Class 7: Machine Learning 1

Nicole Lin (PID: A17826971)

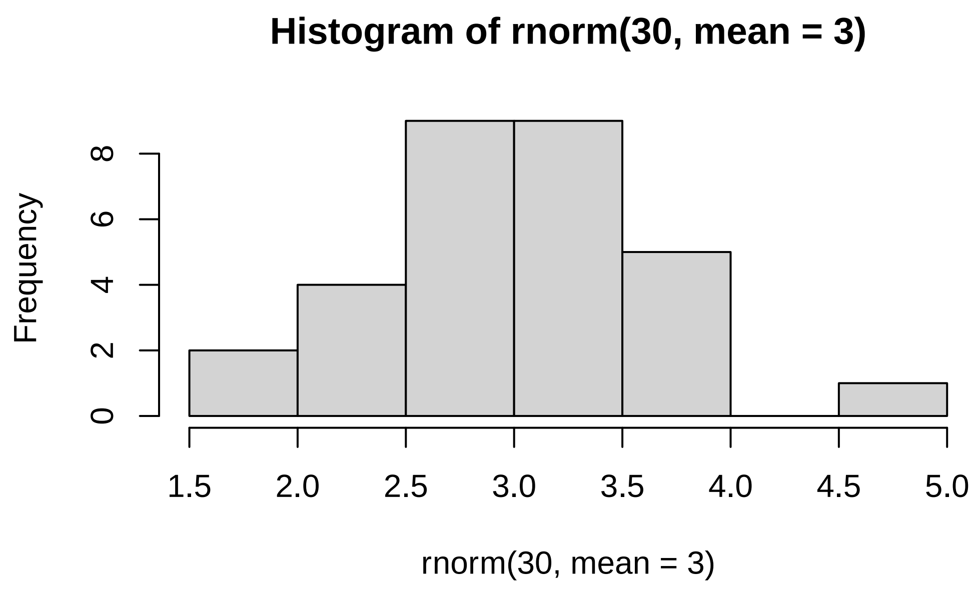
Background

Today we will begin out exploration of some important machine learning methods. Namely **clustering** and **dimensionality reduction**.

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

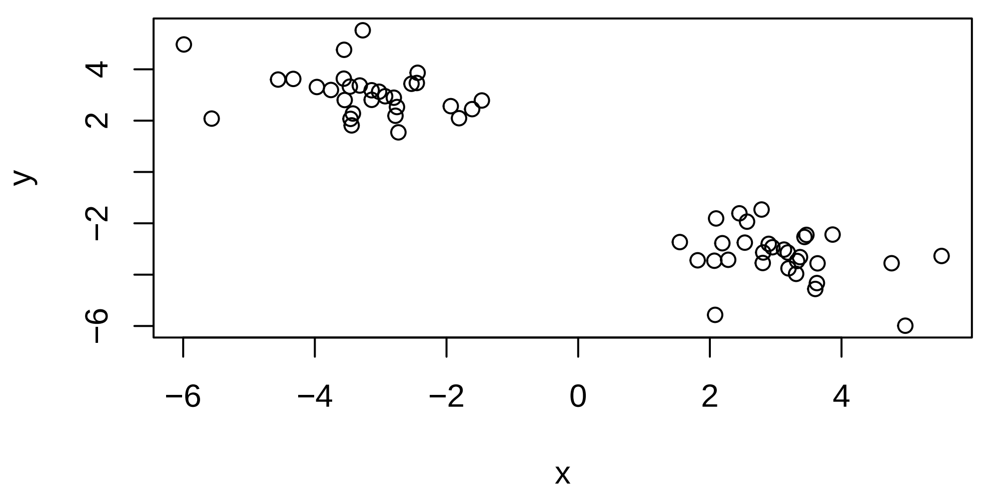
The function `rnorm()` can be useful here.

```r
hist( rnorm(30, mean = 3))
```



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

```
tmp <- c(rnorm(30, 3),
         rnorm(30, -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  3.075108 -3.224556
2 -3.224556  3.075108

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] 53.53667 53.53667
 (between_SS / total_SS =  91.7 %)

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? size

```r
km$size
```

```
[1] 30 30
```

> Q. What component of the results object details the cluster centers?

