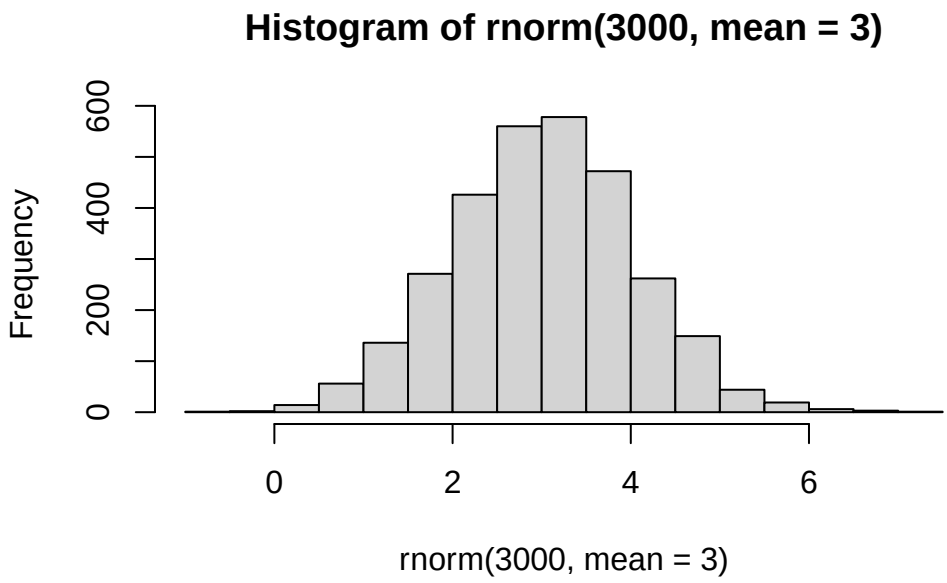
class 7: Machine learning 1

Shivani Lakkaraju (A12652928)

Today we will delve into unsupervised machine learning with a initial focus on clustering and dimensionality reduction.

Let's start by making up some data to cluster: The 'rnorm()' function can help:

```
hist(rnorm(3000, mean=3))
```



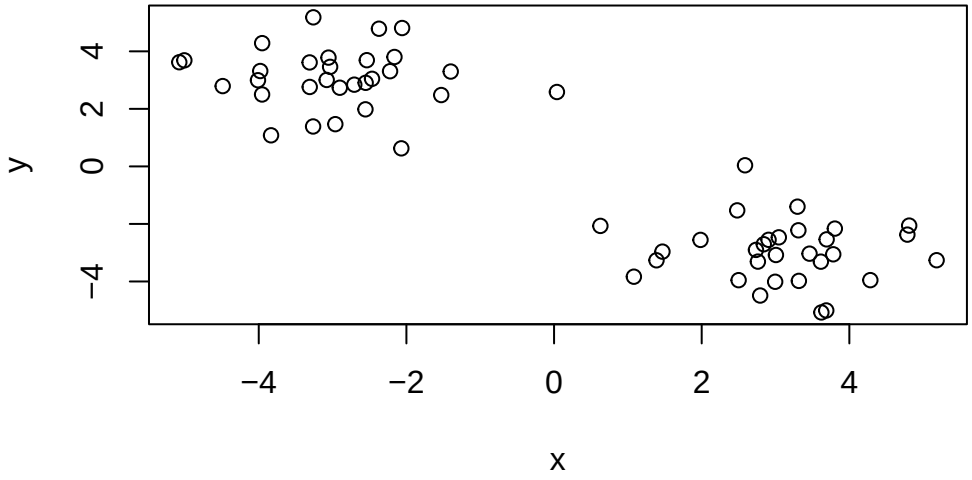
Let's get some data centered at 3, -3 -3, 3

```
# combine 30 +3 values with 30 -3 values
x <- c(rnorm(30, mean=3), rnorm(30, mean=-3))

# Bind these values together
```

```r
z <- cbind(x=x, y=rev(x))
head(z)
```

```
            x         y
[1,] 2.500203 -3.954864
[2,] 1.468282 -2.962934
[3,] 3.621000 -5.076561
[4,] 2.761090 -3.309622
[5,] 4.284288 -3.953983
[6,] 3.297499 -1.399815
```

```r
plot(z)
```



