Class 7: Machine Learning 1

Melanie Alonzo (A17375327)

Table of contents

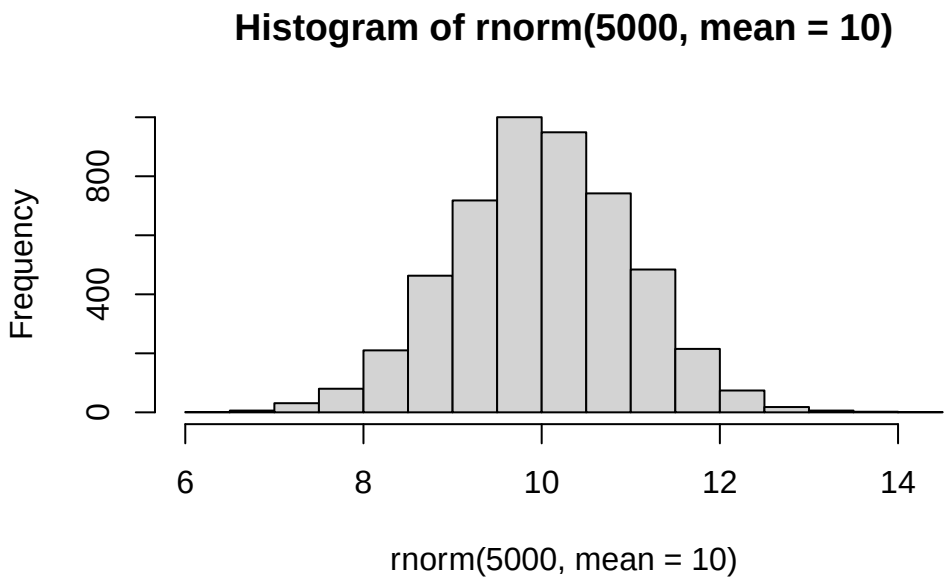
Background

Today we will begin our exploration of some important machine learning methods, namely **clustering** and **dimensionality reduction**.

Let's make up some input data for clustering where we know what the natural "clusters" are.

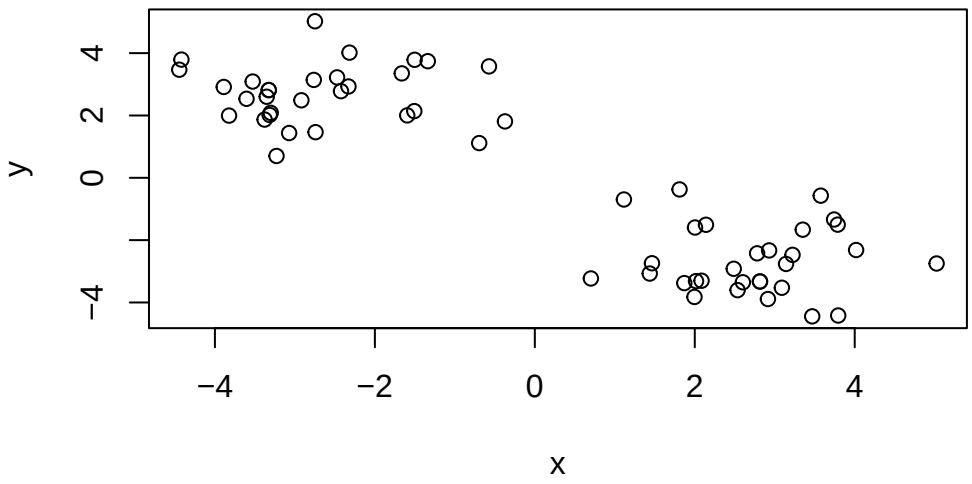
The function `rnorm()` can be useful here.

```
hist( rnorm(5000, mean=10) )
```



Q. Generate 30 random numbers centered at +3 and another 30 centered at -3

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



K-means clustering

The main function in "base R" for K-means clustering is called `kmeans()`:

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

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

Cluster means:
          x         y
1 -2.665887  2.690928
2  2.690928 -2.665887

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] 60.8784 60.8784
 (between_SS / total_SS =  87.6 %)
```

```
Available components:

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

Q. What component of the results object details the cluster size?

```
km$size
```

```
[1] 30 30
```

Q. WHat component of the results object details the cluster centers?

```
km$centers
```

```
          x         y
1 -2.665887  2.690928
2  2.690928 -2.665887
```