```r
km$centers
```

```
          x         y
1  3.075108 -3.224556
2 -3.224556  3.075108
```

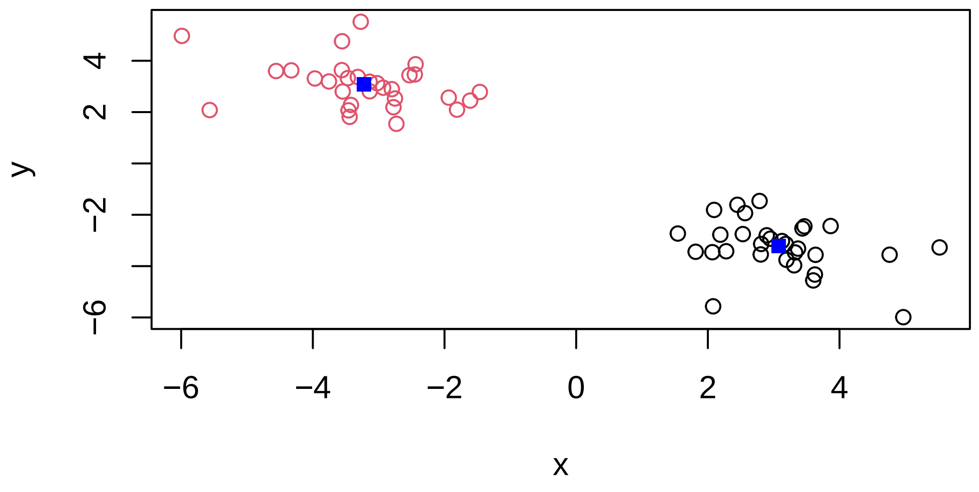
> What component of the results object details the cluster membership vector (i.e. our main result of which points lie in which cluster)?

```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
```

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

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



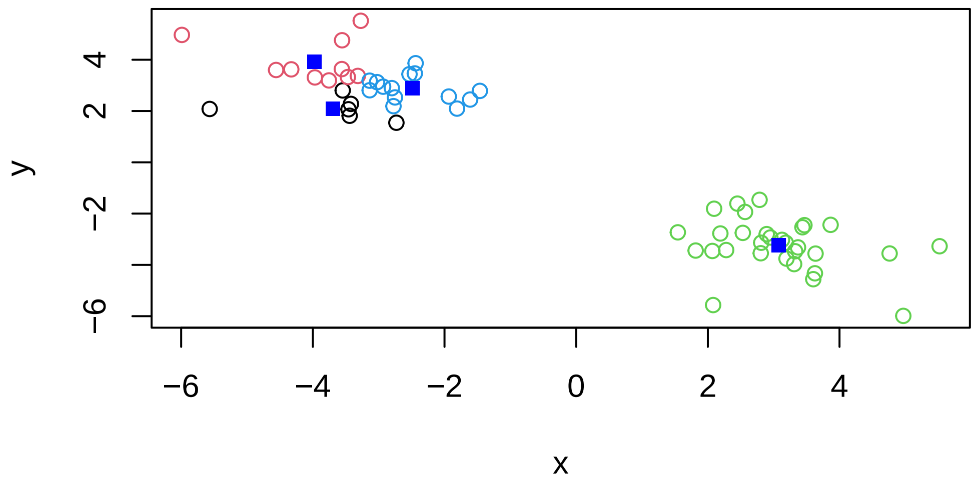
Q. Run `kmeans()` again and this time produce 4 clusters (and call your result object `k4`) and make a results figure like above.

