Machine Learning pt. 1

Kelly_F

10/22/2021

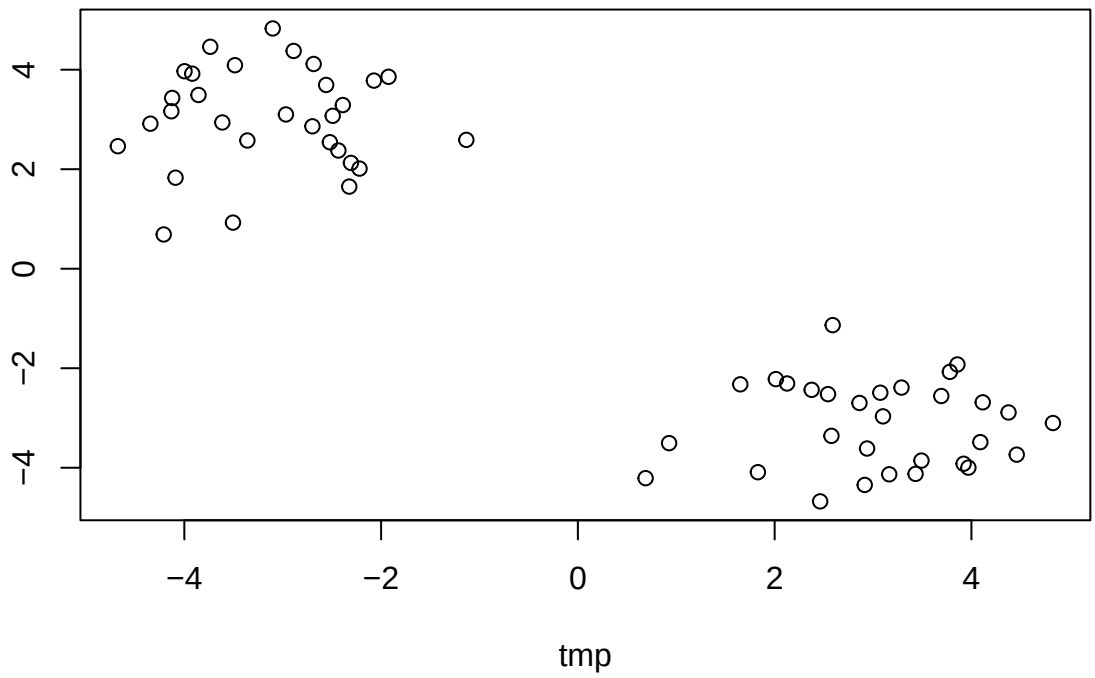
Clustering Methods

1. k-means clustering: R function is "kmeans()".

Test data used below to learn use of kmeans() function.

```
# Create Test Data
tmp <- c(rnorm(30, 3), rnorm(30, -3))
data <- cbind(tmp, rev(tmp)) #two columns, with second column the reverse of tmp vector

# Plot data, which was constructed to have two clear clusters of data.
plot(data)
```



Next, run 'kmeans()' clustering w/ k set to 2, nstart 20.

A condition of kmeans is that you must tell it how many clusters you want.

Note: "clustering vector"(output) Tells you which cluster each element of the data belongs to.

```
# Run kmeans
km_data <- kmeans(data, centers=2, nstart=20)
km_data
```

```
## K-means clustering with 2 clusters of sizes 30, 30
##
## Cluster means:
##         tmp
## 1  3.038177 -3.125645
## 2 -3.125645  3.038177
##
## 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 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
##
## Within cluster sum of squares by cluster:
## [1] 51.85426 51.85426
##  (between_SS / total_SS =  91.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Q. How many points are in each cluster? Tip: use "Value" section of help document.

30 points in each cluster.

```
km_data$size
```

```
## [1] 30 30
```

Q. What "component" of your results object details cluster assigment/membership?