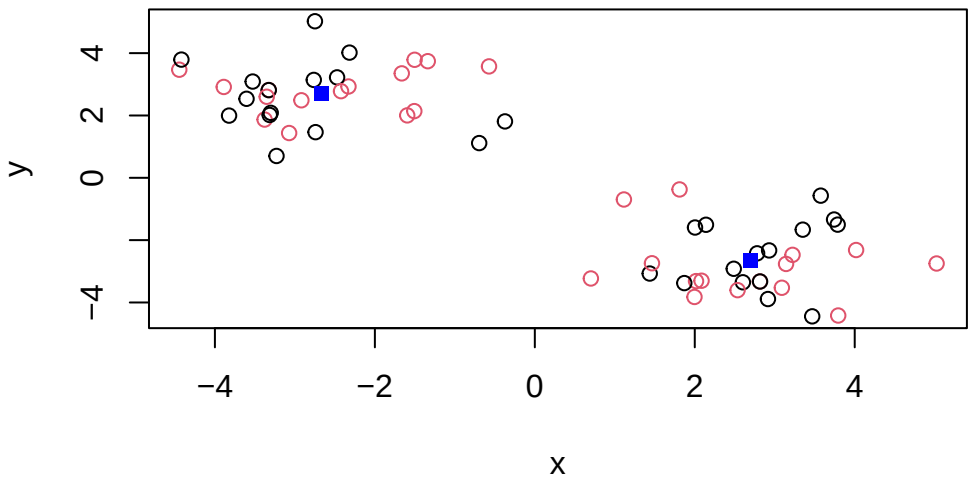
Q. What component of the results object details the cluster membership vector (i.e. our main result of which points lie in which cluster)?

```
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
```

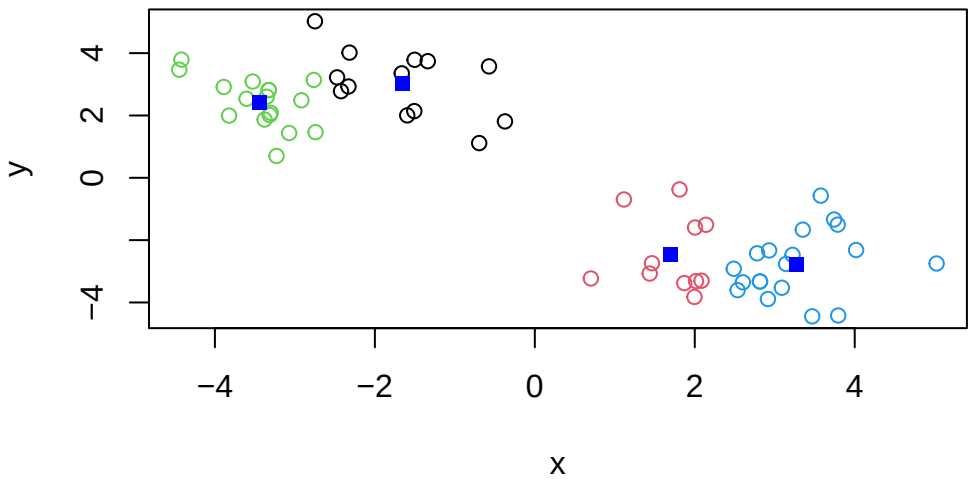
Q. Plot our clustering results with points colored by cluster and also add the cluster centers as new points colored blue?

```
plot(x, col= c(1,2))
points(km$centers, col="blue", pch=15)
```



Q. Run `kmeans()` again and this time produce 4 clusters (and call your result object `k4`) and make a results figure like above?

```
k4 <-kmeans(x,centers=4)
plot(x,col=k4$cluster)
points(k4$center, col="blue", pch=15)
```



The metric

```
km$tot.withinss
```

```
[1] 121.7568
```

```
k4$tot.withinss
```

