Class 7 Lab

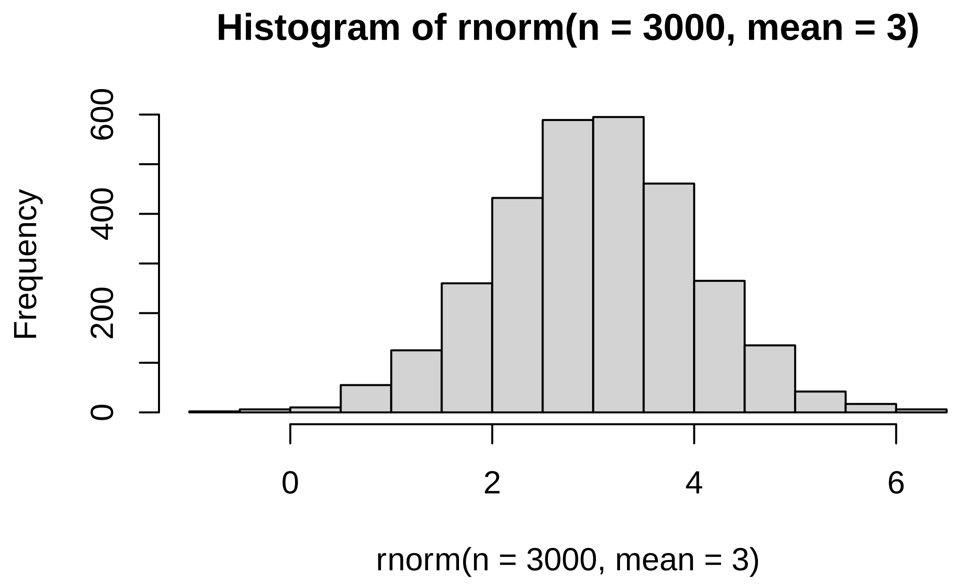
Zainab Fatima (PID: A16880407)

Today we will explore unsupervised machine learning methods including clustering and dimensionality reduction methods.

Let's start by making up some data (where we knoew there are clear groups) that we can use to test out differebt clustering methods.

We can use the `rnorm()` function to help us here.

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



Make data `z` with two "clusters"

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

```
 [1] -4.6003675 -3.9915783 -2.5004397 -3.6699618 -1.7691881 -3.8202792
 [7] -3.1623983 -5.0528424 -3.2103723 -4.8400041 -1.6117839 -1.8304712
[13] -2.5827887 -1.9289335 -0.9275837 -4.0195387 -3.9501986 -2.3590871
[19] -3.7017634 -4.4427436 -3.0781919 -2.5040414 -4.6677744 -3.1378293
[25] -1.9260741 -2.2778718 -2.9752929 -3.6162488 -1.9278957 -1.4613925
```

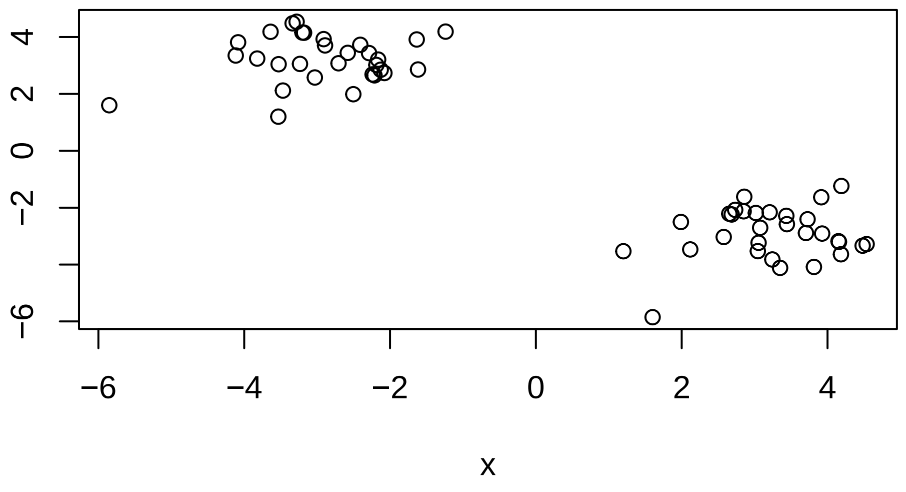
```r
rnorm(30, mean = +3)
```

```
 [1] 3.168859 2.869950 3.776098 2.949435 3.351279 3.860109 2.552469 2.255847
 [9] 2.647273 5.372991 5.161801 2.711285 2.822912 1.661388 1.408402 2.824135
[17] 3.859810 3.114903 4.986921 2.505164 1.391616 2.556659 1.332124 3.253018
[25] 3.993623 4.126187 3.326445 1.093547 3.807944 4.702345
```

```r
 x <- c( rnorm(30, mean= -3), rnorm(30, mean = +3) )

z <- cbind(x = x, rev(x))
#cbind makes columns, stands for "column-bindng"
head(z)
```

```
             x
[1,] -3.638806 4.184377
[2,] -2.163205 3.206731
[3,] -1.238267 4.189877
[4,] -2.077131 2.733840
[5,] -5.850653 1.601154
[6,] -2.891392 3.704191
```

```r
plot(z)
```



Q. How big is dataset z?

```r
nrow(z)
```

```
[1] 60
```

```r
ncol(z)
```

```
[1] 2
```

K-means clustering

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

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

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

Cluster means:
          x
```

```
1  3.230590 -2.903082
2 -2.903082  3.230590

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 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] 44.84541 44.84541
 (between_SS / total_SS =  92.6 %)

Available components:

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

```
#centers = # of clusters, size = we made each cluster have 30
#clustering vector = just a label that says which points are different
```

```
attributes(k)
```

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

$class
[1] "kmeans"
```

> Q. How many points lie in each cluster?

```
k$size
```

```
[1] 30 30
```

> Q. What component of our results tells us about the cluster membership (i.e. which point likes in which cluster)?

