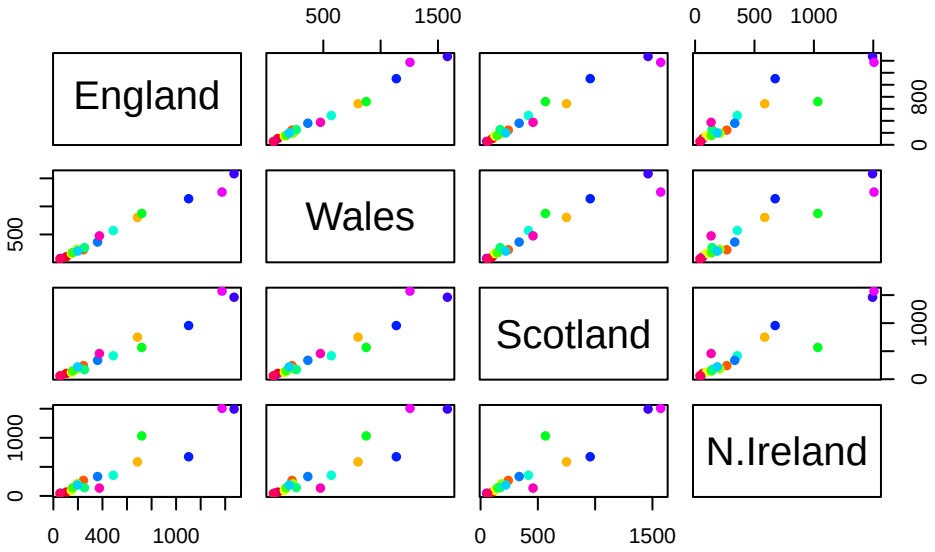
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



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

Changing the "beside" argument from T to F will generate the plot.

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



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?

It's rather hard to make sense of the figures because there are so many different colored points with no legend. However, the positioning of the name of the countries indicate which axis they lie on. For example, the first plot would be Wales vs England. If the plots lie on the diagonal, it means that any food picked in one country would have a similar value to the other country's value for that food as well. Basically, it's a metric of similarity.

So the pairs plot is useful for small datasets, but it can be lots of work to interpret and gets intractable for larger datasets.

PCA to the rescue!

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

```
#transpose by switching rows and columns
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
```

```
attributes(pca)
```

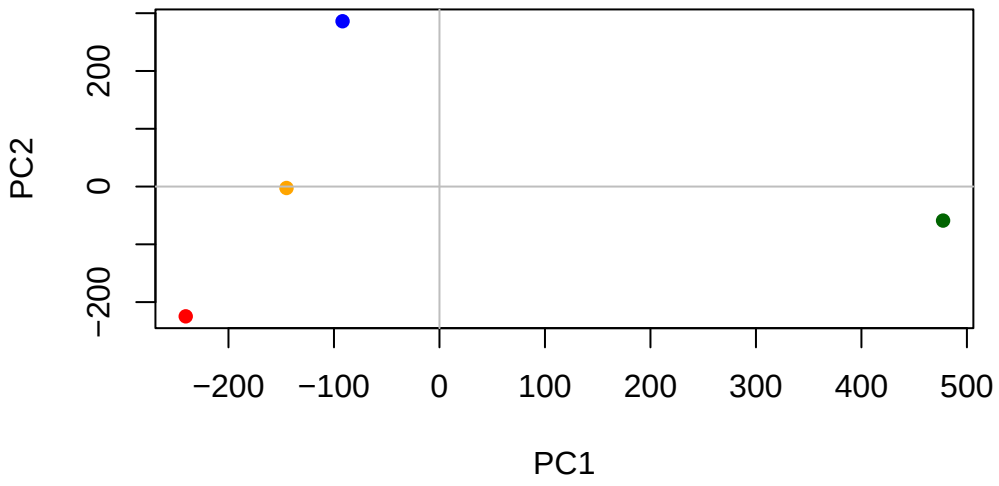
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$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
```

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

```
mycols <- c("orange", "red", "blue", "darkgreen")
plot(pca$x[,1], pca$x[,2], col = mycols, pch = 16, xlab = "PC1", ylab = "PC2")
abline(h = 0, col = "gray")
abline(v = 0, col = "gray")
```



Another important output from PCA is called the "loadings" vector or the "rotation" component - this tells us how much of the original variabes (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

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

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