```r
k4 <- kmeans(x, 4)
k4
```

```
K-means clustering with 4 clusters of sizes 6, 10, 30, 14

Cluster means:
          x         y
1 -3.694187  2.098295
2 -3.977333  3.931372
3  3.075108 -3.224556
4 -2.485587  2.882126

Clustering vector:
 [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 4 2 4 1 1 4 4
[39] 1 2 2 2 2 4 4 2 2 4 1 4 4 1 2 4 4 2 4 4 4 2

Within cluster sum of squares by cluster:
[1]  5.570452 12.290363 53.536665  7.462507
 (between_SS / total_SS =  93.9 %)
```

```
Available components:

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

```
plot(x, col=k4$cluster)
points(k4$centers, col="blue", pch=15)
```



The metric

```
km$tot.withinss
```

```
[1] 107.0733
```

```
k4$tot.withinss
```

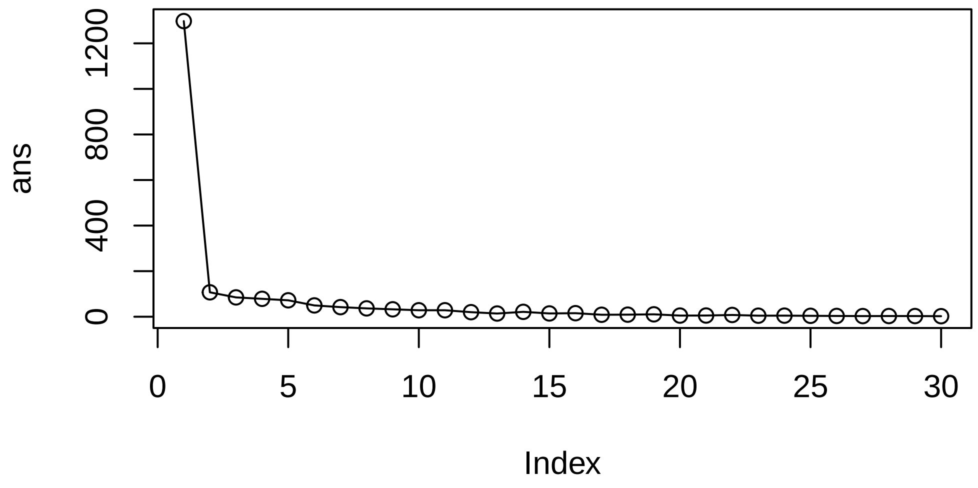
```
[1] 78.85999
```

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

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

ans
```

```
 [1] 1297.646423  107.073330   84.796798   78.498319   72.022830   49.597681
 [7]   42.091682   36.589927   32.475752   28.362635   28.413756   19.967946
[13]   14.023506   21.418543   14.346024   15.484308    8.669601    9.096797
[19]   10.440591    5.004807    5.178799    7.660381    4.475855    4.745259
[25]    3.912805    3.372393    2.795203    2.993818    2.877597    2.257538
```