```
k$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 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
```

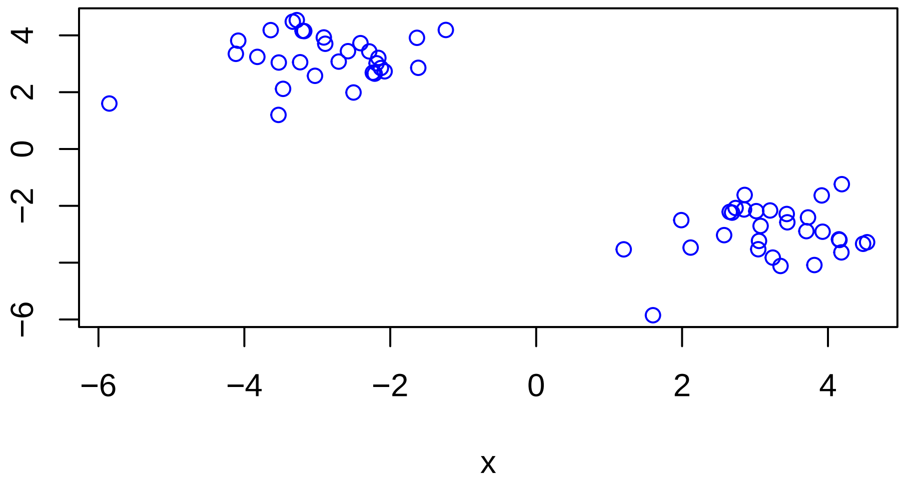
Q. Center of each cluster?

```
k$center
```

```
          x
1  3.230590 -2.903082
2 -2.903082  3.230590
```

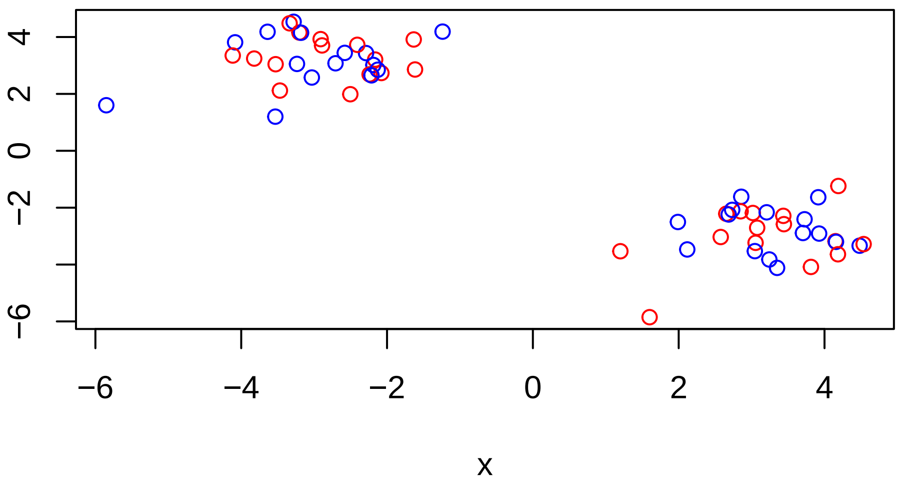
Q. Put this result info together and make a little "base R" plot of our clustering result. Also add the cluster center points to this plot.

```
plot(z, col="blue") #makes a base figure of clustering results but we did not add center poi
```



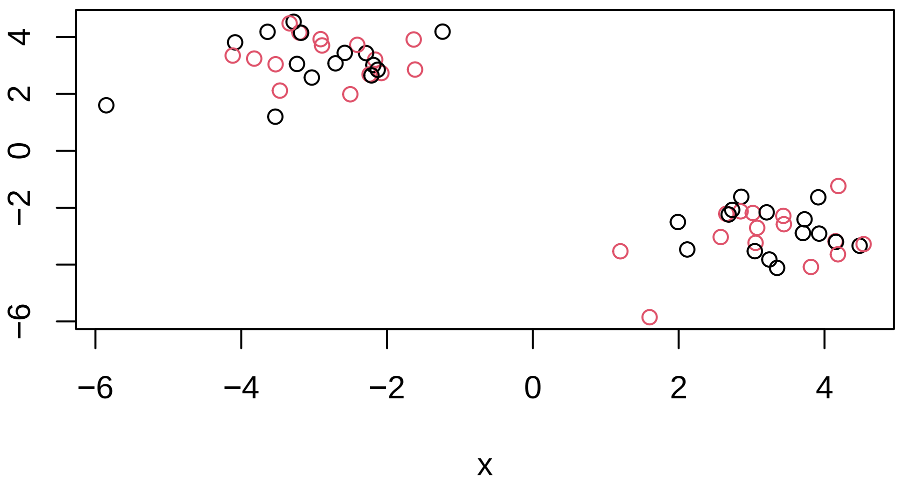
Alternating points can have different colors

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



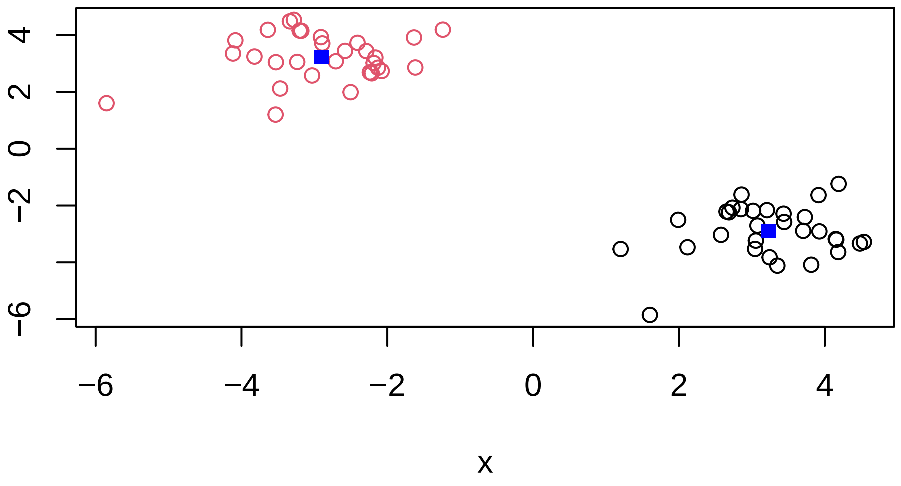
You can color by number.