##k-means now we can see how k-means clusters this data. The main function for k-means clustering in 'base R" is called 'kmeans()'

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

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

```
Cluster means:
          x         y
1 -2.969042  3.061433
2  3.061433 -2.969042

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] 66.20707 66.20707
 (between_SS / total_SS =  89.2 %)

Available components:

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

```
attributes(km)
```

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

$class
[1] "kmeans"
```

Q. what size is each cluster?

```
km$size
```

```
[1] 30 30
```

Q. cluster membership vector (ie: the answer: cluster to which each point is allocated)

```
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
```

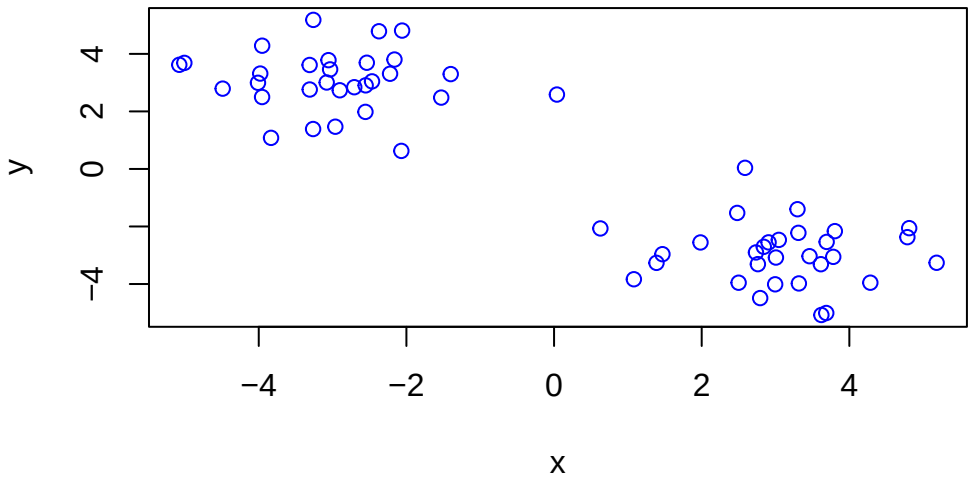
Q. cluster center

```
km$centers
```

```
          x         y
1 -2.969042  3.061433
2  3.061433 -2.969042
```

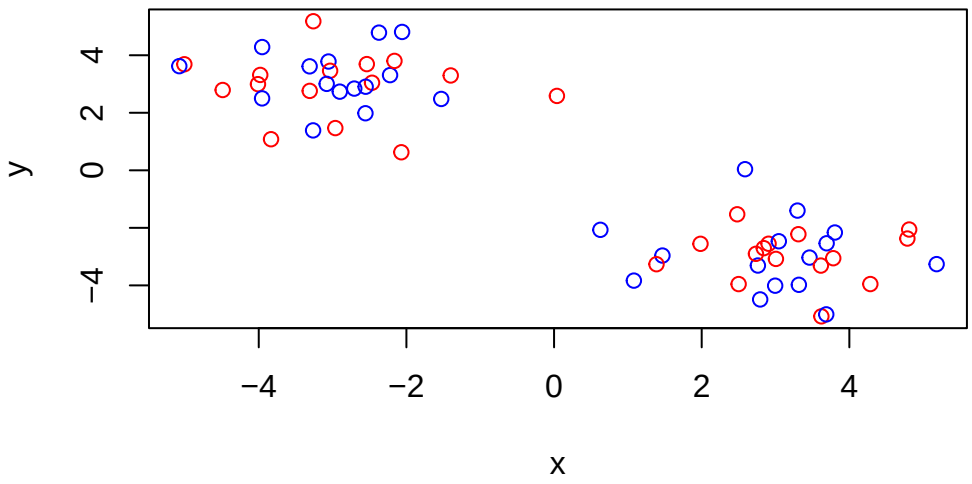
Q. make a results figure, i.e.: plot the data 'z' colored by cluster membership and show cluster centers.

