

# Class 07: Machine Learning 1

Kimberly Navarro (A17485724)

## Table of contents

Background . . . . .	1
K-means clustering . . . . .	3
Hierarchical Clustering . . . . .	7
PCA of UK food data . . . . .	9
Spotting major differences and trends . . . . .	12
Pairs plots and heatmaps . . . . .	13
PCA to the rescue . . . . .	15

## Background

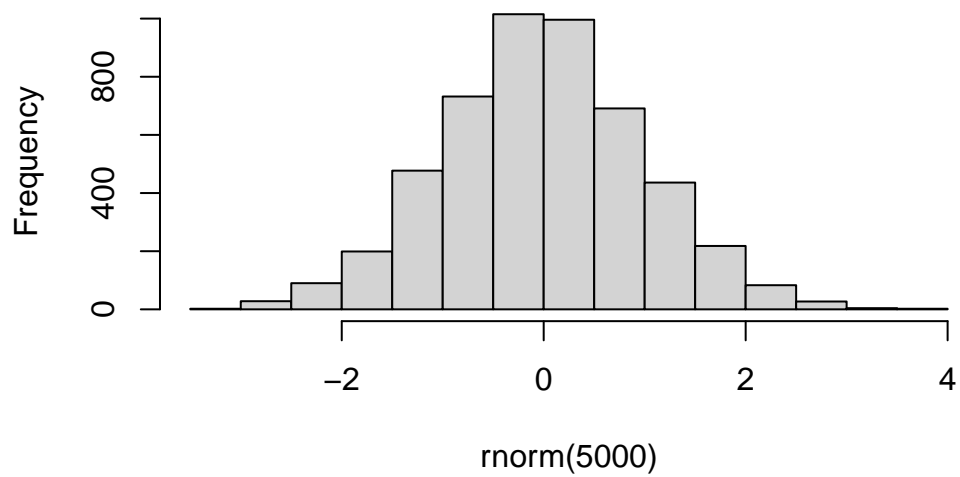
Today we will begin our exploration of some important machine learning methods, namely **clustering** and **dimensionality reduction**.

lets make up some input data for clustering where we know what natural **clusters** are:

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

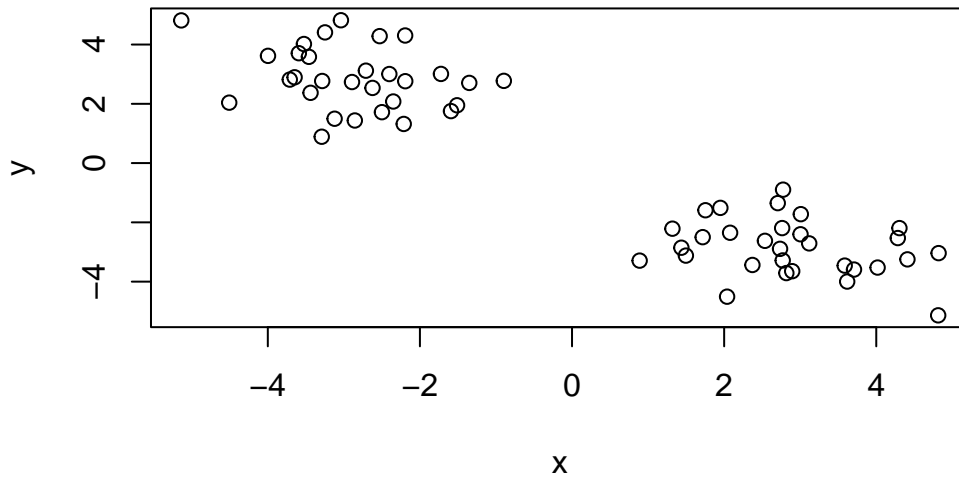
```
hist( rnorm(5000))
```

## Histogram of rnorm(5000)



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	-2.851421	2.858105
2	2.858105	-2.851421

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 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Within cluster sum of squares by cluster:

```
[1] 58.40937 58.40937
(between_SS / total_SS = 89.3 %)
```

Available components:

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

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

```
km$size
```

```
[1] 30 30
```