```r
plot(z, col= c(1,2))
```



Plot colored by cluster membership. To make different clusters with different colors, we can use:

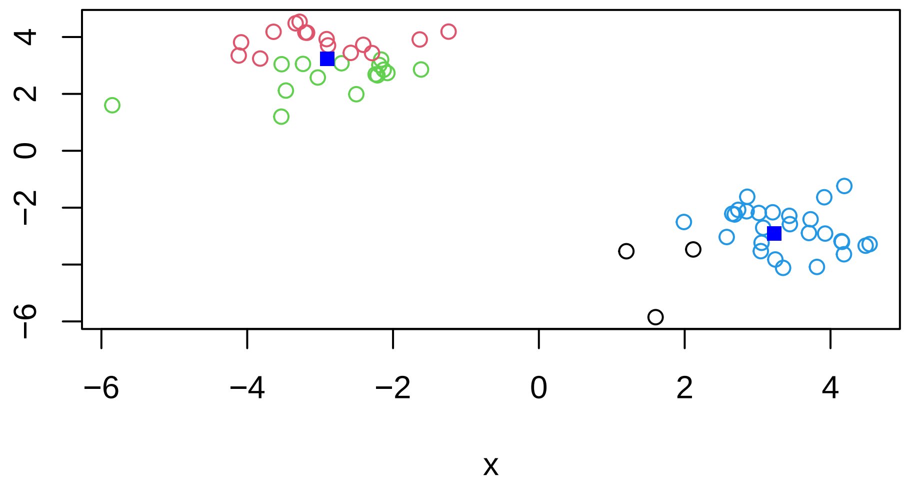
```r
plot(z, col = k$cluster)
points(k$center, col = "blue", pch =15)
```



Q. Run kmeans on our input **z** and define 4 clusters making the same result vizualization plot as above (plot of z colored by cluster membership)

```r
k4 <- kmeans(z, centers = 4)
```

```r
plot (z, col = k4$cluster)
points(k$center, col = "blue", pch =15)
```



```
#to measure how well the clustering was:
k4$tot.withinss
```

```
[1] 61.94827
```

Hierarchical Clustering

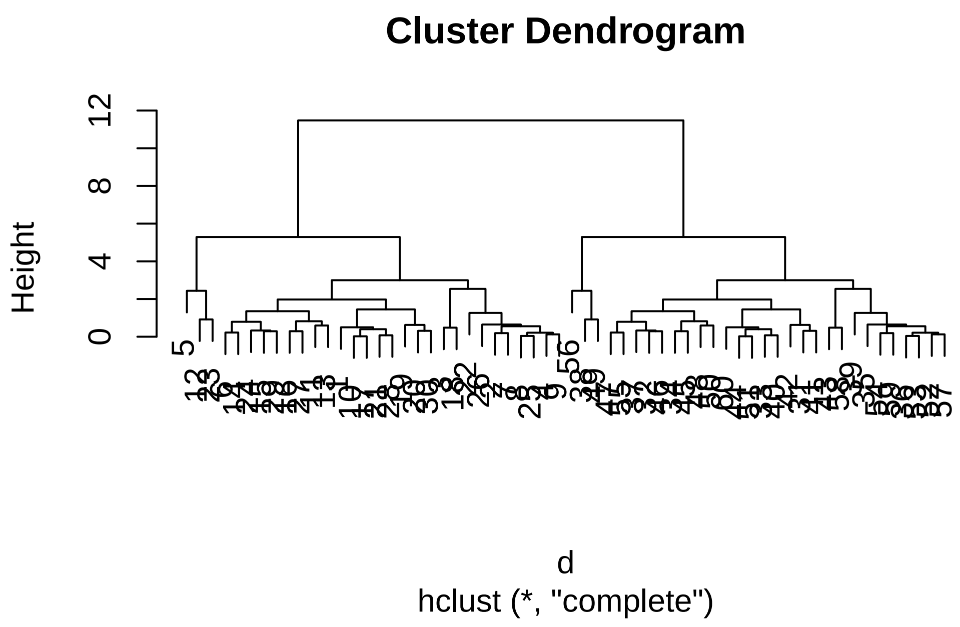
The main function in base R for this is called `hclust()` it will take as input a distance matriz (key point is that you can't just give your raw data as input - you have to first calculate a distance matrix from your data).

```
# distance matrix = distance from each point to every other point
d <- dist(z)
hc <- hclust(d)
hc
```

```


Call:
hclust(d = d)

Cluster method   : complete
```

```
Distance         : euclidean
Number of objects: 60
```

```r
plot(hc)
```

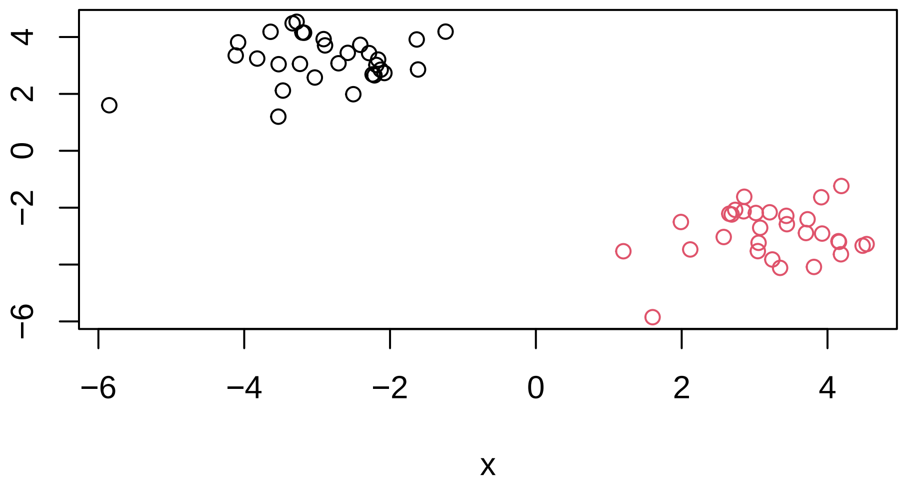


```r
#in the plot, one cluster is 1 - 30, other cluster is 30 - 60
```

Once I inspect the "tree"/dendrogram I can "cut" the tree to yield my groupings or clusters. The function to this is called `cutree()`

```r
# h = height where tree is cut
grps <- cutree(hc, h= 10)
```

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



Hands on with Principal Component Analysis (PCA)

Let's examine some silly 17-dimensional data detailing food consumption in the UK (England, Scotland, Wales, and N. Ireland). Are these countries eating habits different or similar and if so how?

Data Import

```r
url <- "https://tinyurl.com/UK-foods"
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
```

> Q1. How many rows and columns are in your new dataset. Which R functions can you use?