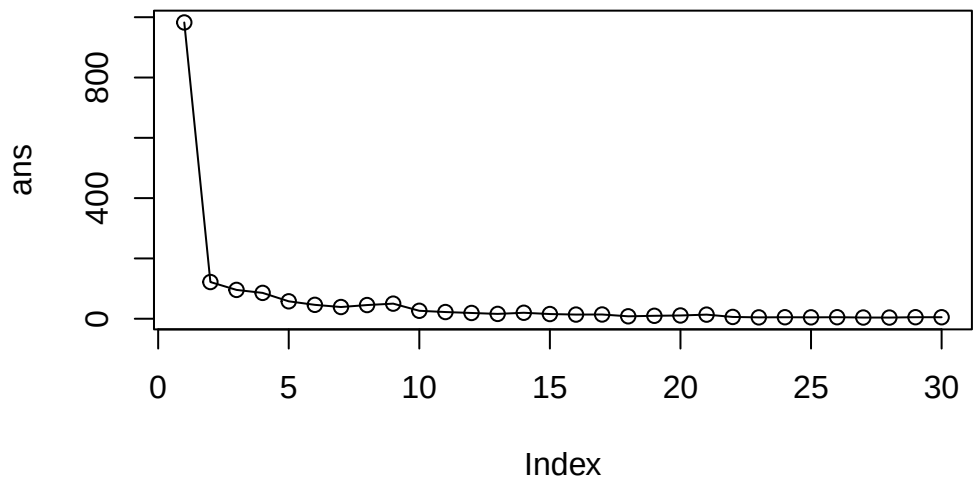
```
[1] 77.62604
```

> Q. Let's try different number of k (centers) from 1 to 30 and see what the best results is?

```
i <-1
ans <- NULL
for (i in 1:30) {
ans <- c(ans, kmeans(x, centers=i)$tot.withinss)
}

ans
```

```
 [1] 982.620801 121.756810  95.637217  85.734450  57.788111  46.058599
 [7]  38.812114  45.366701  50.073939  26.390376  22.255789  19.154714
[13]  15.985639  19.842222  15.568101  13.792783  14.061086   8.101669
[19]   9.740967  10.826640  13.631115   5.962910   4.558157   4.943509
[25]   4.647703   5.048145   4.011763   3.880538   5.069867   4.945483
```

```
plot(ans, typ="o")
```

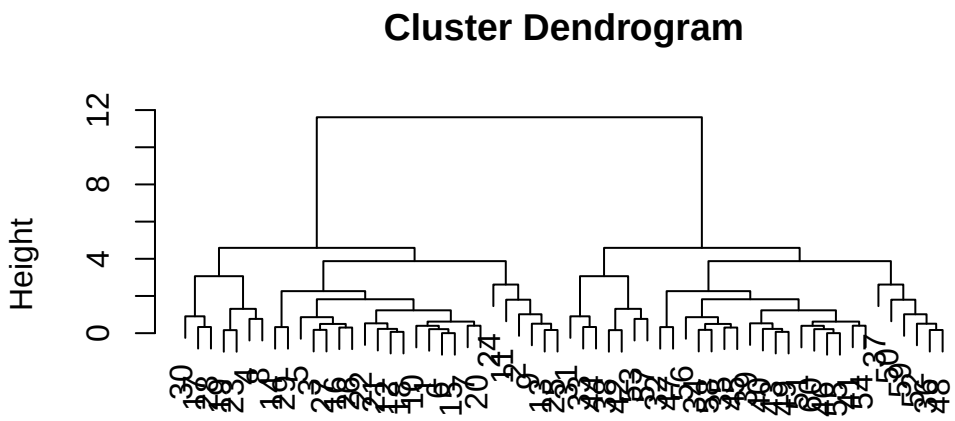


Key-Point: K-means will impose a clustering structure on your data even if it is not there - it will always give you the answer you asked for even if that answer is silly!

Hierarchical CLustering

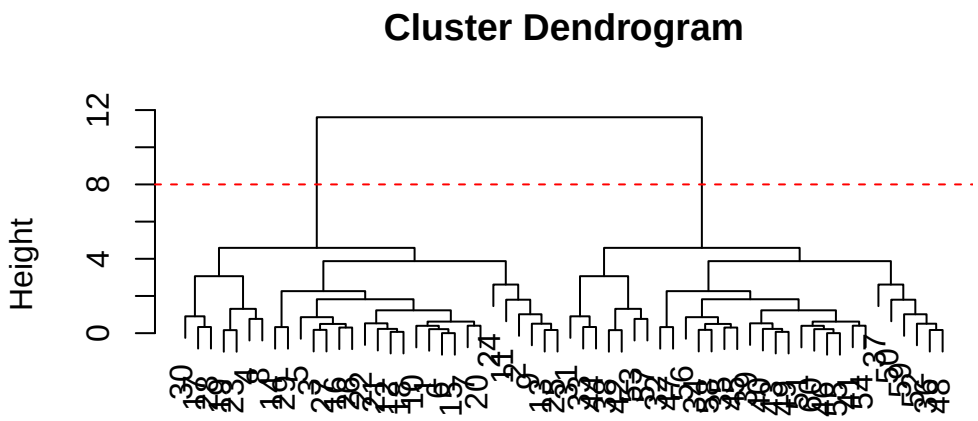
The main function for Hierarchical clustering is called `hclust()`. Unlike `kmeans()` (which does all the work for you) you can't just pass `hclust()` our raw input data. It needs a "distance matrix" like the one returned from the `dist()` function.