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

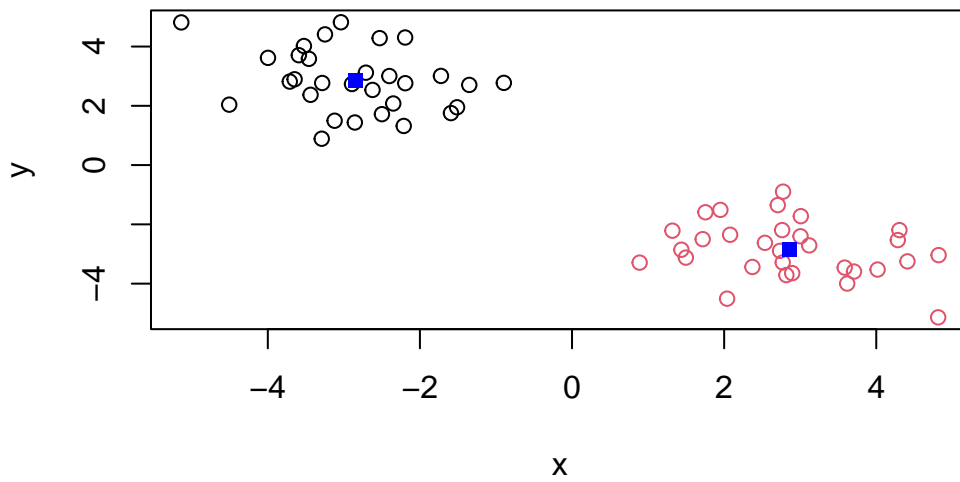
Q. What component of the results object details the cluster membership vector (i.e. our main result of which points lie in which cluster)?

```
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 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

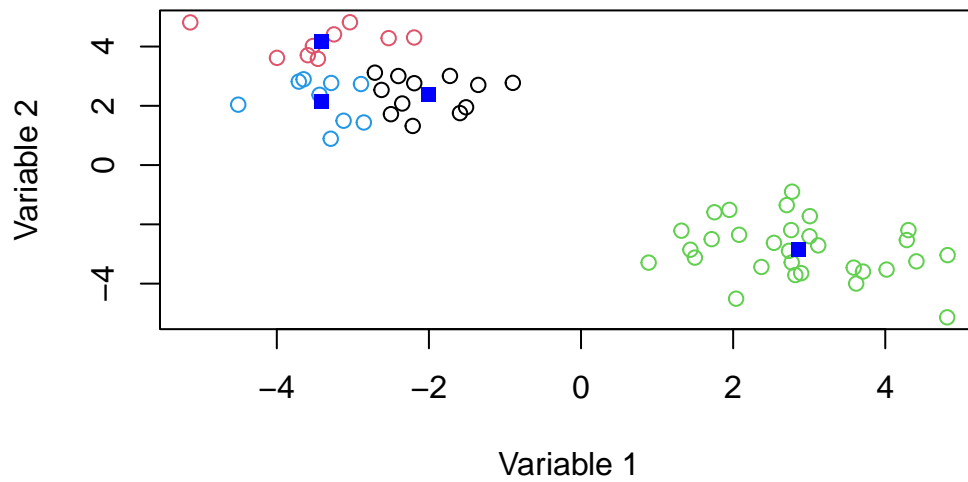
Q. Plot our clustering results which ppoints colored by clusters and also add the cluster centers as new points colored blue?

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



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

```
kd <- kmeans(x, centers = 4)
plot(x, col = kd$cluster,
      xlab = "Variable 1", ylab = "Variable 2")
points(kd$centers, col = "blue", pch = 15)
```



The metric

```
km$tot.withinss
```

```
[1] 116.8187
```

```
kd$tot.withinss
```

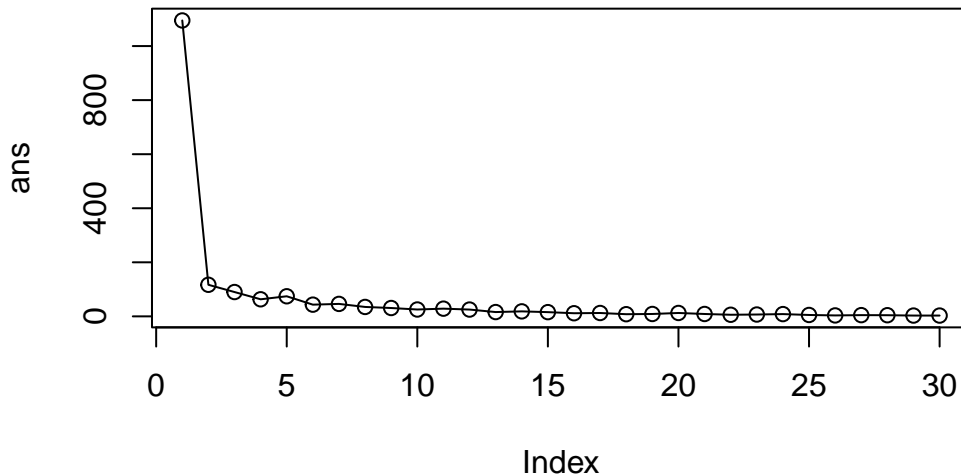
```
[1] 79.98502
```

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

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

[1]	1094.779160	116.818739	89.995958	63.173178	74.334639	43.297615
[7]	46.072705	34.828480	30.865676	25.412626	28.570085	25.186343
[13]	15.881579	18.476562	15.570820	11.510940	12.417254	7.989341
[19]	8.769585	12.550108	8.807445	5.971022	6.849298	8.393531
[25]	5.155750	3.504319	4.524611	4.223116	2.935456	2.987907