```r
nrow(x)
```

```
[1] 17
```

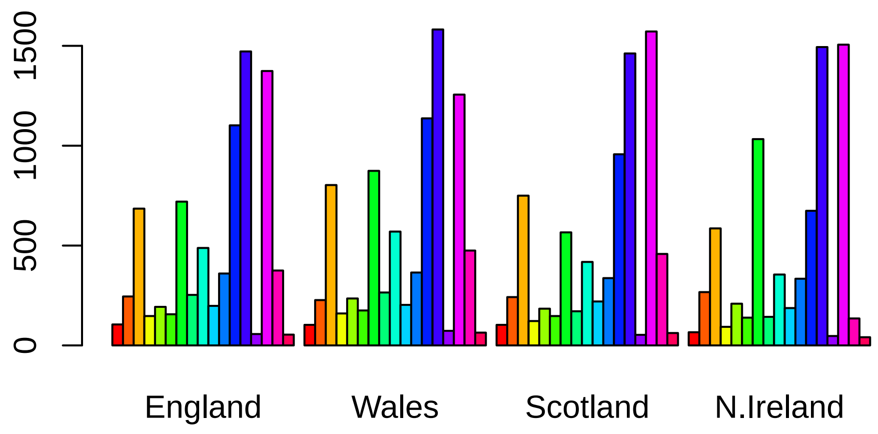
```r
ncol(x)
```

```
[1] 4
```

```r
dim(x)
```

```
[1] 17  4
```

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

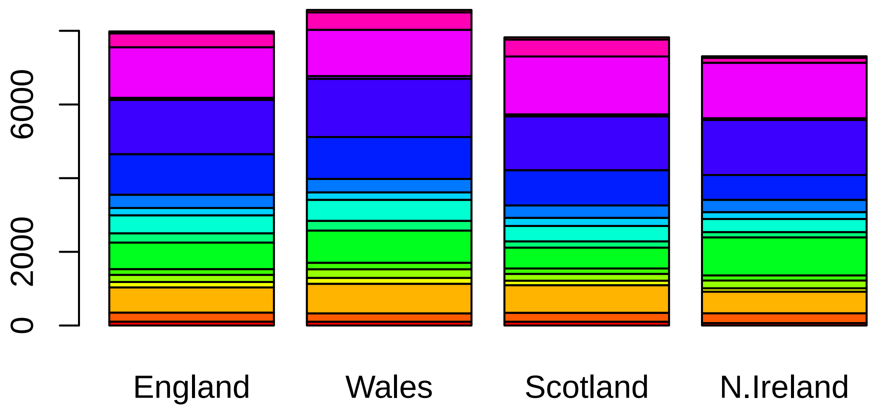


> Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

The second approach—using read.csv(url, row.names=1)—is preferred because it's more concise.

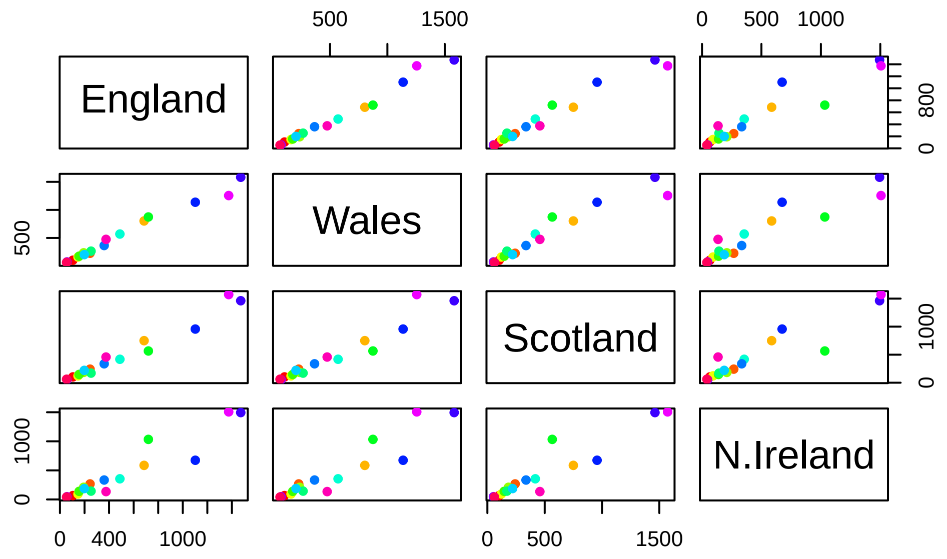
> Q3: Changing what optional argument in the above barplot() function results in the following plot?

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



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?

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



```
#if something is on a straight line, it's similar amount in both countries
```

Looking at these types of "pairwise plots" can be helpful but it does not scale well and is more time-consuming/error-prone. There must be a better way..

PCA to the rescue!

> Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set?

The main function for PCA in base R is called `prcomp`. This function wants the transpose of our input data - i.e. the important food categories in as columns and the countries as rows.