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



To extract our cluster membership vector from a `hclust()` result object we have to "cut" our tree at a given height to yield separate "groups"/"branches".

```
plot(hc)
abline(h=8, col="red", lty=2)
```



To do this we used the `cutree()` function on our `hclust()` object:

```
grps <-cutree(hc, h=8)
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 2 2
```

```
table(grps, km$cluster)
```

```
grps  1  2
   1  0 30
   2 30  0
```

PCA of UK food data

Import the dataset of food consumption in the UK:

```
url <- "https://tinyurl.com/UK-foods"
x <-read.csv(url)
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
7      Fresh_potatoes     720   874      566      1033
8          Fresh_Veg     253   265      171       143
9          Other_Veg     488   570      418       355
10 Processed_potatoes     198   203      220       187
11     Processed_Veg     360   365      337       334
12       Fresh_fruit    1102  1137      957       674
13           Cereals    1472  1582     1462      1494
14         Beverages      57    73       53        47
15       Soft_drinks    1374  1256     1572      1506
16   Alcoholic_drinks     375   475      458       135
17      Confectionery      54    64       62        41
```

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

```
dim(x)
```

```
[1] 17  5
```

One solution to set the row names is to do it by hand...

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

To remove the first column I can use the minus index trick

```
x <-x[,-1]
```

A better way to do this is to set the row names to the first column with `read.csv()`

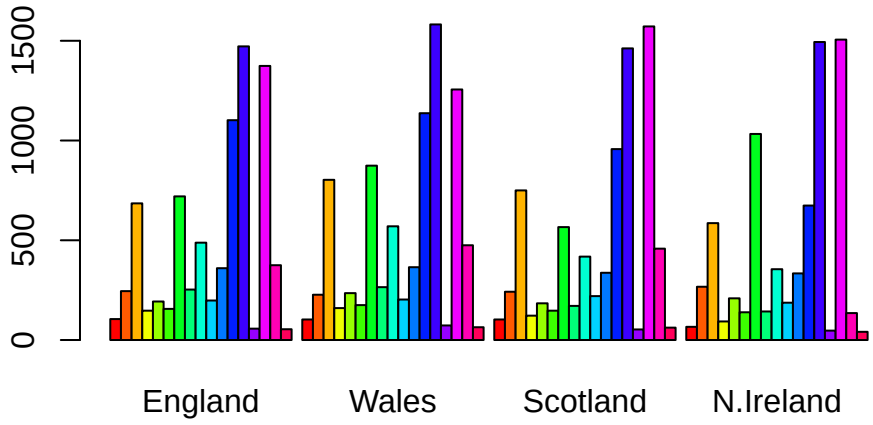
```
x <- read.csv(url, row.names=1)
x
```

	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

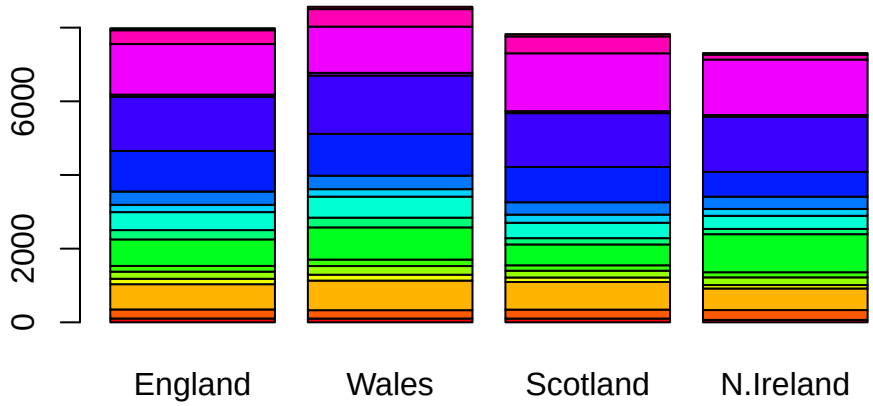
Spotting manjor differences and trends

It is difficult even in this wee 17D dataset...