```
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

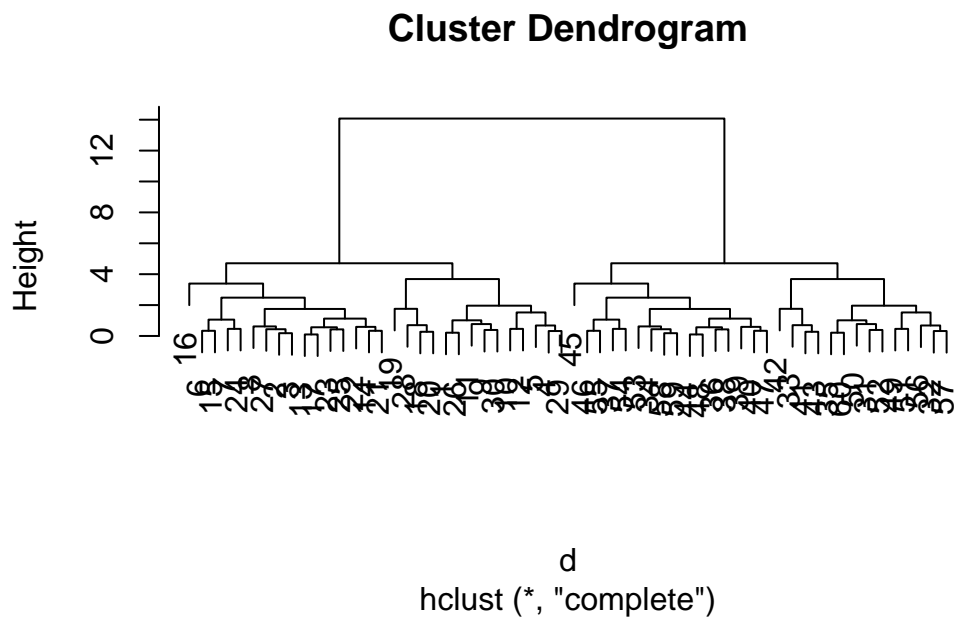
The main function for Hierarchical Clustering is called `hclust()` Unlike `kmeans()` (which does all the work for you) you can't just pass `hclust()` our raw input data. It needs a "distance matrix" like the new one returned from the `dist()` function.

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

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

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

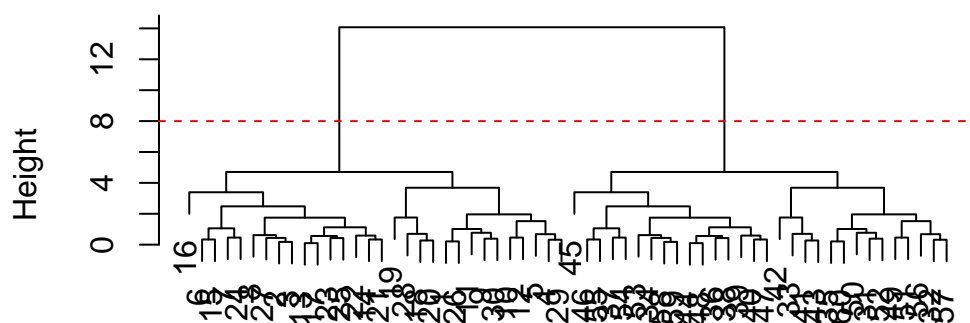
```
plot(hc)
```



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

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

## Cluster Dendrogram



d  
hclust (\*, "complete")

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

```
cutree(hc, h=8)
```

```
[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 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

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

```
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

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

```
dim(x)
```

```
[1] 17  5
```