```
km_data$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 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
```

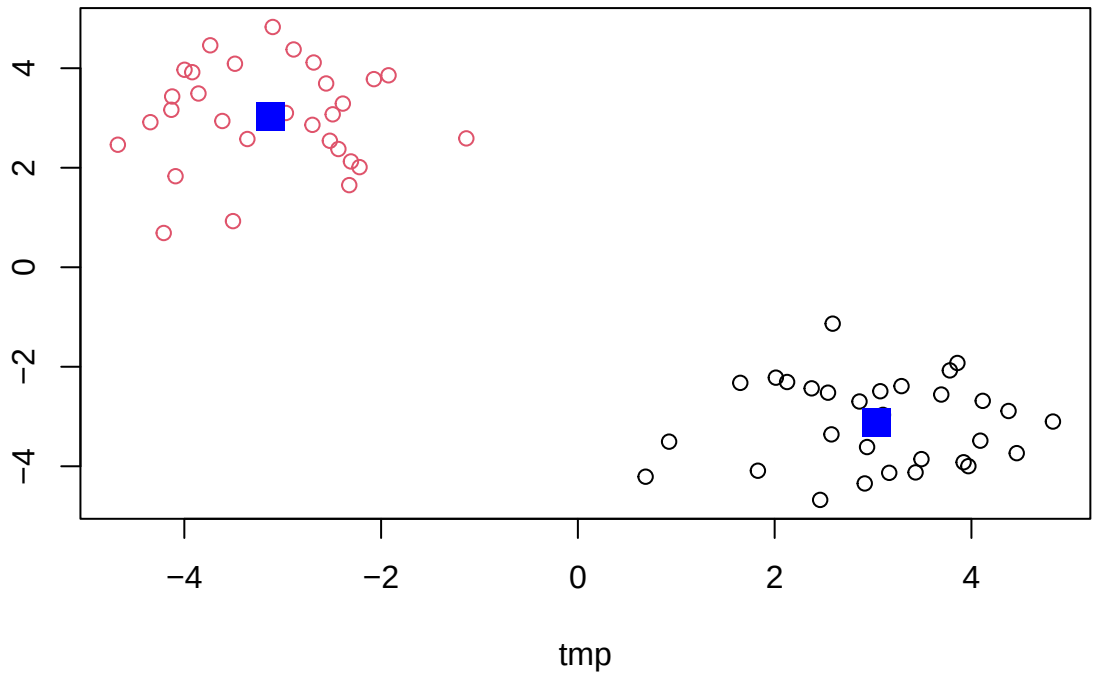
Q. What "compoennet" of your results object details cluster center?

```
km_data$centers
```

```
##         tmp
## 1  3.038177 -3.125645
## 2 -3.125645  3.038177
```

Q. Plot x colored by the kmeans cluster assignemnt and add cluster centers as blue points.