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



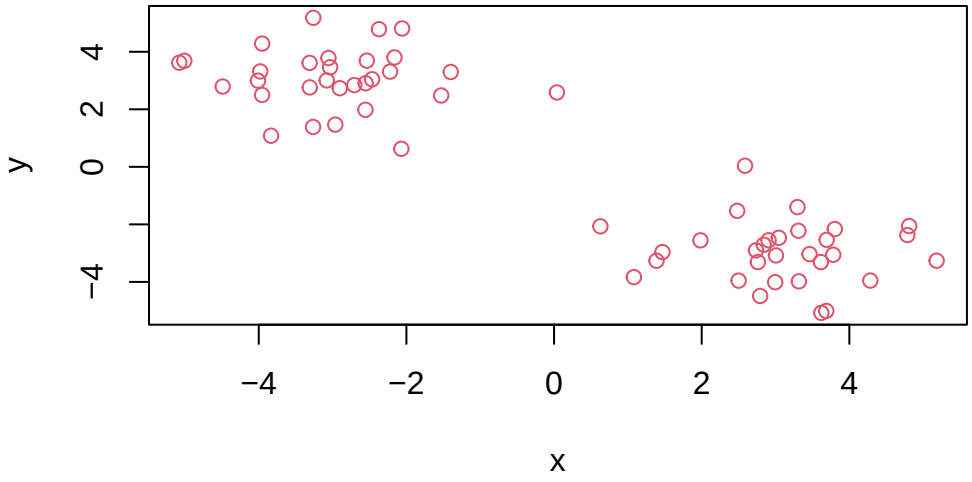
This uses recycling under the hood:

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



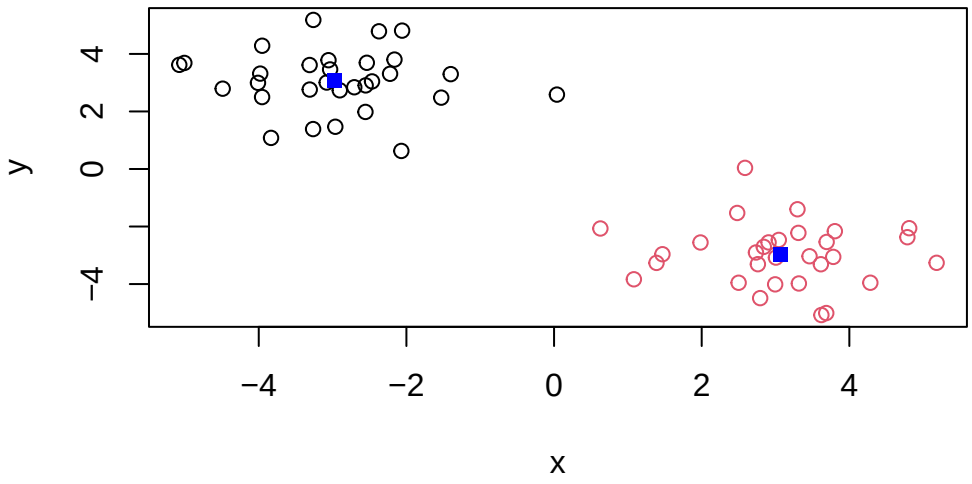
you can specify color based on a number, where 1 is black, 2 is red