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

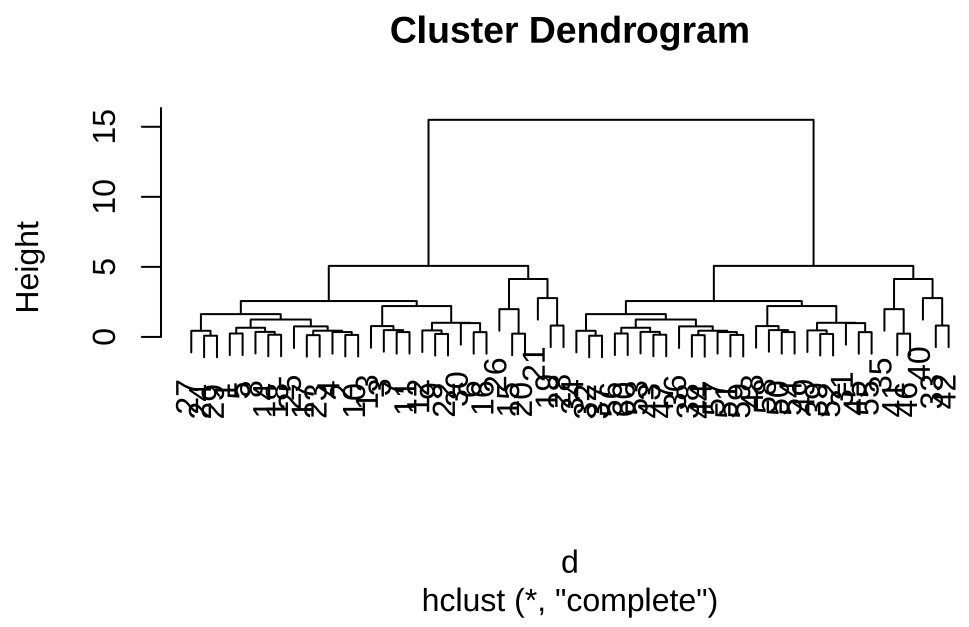


Key point: K-means will impose a clustering structure on your data even if it is not there - it will alwyas give you the answer you asked for even if that answer is silly! The best result is the one with 2 clusters (first one that drops off the "cliff") because it has the fewest number of clusters but the smallest tots.withinss, meaning that point has clusters that are the tighter and better-defined while still having a small number of clusters.

Hierarchical Clustering

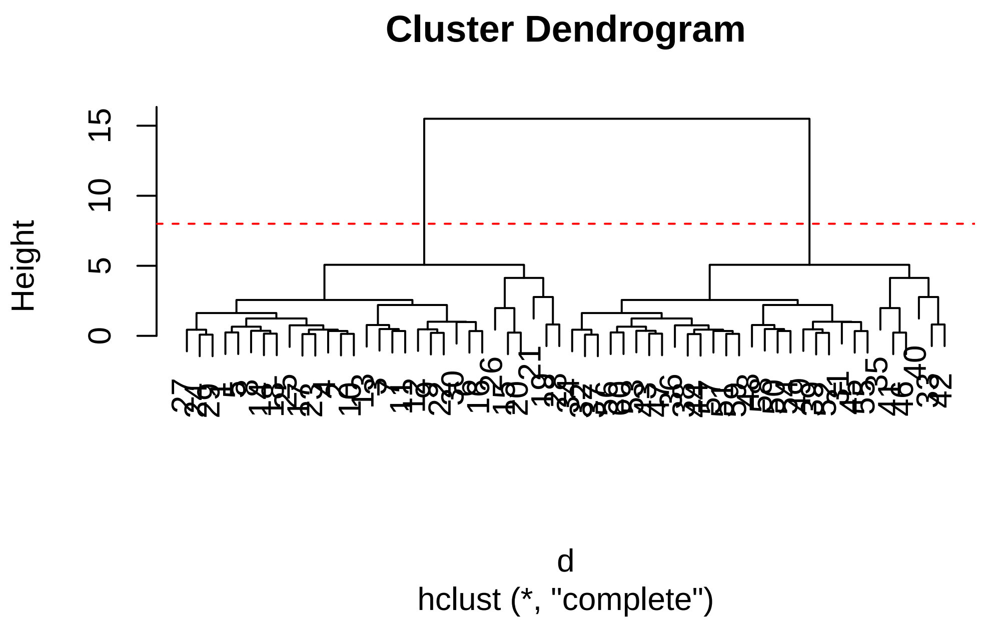
The main function for Hierarchical Clustering is called `hclust()`. Unlike `kmeans()` (which does all the work for you), you can't just pass `hclust()` our raw input data. It needs a "distance matrix" like the one returned from the `dist()` function.

```r
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."

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



To do this, we use 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 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
```

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

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

PCA of UK food data

Import the data set 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?