```
plot(data, col=km_data$cluster)
points(km_data$centers, col="blue", pch=15, cex=2)
```



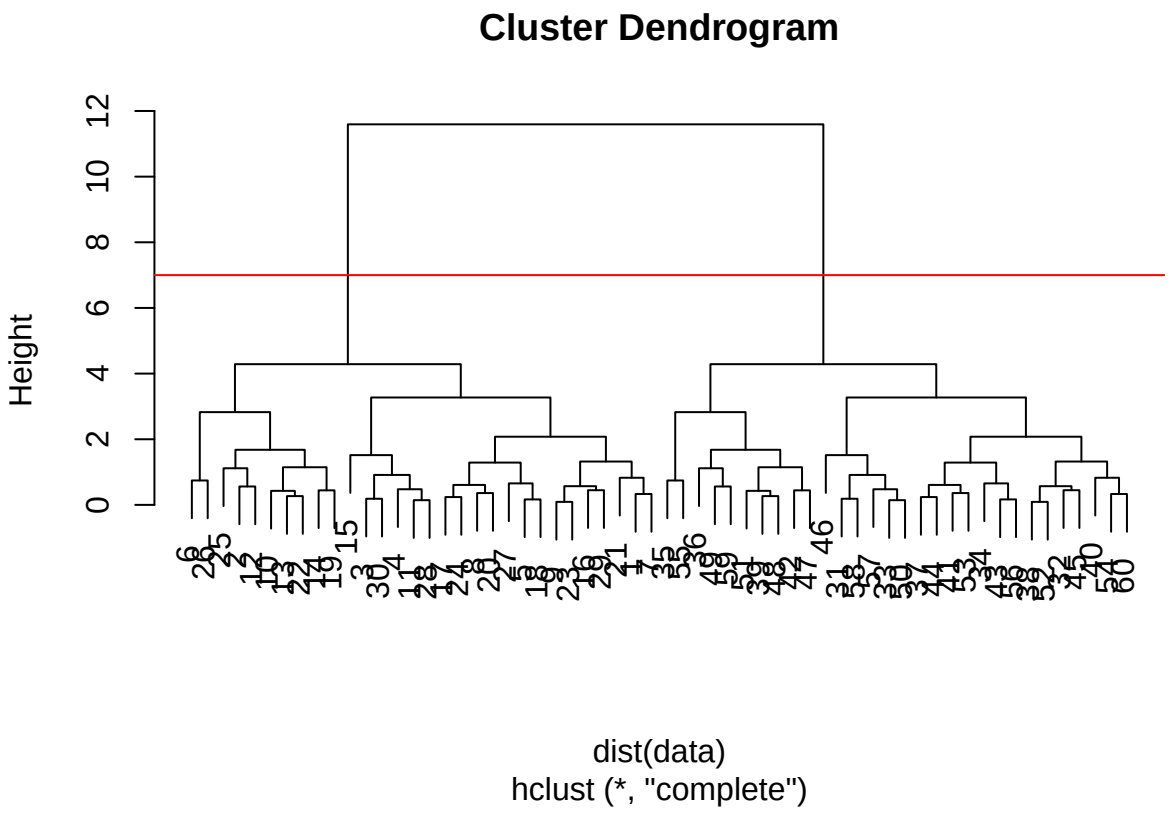
Hierarchical Clustering

We will use the 'hclust()' function on the same data as kmeans example to see how this method works.

```
# hclust needs a distance matrix as an input
hc <- hclust(dist(data))
hc
```

```
##
## Call:
## hclust(d = dist(data))
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 60
```

```
# Plot Dendrogram to investigate between-cluster differences.
# Further distance on the dendrogram = more dissimilar
plot(hc)
abline(h=7, col="red")
```



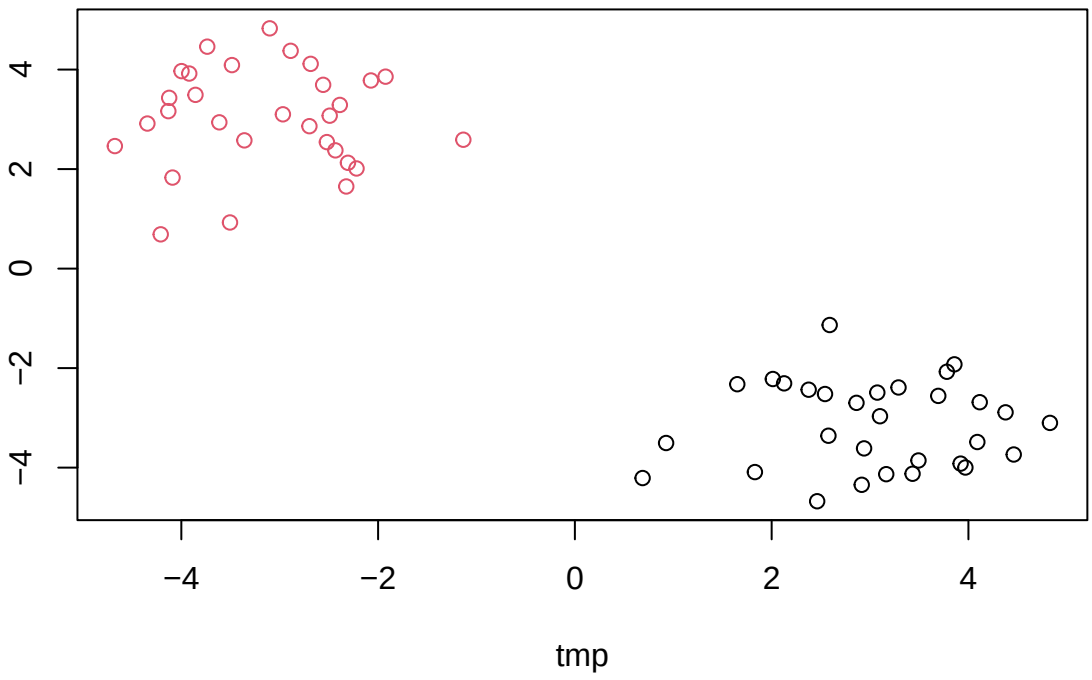
To find our membership vector we need to "cut" the tree into its respective branches (clusters). For this we will use the 'cutree()' function and tell it the height to cut at.