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



you can use the membership vector to color by cluster

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



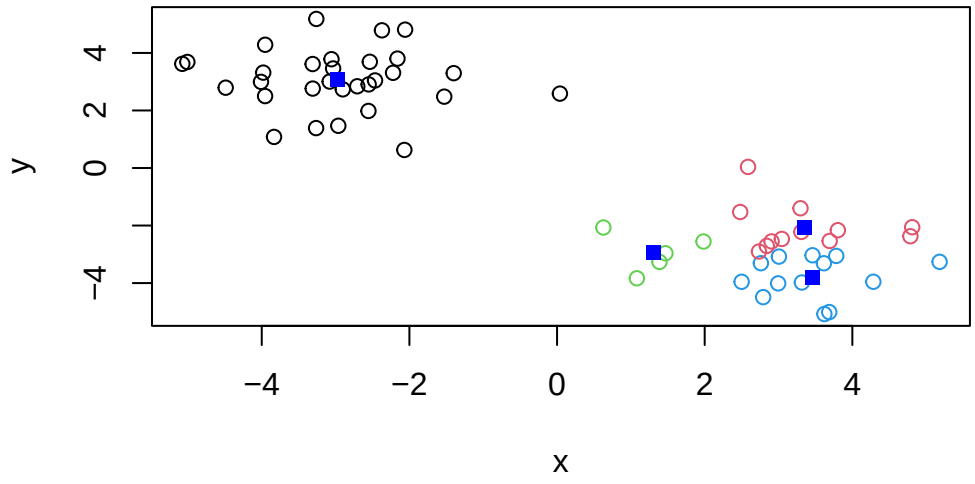
Q. rerun k-means clustering with 4 clusters and plot results as above

```
km2 <- kmeans(z, centers=4)
km2
```

```
K-means clustering with 4 clusters of sizes 30, 12, 5, 13

Cluster means:
          x         y
1 -2.969042  3.061433
2  3.357771 -2.072032
3  1.309677 -2.936824
4  3.461642 -3.809443

Clustering vector:
 [1] 4 3 4 4 4 2 2 2 2 4 2 3 4 4 2 3 3 2 4 2 2 2 3 4 2 4 4 4 2 4 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] 66.207075 13.804475  2.814883 12.266232
 (between_SS / total_SS =  92.2 %)

Available components:

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

```
plot(z, col=km2$cluster)
points(km2$centers, col="blue", pch=15)
```



hierarchical clustering

The main "base R" function for this is "hclust()". Unlike 'kmeans()' you can't just give your dataset as input, you need to provide a distance matrix.

We can use the 'dist()' function for this.

```
d <- dist(z)
dim(z)
```

```
[1] 60  2
```

```
#hclust()
```

```
hclust(d)
```

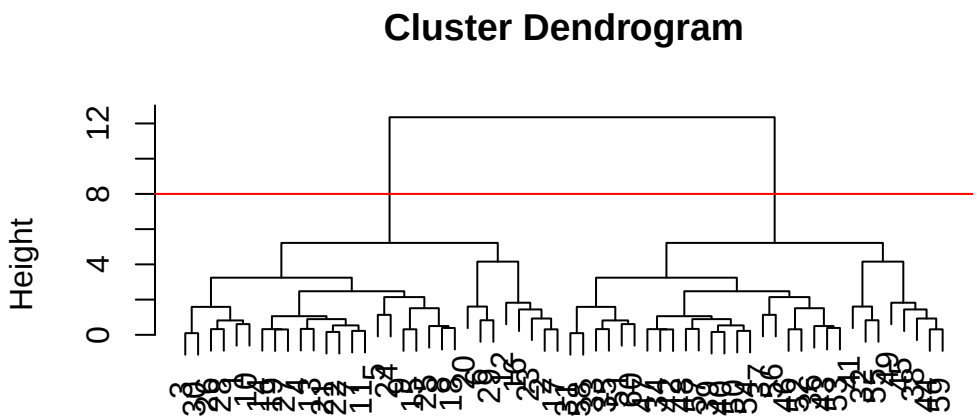
```
Call:
hclust(d = d)

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

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

```
Call:
hclust(d = d)

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

There is a custom plot() for hclust objects, let's see it.