```r
dim(x)
```

```
[1] 17  5
```

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

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

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

```r
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()`.

```r
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
```

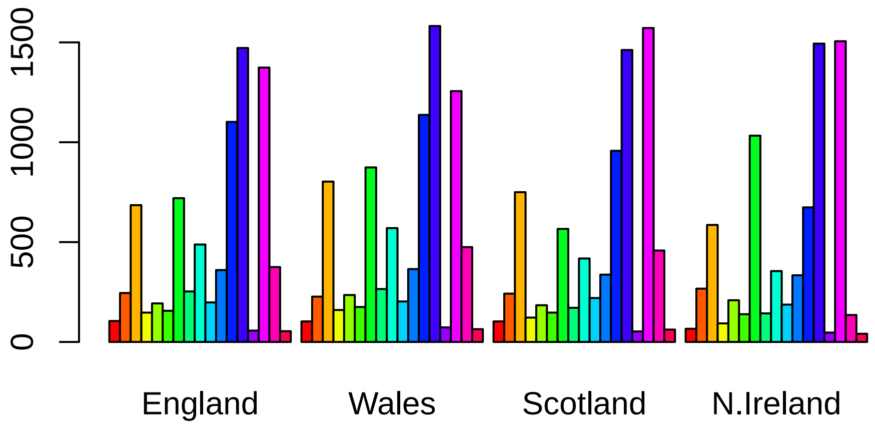
> Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

Prefer the second method because it doesn't take away a column each time you run the code. Also is shorter.

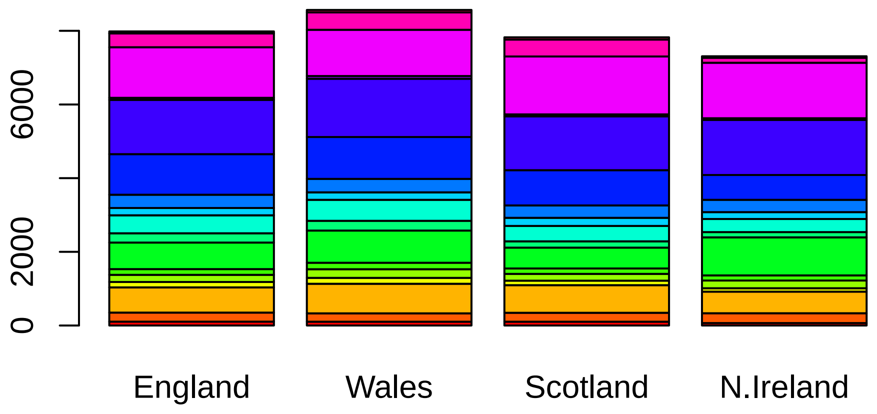
Spotting major differences and trends

It's difficult even in this wee 17 dimention data set...

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



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

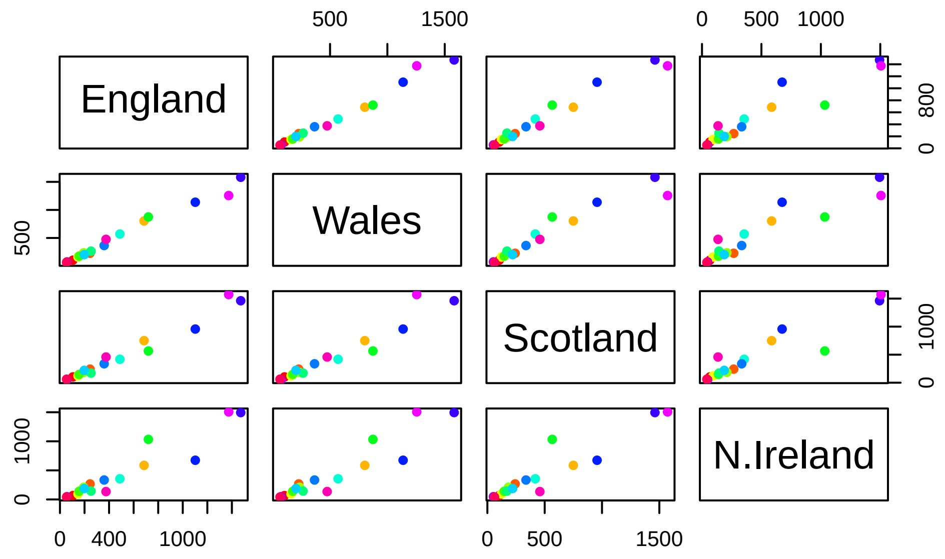


Q3: Changing what optional argument in the above barplot() function results in the following plot?