```
# Cut at height=7
cutree(hc, h=7)
```

```
##  [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
```

```
# Alternatively, we can instead tell 'cutree()' how many clusters we want.
grps <- cutree(hc, k=2)

# Plot data and color by hclust grouping
plot(data, col=grps)
```



Kmeans recap

- clusters data, but must tell it how many centers you want.
- functions needs euclidean distances.

Hclust recap

- doesn't take raw data. Must give it a distance matrix.
- doesn't require euclidean distances as input.

Principal Component Analysis w/ UK Food Data

Analysis goal: is the diet composition across the 4 countries of interest different?

```
# Import data
url <- "https://tinyurl.com/UK-foods"
uk_food <- read.csv(url, row.names = 1)

# Determine the number of rows and columns in the dataframe
dim(uk_food)
```

```
## [1] 17  4
```

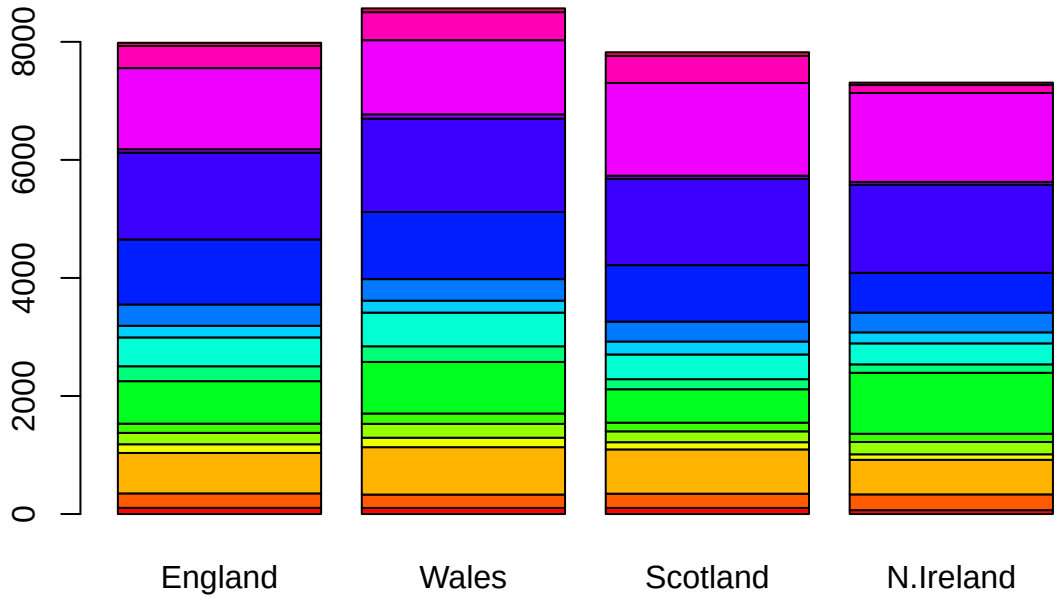
Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

Initially, 17 rows, 5 columns. You can use 'dim()' function to determine this. Note, there are only 4 countries, so when .csv is read in, you must tell it that rownames are in the first column, row.names=1.

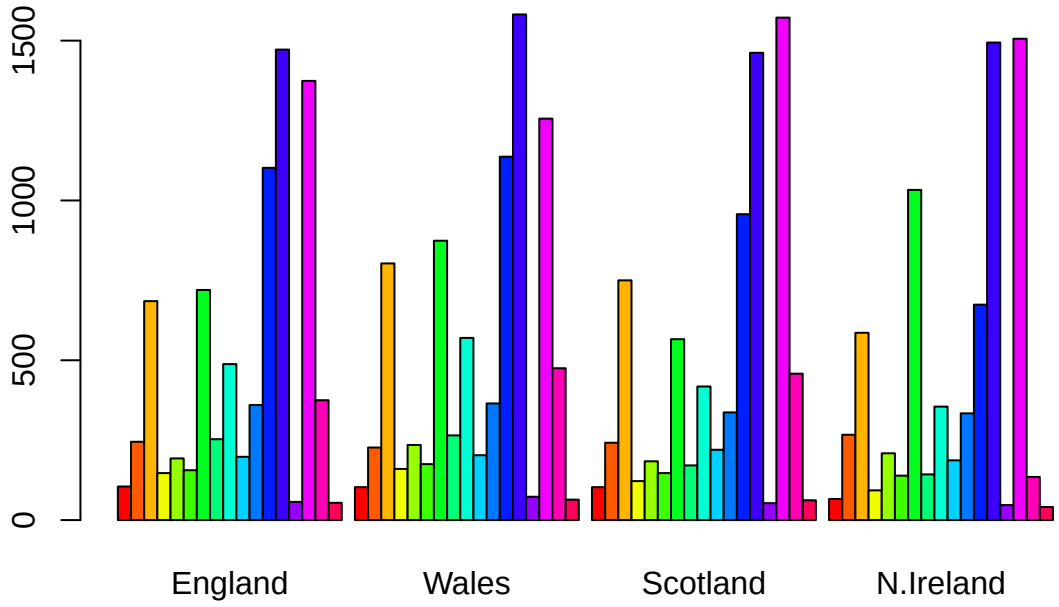
Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

The row.names option when reading in the .csv, because it doesnt overwrite or obstruct the file you are importing.