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

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

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

```
x <- x[,-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

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

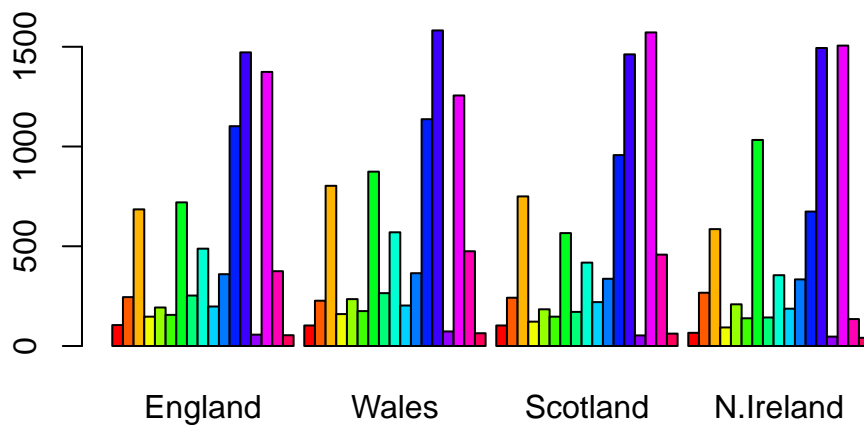
Q2. Which approach to solving the ‘row-names problem’ mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

Note: I personally prefer the `x <- read.csv(url, row.names = 1)` way because it seems easier.

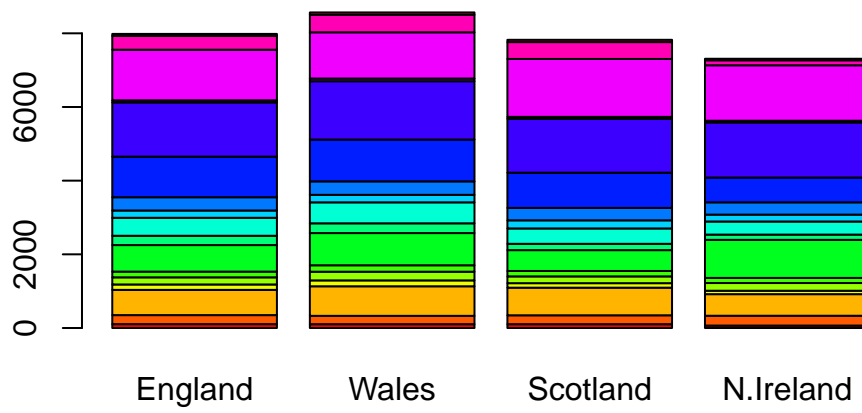
## Spotting major differences and trends

Is difficult even in this wee 17D dataset...

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

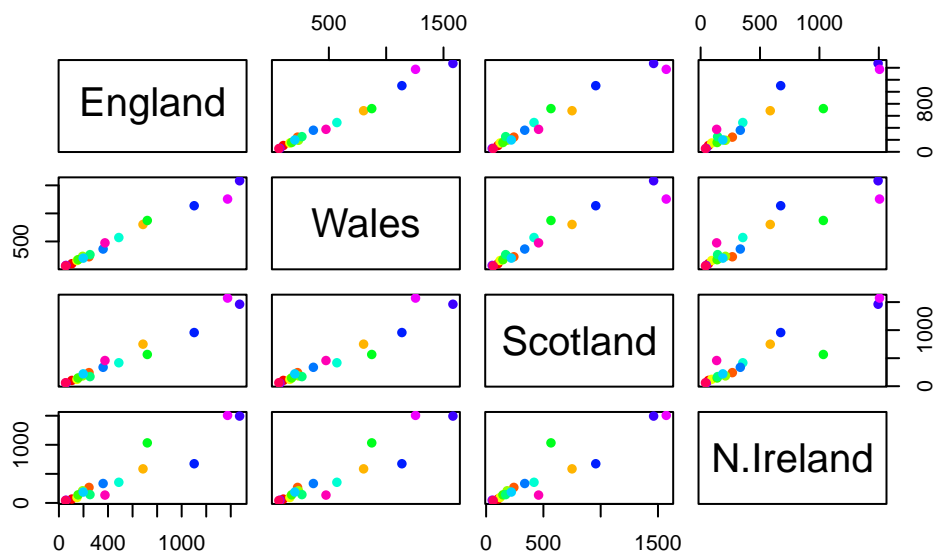