```
#head(t(x)) shows transposed tables
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
```

```
#PC1-4 are the new axis created
# 96% of data captures by PC1 and PC2 so we can plot on PC1 and PC2
```

Let's see what is in our PCA result object `pca`

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

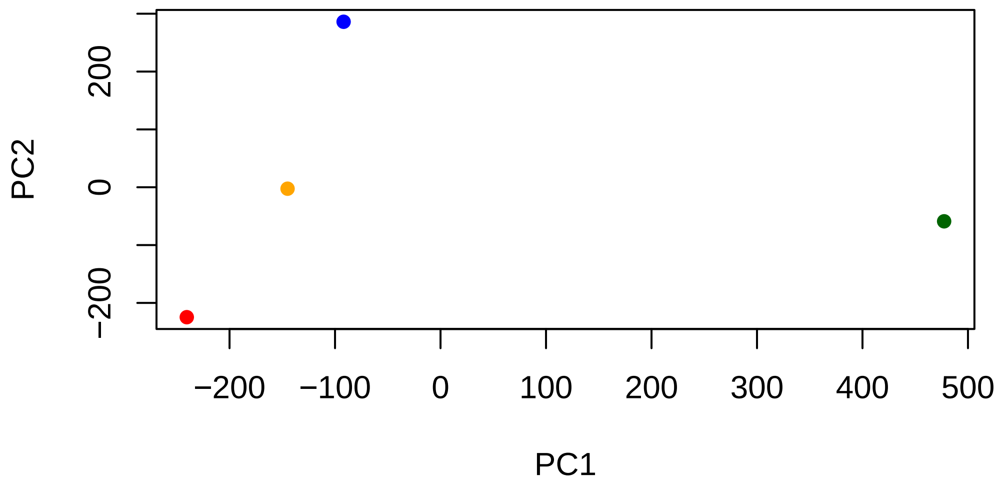
The `pca$x` result object is where we will focus first as this details how the countries are related to each other in terms of our new "axis" (aka "PCs", "eigenvectors", etc.)

```
head(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
```

> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

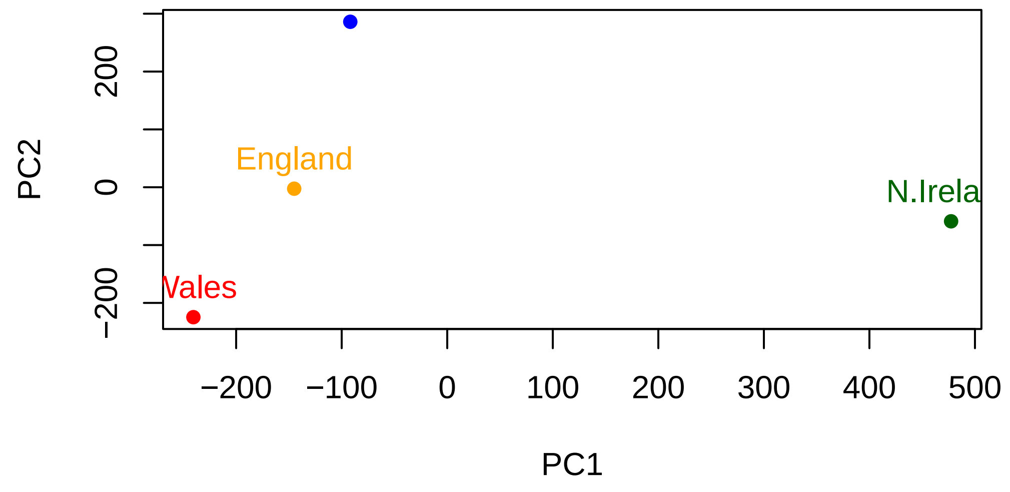
```
plot(pca$x[,1], pca$x[,2], pch = 16,
     col=c("orange", "red", "blue", "darkgreen"),
     xlab = "PC1", ylab = "PC2")
```



```
# england = orange, wales = red, scotland = blue, north ireland = dark green
```

Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
plot(pca$x[,1], pca$x[,2], pch = 16,
     col=c("orange", "red", "blue", "darkgreen"),
     xlab = "PC1", ylab = "PC2")

text(pca$x[,1], pca$x[,2], labels = rownames(pca$x),
     col=c("orange", "red", "blue", "darkgreen"), pos = 3)
```



Below we can use the square of pca$sdev , which stands for "standard deviation", to calculate how much variation in the original data each PC accounts for.

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100)
v
```

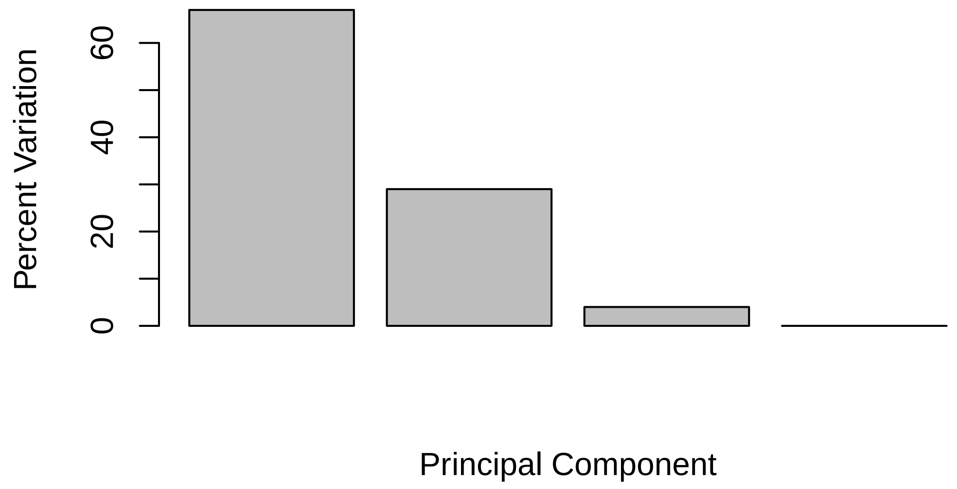
```
[1] 67 29  4  0
```

```
## or the second row here...
z <- summary(pca)
z$importance
```

```
                             PC1       PC2      PC3          PC4
Standard deviation     324.15019 212.74780 73.87622 3.175833e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