```
# Start to visualize data
barplot(as.matrix(uk_food), col=rainbow(17))
```

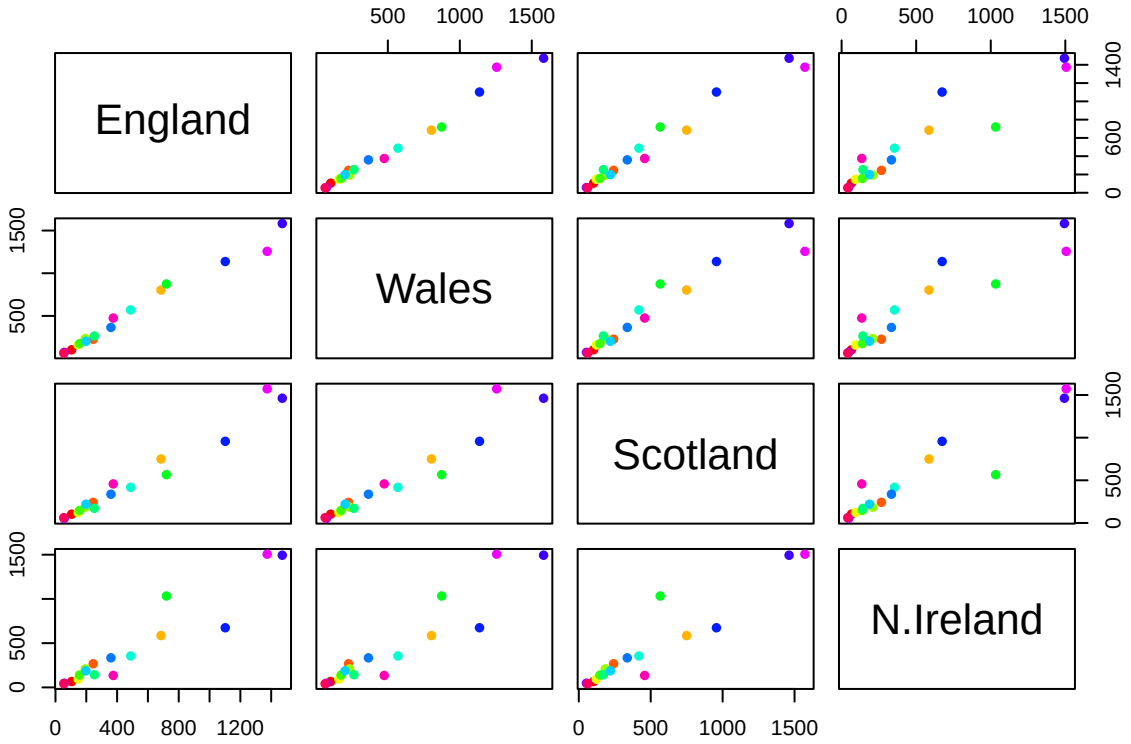