```
hc <- hclust(d)
plot(hc)
abline(h=8, col="red")
```



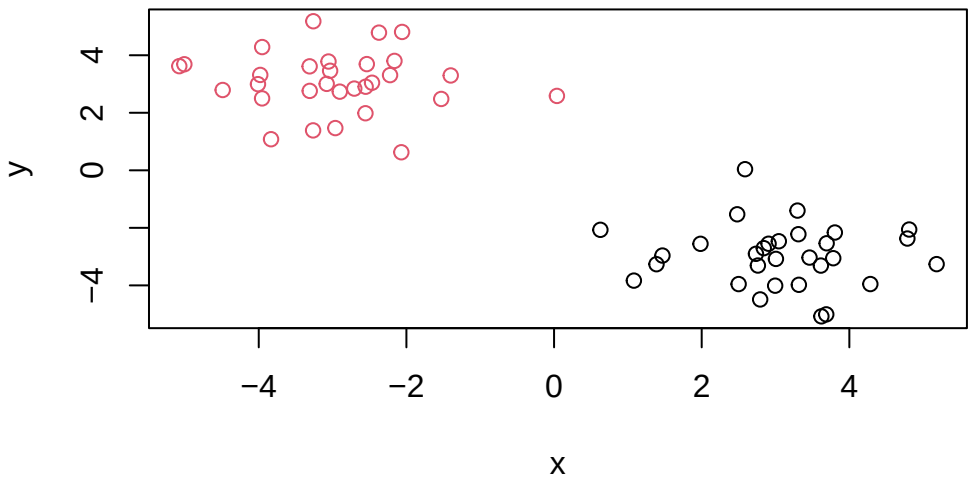
The function to extract clusters/grps from a hclust tree/object is called 'cutree()':

```
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
```

Q. Plot data with hclust clusters:

```
plot(z, col=grps)
```



Principal component analysis (PCA)

The main function for PCA in base R for PCA is called 'prcomp()'. There are many add on packages with PCA functions tailored to particular data types (RNASeq, protein structures, metagenomics, etc…)

PCA of UK food data

read the data into R, it is a csv so we can use 'read.csv()':

```
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
```

I want the food names as row names, not their own column of data (first column currently). I can fix this like so:

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

A better way to do this is to do it at the time of data import:

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

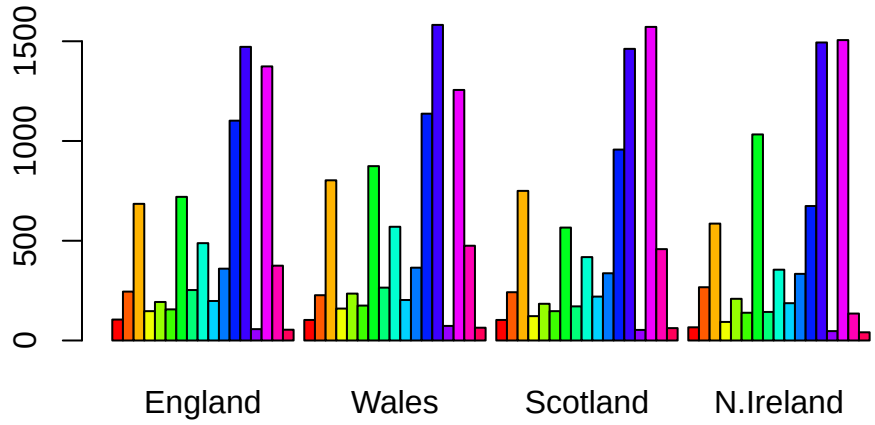
```
               England Wales Scotland N.Ireland
Cheese             105   103      103        66
Carcass_meat       245   227      242       267
Other_meat         685   803      750       586
Fish               147   160      122        93
Fats_and_oils      193   235      184       209
Sugars  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

Let's make some plots and dig into the data a little.