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

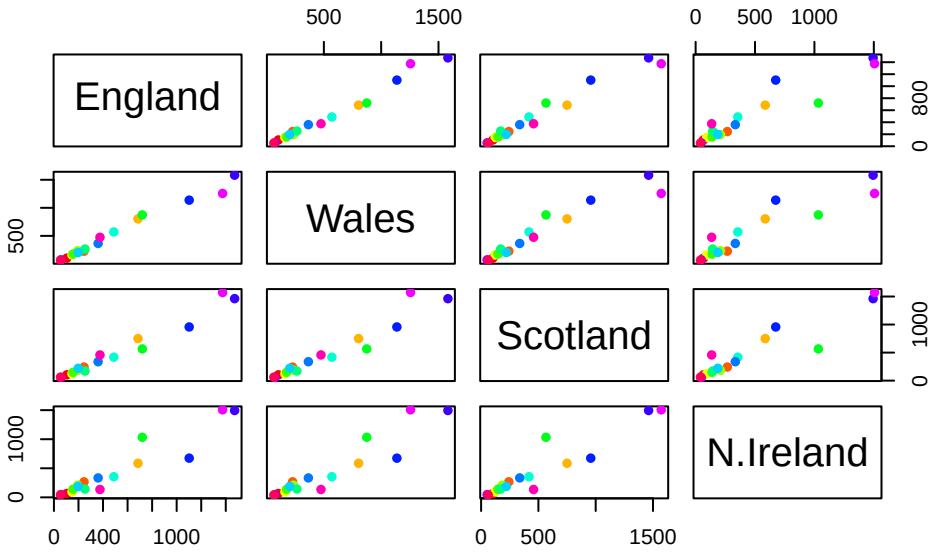



```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```

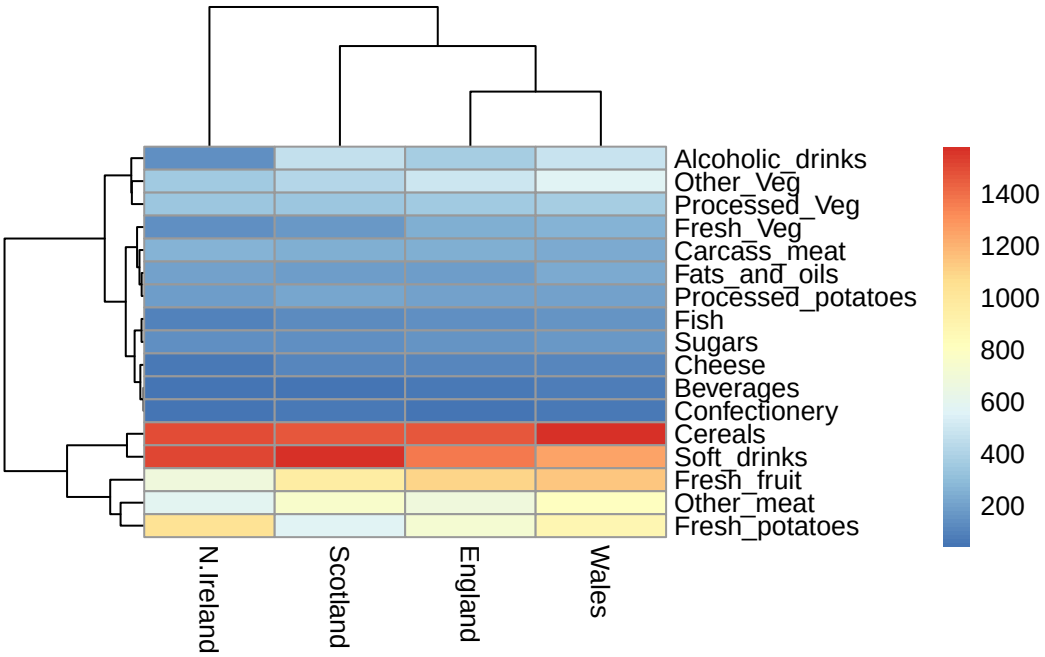


Paris Plots and heat maps

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




```
library(pheatmap)

pheatmap( as.matrix(x) )
```



PCA to the rescue

The main PCA function in "base R" is called `prcomp()`. This function wants the transpose of our food data as input (i.e. the foods as columns and the countries as rows).