Plotting variance with respect to the principal component number

```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```



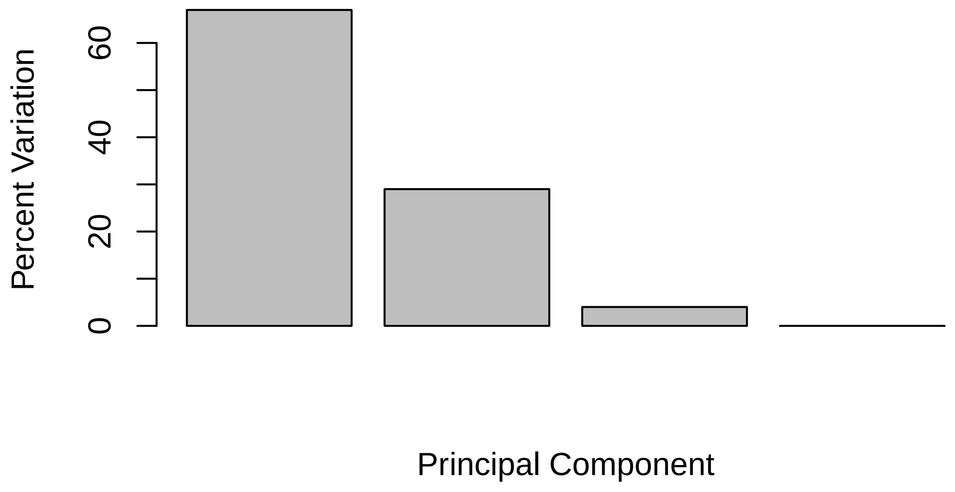
Digging Deeper (variable loadings)

We can look at the so-called PC "loadings" result object to see how the original foods contribute to our new PCs (ie how the original variables contribute to our new better variables).

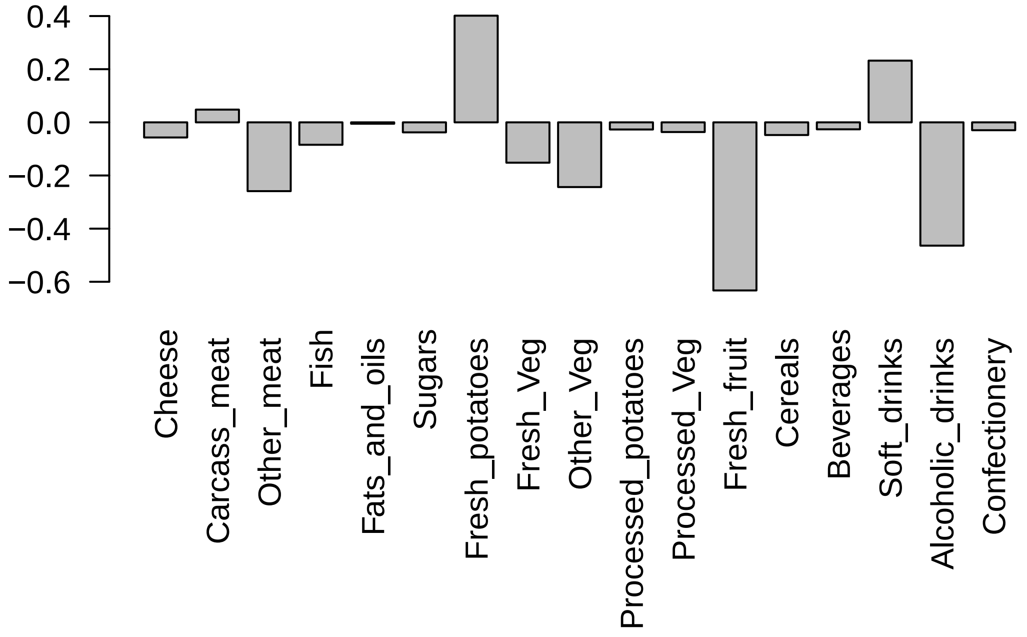
```
pca$rotation[,1] #how much each variable contributes to PCA1
```

```
         Cheese       Carcass_meat         Other_meat               Fish 
   -0.056955380        0.047927628       -0.258916658       -0.084414983 
 Fats_and_oils             Sugars     Fresh_potatoes          Fresh_Veg 
   -0.005193623       -0.037620983        0.401402060       -0.151849942 
      Other_Veg  Processed_potatoes      Processed_Veg        Fresh_fruit 
   -0.243593729       -0.026886233       -0.036488269       -0.632640898 
        Cereals          Beverages        Soft_drinks    Alcoholic_drinks 
   -0.047702858       -0.026187756        0.232244140       -0.463968168 
 Confectionery 
   -0.029650201 
```