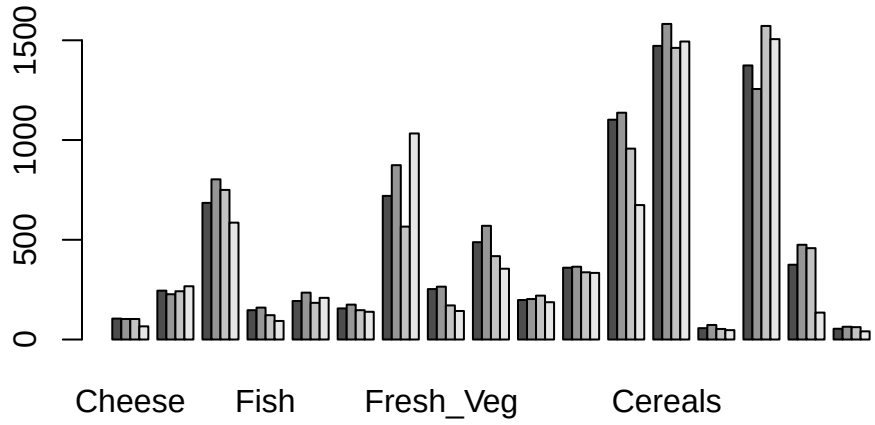
```
rainbow(nrow(food))
```

```
 [1] "#FF0000" "#FF5A00" "#FFB400" "#F0FF00" "#96FF00" "#3CFF00" "#00FF1E"
 [8] "#00FF78" "#00FFD2" "#00D2FF" "#0078FF" "#001EFF" "#3C00FF" "#9600FF"
[15] "#F000FF" "#FF00B4" "#FF005A"
```

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

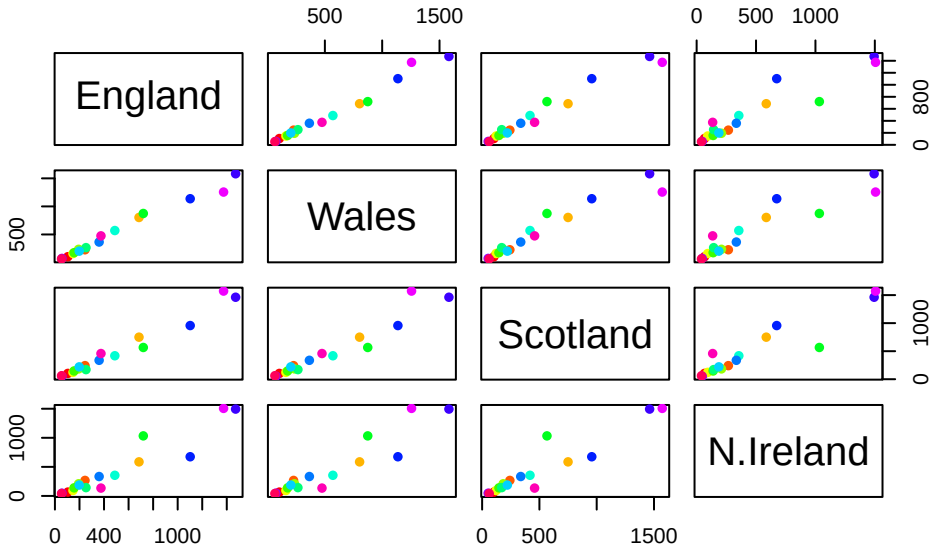


```
barplot(as.matrix(t(food)),beside=T, nrow(t(food)))
```



How about a "pairs" plot where we plot each country against all other countries.

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



There has to be a better way ..

PCA to the rescue

We can run PCA for this data with the 'prcomp()' function.

```
head(food)
```

```
               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
```

We need to take the transpose to get the foods in columns and countries in rows:

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

what is in my 'pca' result object?