```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



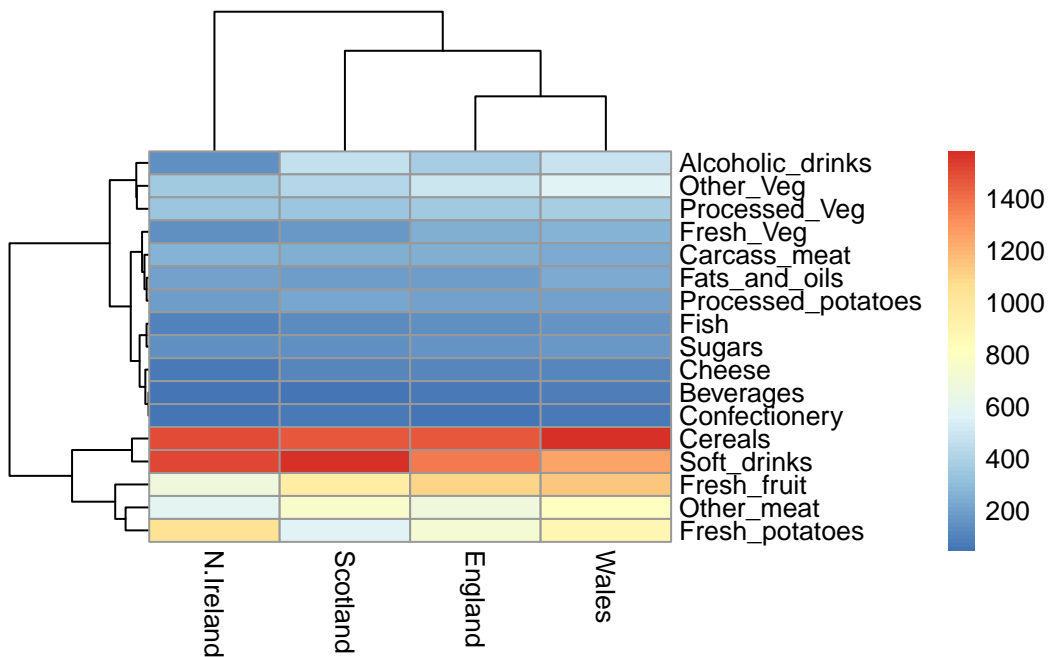
### Pairs plots and heatmaps

```
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 countries as rows).

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

```
summary(pca)
```

Importance of components:

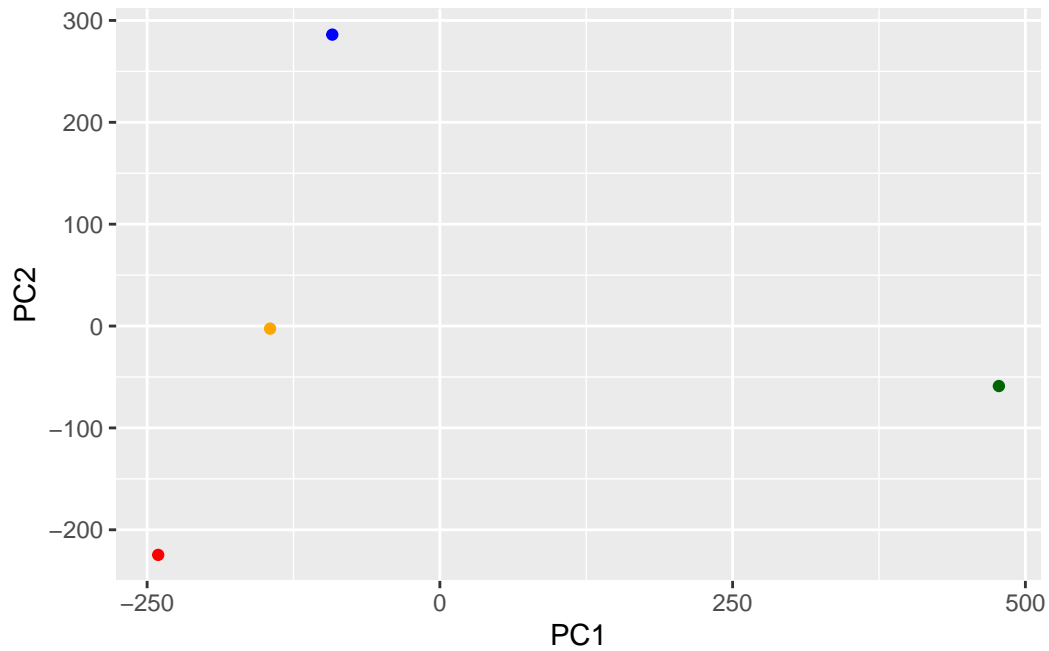
	PC1	PC2	PC3	PC4
Standard deviation	324.1502	212.7478	73.87622	2.7e-14
Proportion of Variance	0.6744	0.2905	0.03503	0.0e+00
Cumulative Proportion	0.6744	0.9650	1.00000	1.0e+00

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

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

```
library(ggplot2)

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



The second major result figure is called a “loading plot” of “variable contributions plot” or “weight plot”

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