```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```

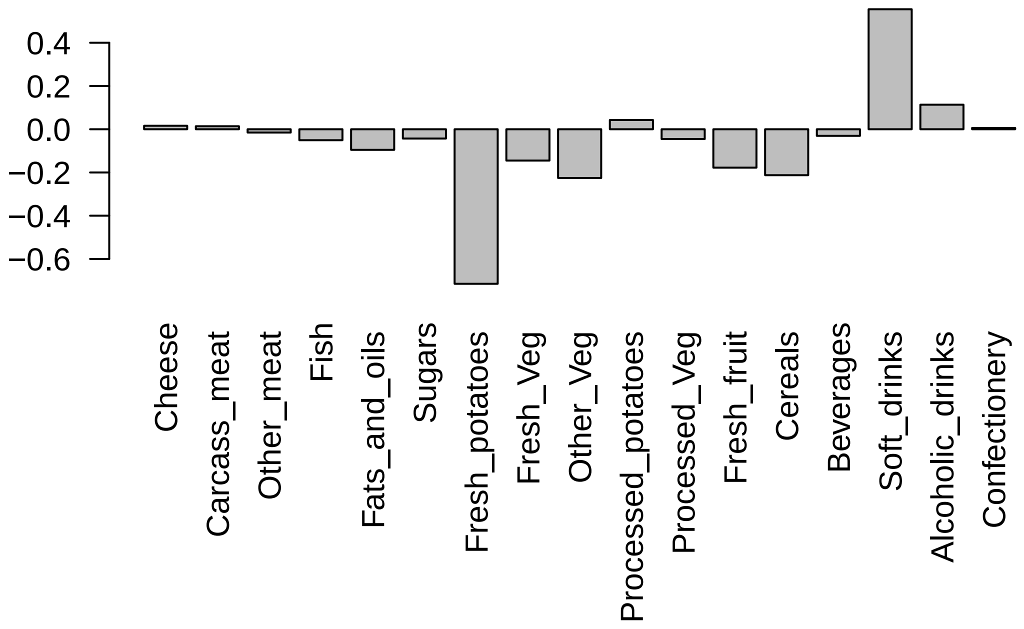


```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



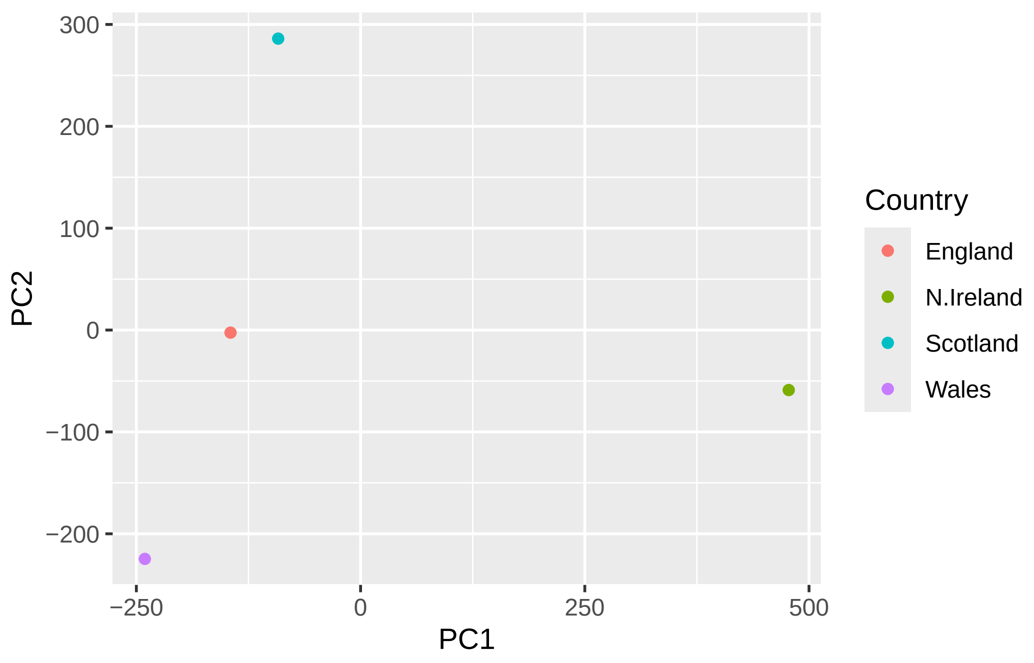
Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about?

```r
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2)
```



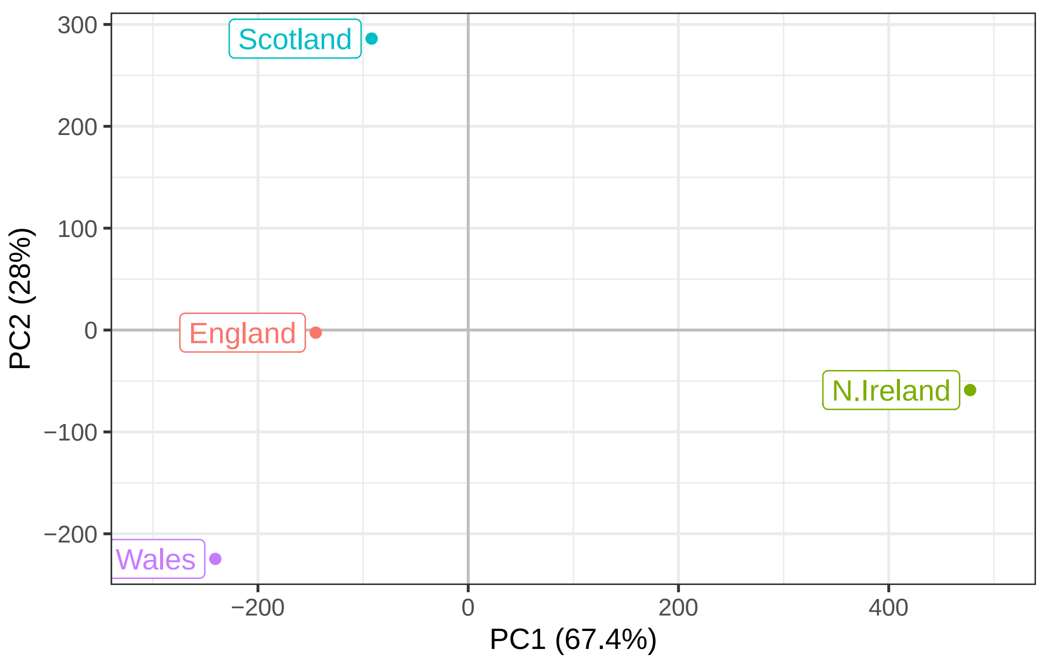
Using ggplot for these figures

```r
library(ggplot2)

df <- as.data.frame(pca$x)
df_lab <- tibble::rownames_to_column(df, "Country")