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

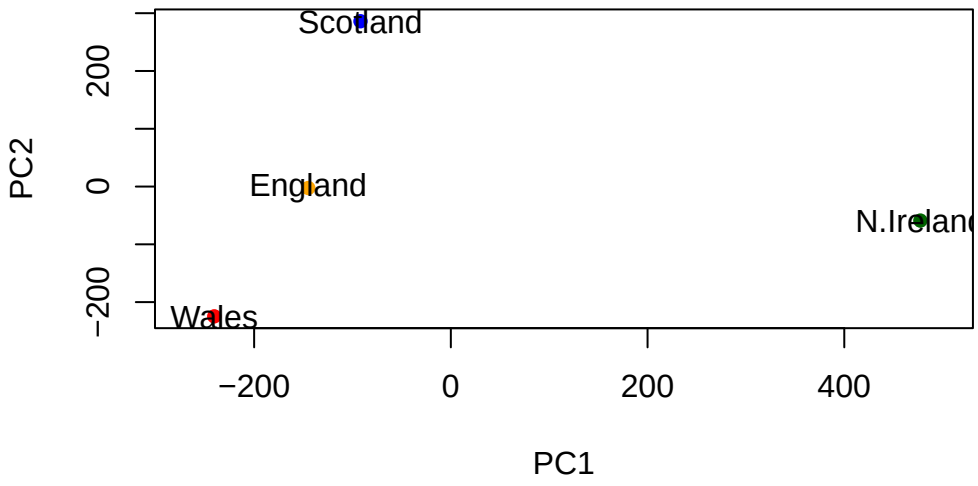
The scores along the new PCs:

```
pca$x
```

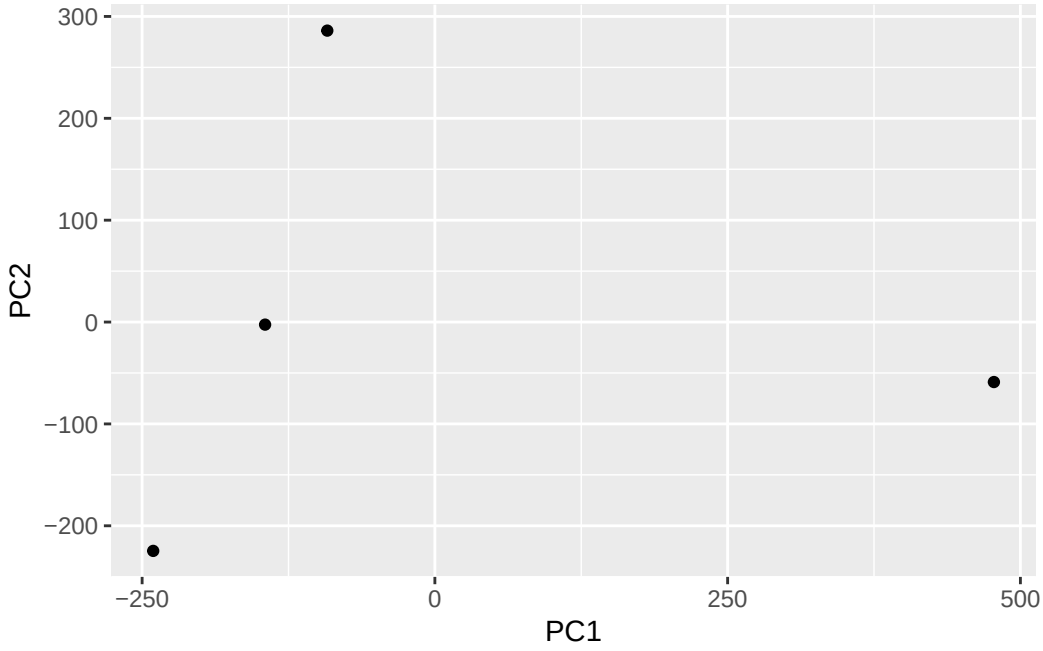
```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -4.894696e-14
Wales     -240.52915 -224.646925 -56.475555  5.700024e-13
Scotland   -91.86934  286.081786 -44.415495 -7.460785e-13
N.Ireland  477.39164  -58.901862  -4.877895  2.321303e-13
```