```
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
```

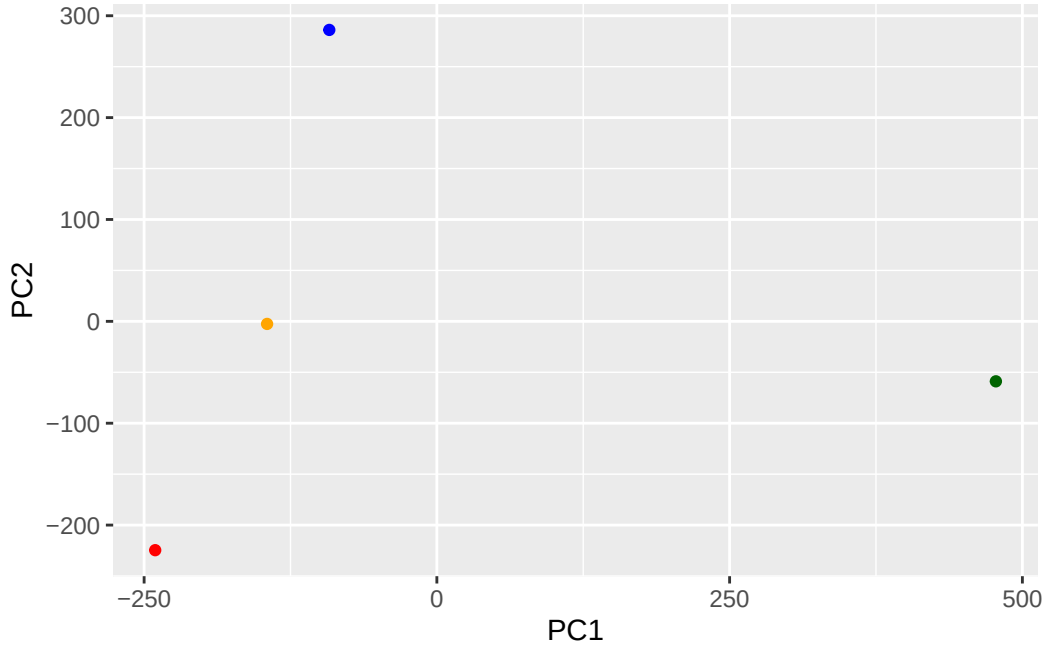
```
attributes(pca)
```

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

$class
[1] "prcomp"
```

To make one of main PCA result figures we turn to `pca$x` the scores along our new PCs. This is called "PC plot" or "score plot" or "oriendation plot"...

```
my_cols<- c("orange", "red", "blue", "darkgreen")
```

```
library(ggplot2)

ggplot(pca$x)+
  aes(PC1, PC2)+
  geom_point(col= my_cols)
```



The second major result figure is called a "loadings plot" of "variable contributions plot" or "weight plot"

```
pca$rotation
```

```
                      PC1          PC2         PC3          PC4
Cheese        -0.056955380  0.016012850  0.02394295 -0.694538519
Carcass_meat   0.047927628  0.013915823  0.06367111  0.489884628
Other_meat    -0.258916658 -0.015331138 -0.55384854  0.279023718
Fish          -0.084414983 -0.050754947  0.03906481 -0.008483145
Fats_and_oils -0.005193623 -0.095388656 -0.12522257  0.076097502
```

```
Sugars               -0.037620983 -0.043021699 -0.03605745  0.034101334
Fresh_potatoes        0.401402060 -0.715017078 -0.20668248 -0.090972715
Fresh_Veg            -0.151849942 -0.144900268  0.21382237 -0.039901917
Other_Veg            -0.243593729 -0.225450923 -0.05332841  0.016719075
Processed_potatoes   -0.026886233  0.042850761 -0.07364902  0.030125166
Processed_Veg        -0.036488269 -0.045451802  0.05289191 -0.013969507
Fresh_fruit          -0.632640898 -0.177740743  0.40012865  0.184072217
Cereals              -0.047702858 -0.212599678 -0.35884921  0.191926714
Beverages            -0.026187756 -0.030560542 -0.04135860  0.004831876
Soft_drinks           0.232244140  0.555124311 -0.16942648  0.103508492
Alcoholic_drinks     -0.463968168  0.113536523 -0.49858320 -0.316290619
Confectionery        -0.029650201  0.005949921 -0.05232164  0.001847469
```

```
ggplot(pca$rotation)+
  aes(PC1, rownames(pca$rotation))+
  geom_col()
```