# Our first basic plot
ggplot(df_lab) +
  aes(PC1, PC2, col=Country) +
  geom_point()
```

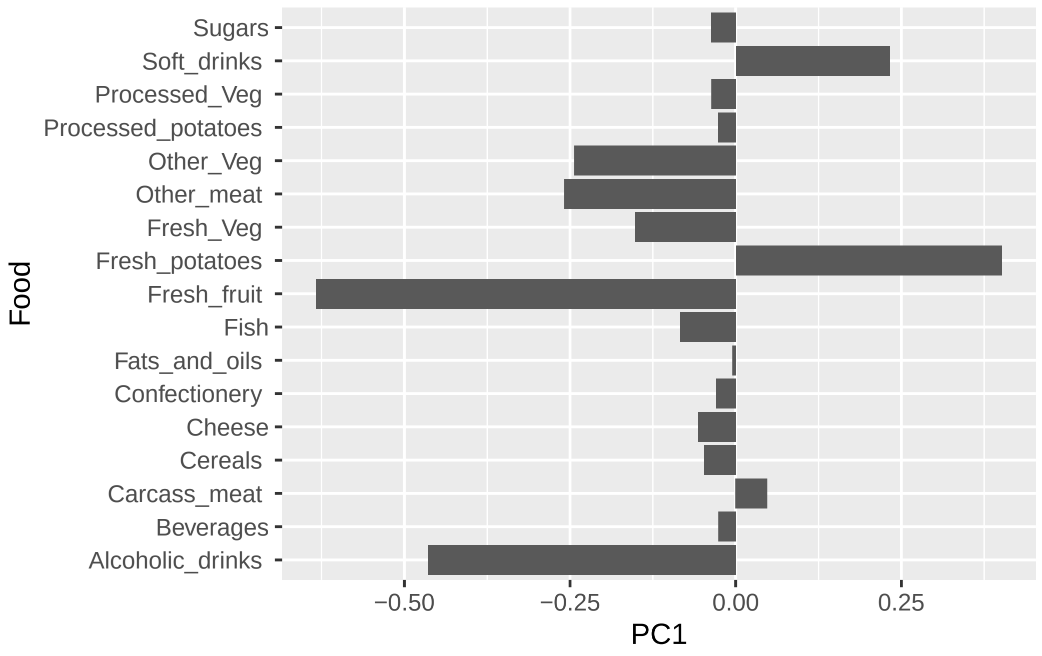


We can make this look nicer.

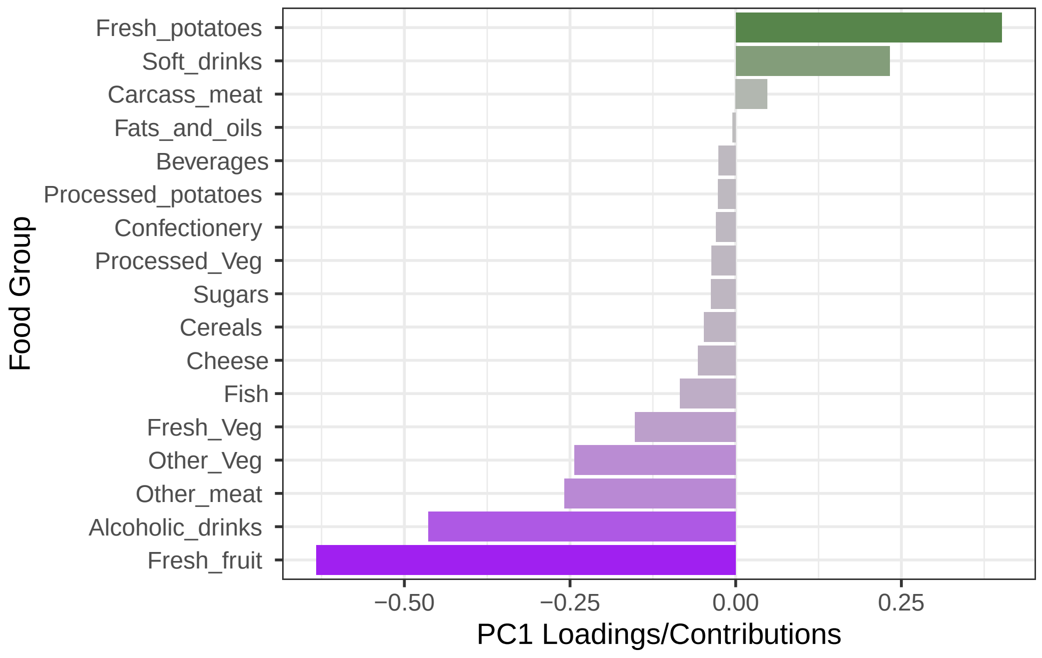
```
ggplot(df_lab) +
  aes(PC1, PC2, col=Country, label=Country) +
  geom_hline(yintercept = 0, col="gray") +
  geom_vline(xintercept = 0, col="gray") +
  geom_point(show.legend = FALSE) +
  geom_label(hjust=1, nudge_x = -10, show.legend = FALSE) +
  expand_limits(x = c(-300,500)) +
  xlab("PC1 (67.4%)") +
  ylab("PC2 (28%)") +
  theme_bw()
```



```r
ld <- as.data.frame(pca$rotation)
ld_lab <- tibble::rownames_to_column(ld, "Food")

ggplot(ld_lab) +
  aes(PC1, Food) +
  geom_col()
```

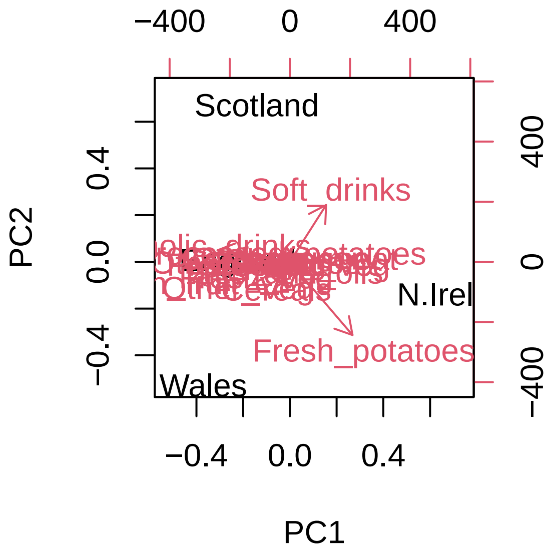


```r
ggplot(ld_lab) +
  aes(PC1, reorder(Food, PC1), bg=PC1) +
  geom_col() +
  xlab("PC1 Loadings/Contributions") +
  ylab("Food Group") +
  scale_fill_gradient2(low="purple", mid="gray", high="darkgreen", guide=NULL) +
  theme_bw()
```



Biplots

```r
## The inbuilt biplot() can be useful for small datasets
biplot(pca)
```


−400
0
400
Scotland
Soft_drinks
Fresh_potatoes
N.Irel
Wales
PC2
PC1
−0.4
0.0
0.4