To make my main result figure, often called a PC plot, or score plot, oridenation plot, or PC1 V PC2 plot, etc)

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", col=c("orange", "red", "blue", "darkgreen"
text(pca$x[,1], pca$x[,2], row.names(pca$x))
```




```
library(ggplot2)

data <- as.data.frame(pca$x)

ggplot(data, aes(PC1, PC2)) + geom_point()
```



To see the contributions of the original variables (foods) to these new PCs we can look at the 'pca$rotation' component of our results object.

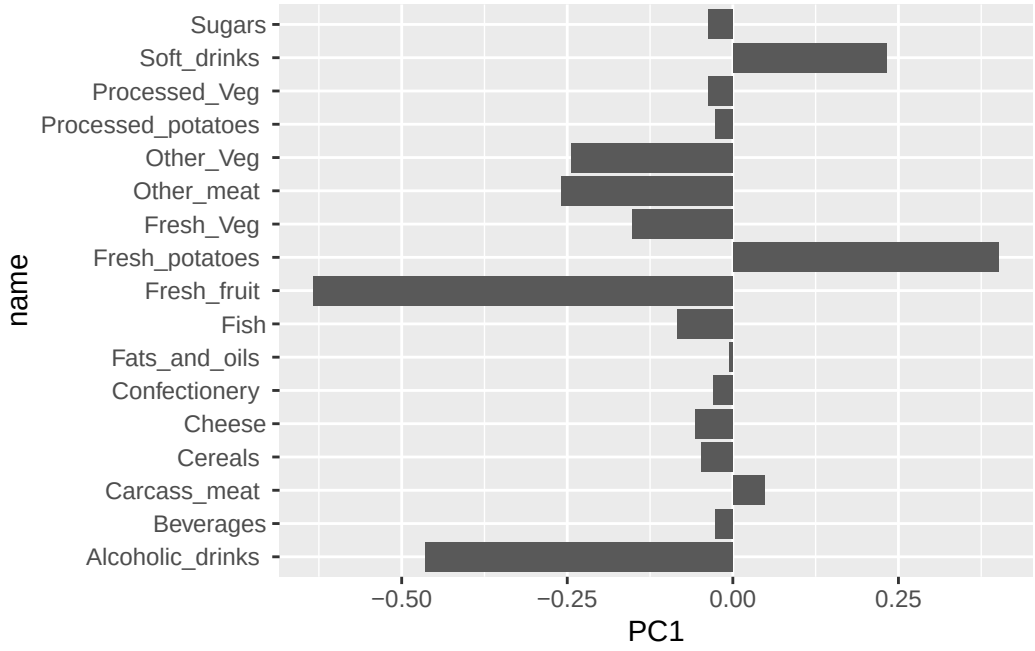
```
pca$rotation
```

	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

```r
loadings <- as.data.frame(pca$rotation)

loadings$name <- rownames(loadings)
loadings
```

```
                           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
                                 name
Cheese                         Cheese
Carcass_meat             Carcass_meat
Other_meat                 Other_meat
Fish                             Fish
Fats_and_oils           Fats_and_oils
Sugars                         Sugars
Fresh_potatoes         Fresh_potatoes
Fresh_Veg                   Fresh_Veg
Other_Veg                   Other_Veg
Processed_potatoes Processed_potatoes
Processed_Veg           Processed_Veg
Fresh_fruit               Fresh_fruit
Cereals                       Cereals
Beverages                   Beverages
Soft_drinks               Soft_drinks
Alcoholic_drinks     Alcoholic_drinks
Confectionery           Confectionery
```

```
ggplot(loadings, aes(PC1, name)) + geom_col()
```




```
ggplot(loadings, aes(PC2, name)) + geom_col()
```