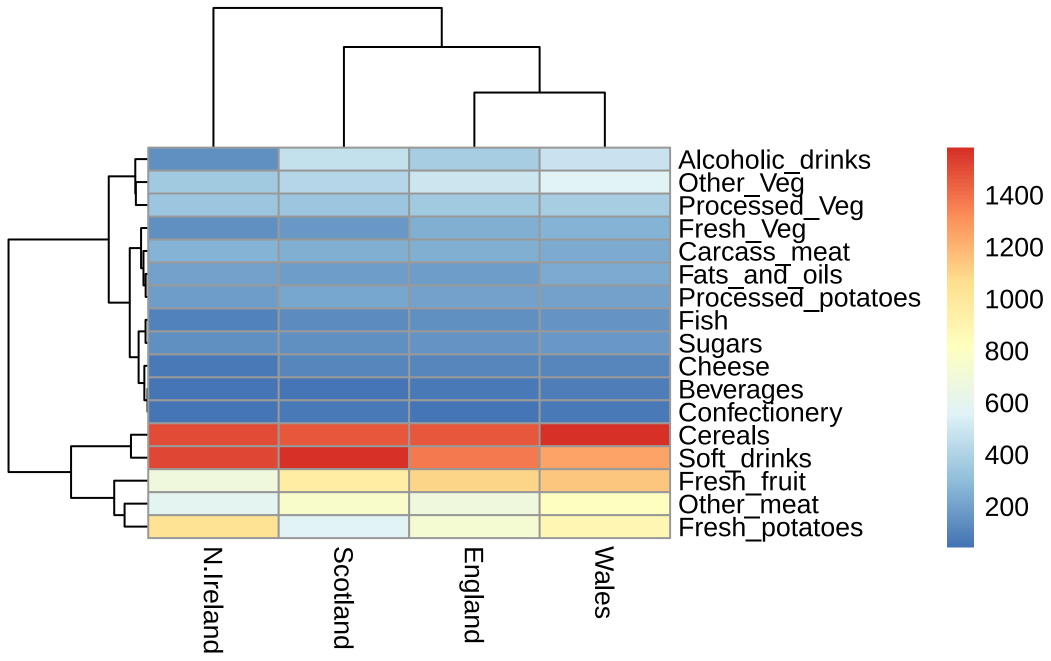
Changing the "beside" argument from T to F changes it from a histogram graph to a stacked graph.

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 the countries as rows).

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

```r
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
```

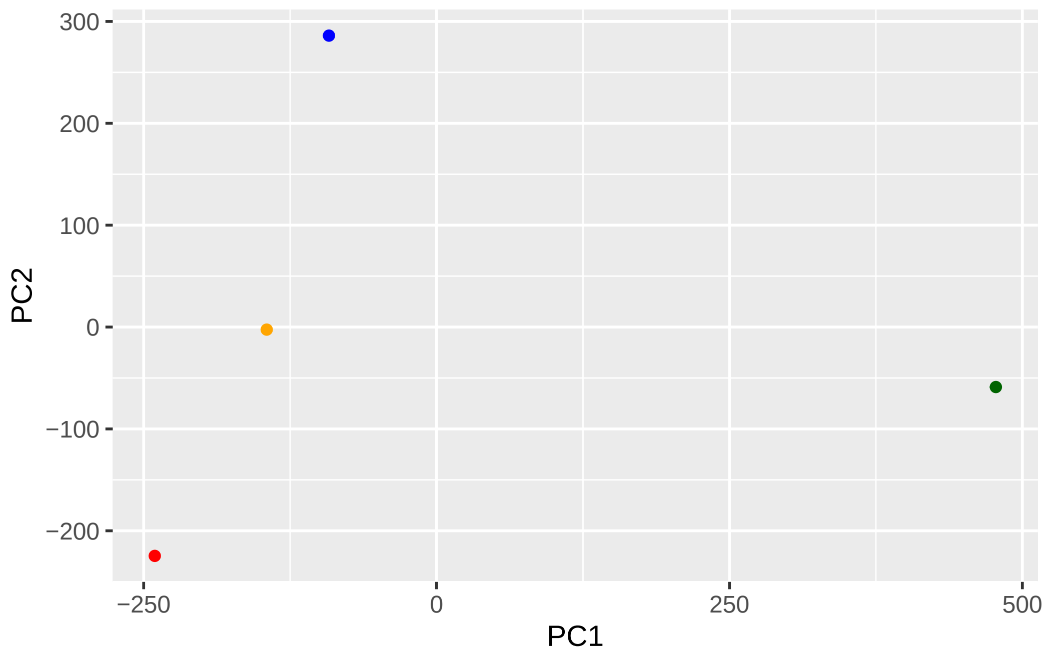
```r
attributes(pca)
```

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

$class
[1] "prcomp"
```

To make one of our 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"…

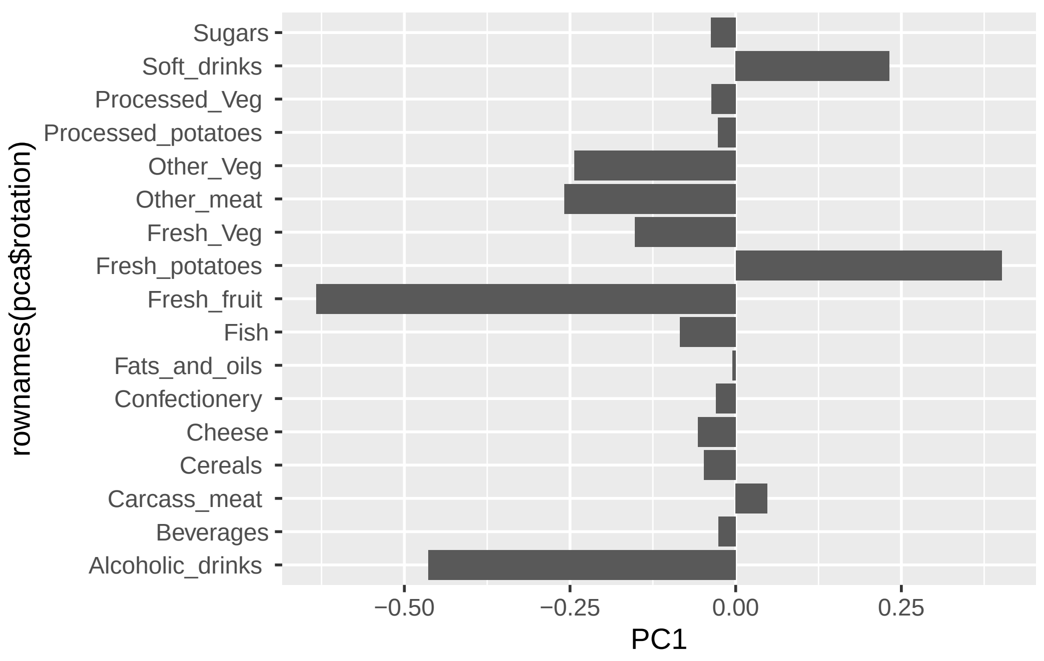
```r
my_cols <- c("orange", "red", "blue", "darkgreen")
```

```r
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”.

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


Sugars
Soft_drinks
Processed_Veg
Processed_potatoes
Other_Veg
Other_meat
Fresh_Veg
Fresh_potatoes
Fresh_fruit
Fish
Fats_and_oils
Confectionery
Cheese
Cereals
Carcass_meat
Beverages
Alcoholic_drinks
rownames(pca$rotation)
−0.50
−0.25
0.00
0.25
PC1