```
barplot(as.matrix(uk_food), beside=T, col=rainbow(nrow(uk_food))) # plot each food category separately
```




```
mycols <- rainbow(nrow(uk_food)) #generate colors # of food categories

# Plot all possible pairwise correlation plots
pairs(uk_food, col=mycols, pch=16) #alternatively can set col=rainbow(17). Accomplishes the same thing
```



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

From the limited analysis we've completed, it seems that N. Ireland has lower diversity in what they eat compared to other countries. I.e., there are a few predominant food groups w/ high frequency, and the remaining food groups have lower frequency.

PCA to the rescue!

Here we will use the base R function for PCA, which is called 'prcomp()'.

```
# Run PCA
#As we noted in the lecture portion of class, prcomp() expects the observations to be rows and the vari

pca <- prcomp(t(uk_food)) # transpose dataframe for PCA analysis
summary(pca)
```

```
## Importance of components:
##                             PC1      PC2      PC3          PC4
## Standard deviation     324.1502 212.7478 73.87622 4.189e-14
## Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
## Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

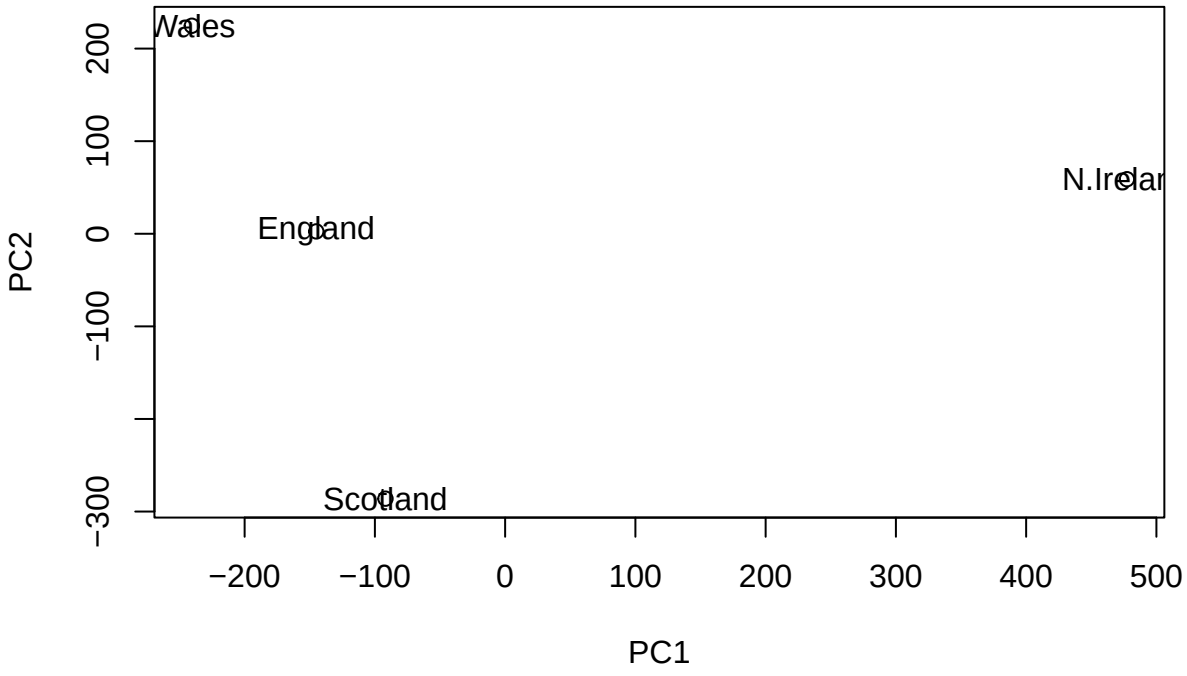
```
attributes(pca)
```

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

```
pca$x
```

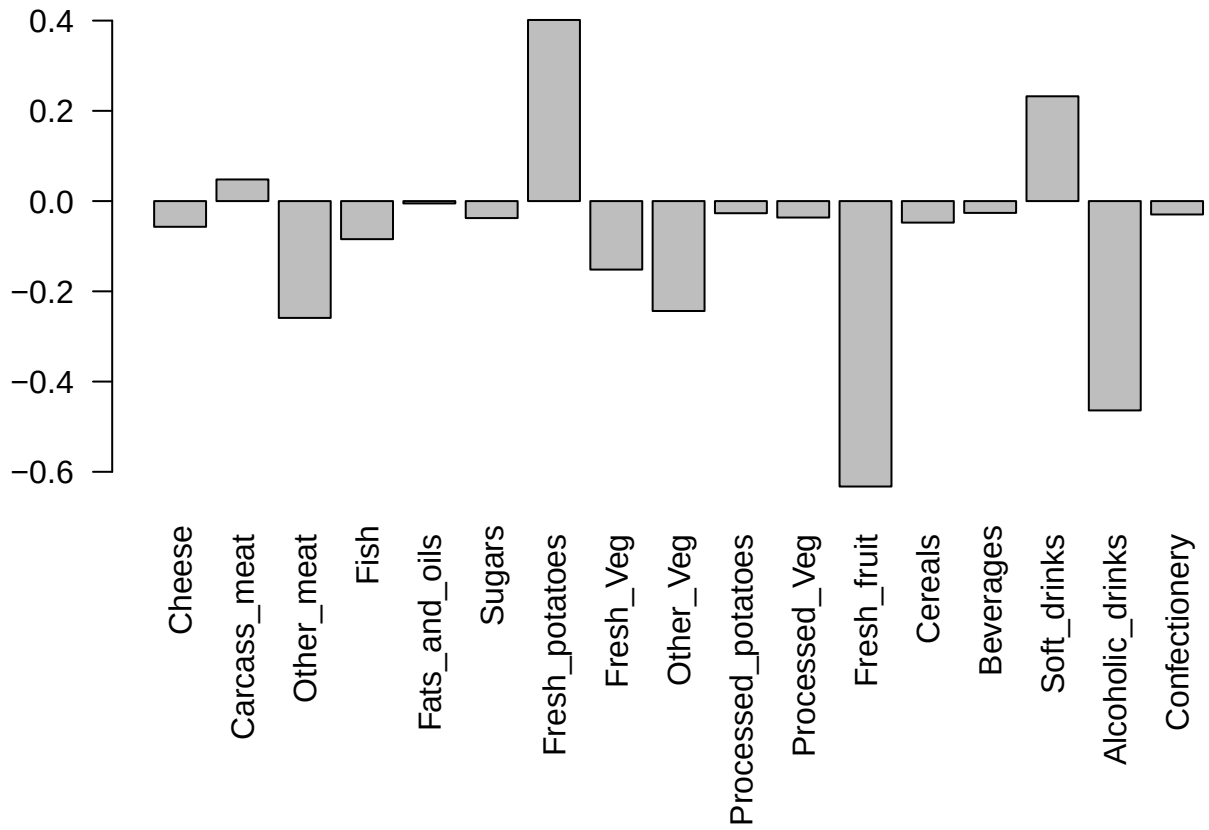
```
##                   PC1         PC2         PC3           PC4
## England   -144.99315    2.532999 -105.768945  2.842865e-14
## Wales     -240.52915  224.646925   56.475555  7.804382e-13
## Scotland   -91.86934 -286.081786   44.415495 -9.614462e-13
## N.Ireland  477.39164   58.901862    4.877895  1.448078e-13
```

```
# Plot PCA
plot(pca$x[, 1:2])#plot PCA1 & 2 for each country
text(pca$x[,1], pca$x[,2], colnames(uk_food))
```



We can also examine the PCA "loadings" and investigate how much each individual food group contributes to each PC

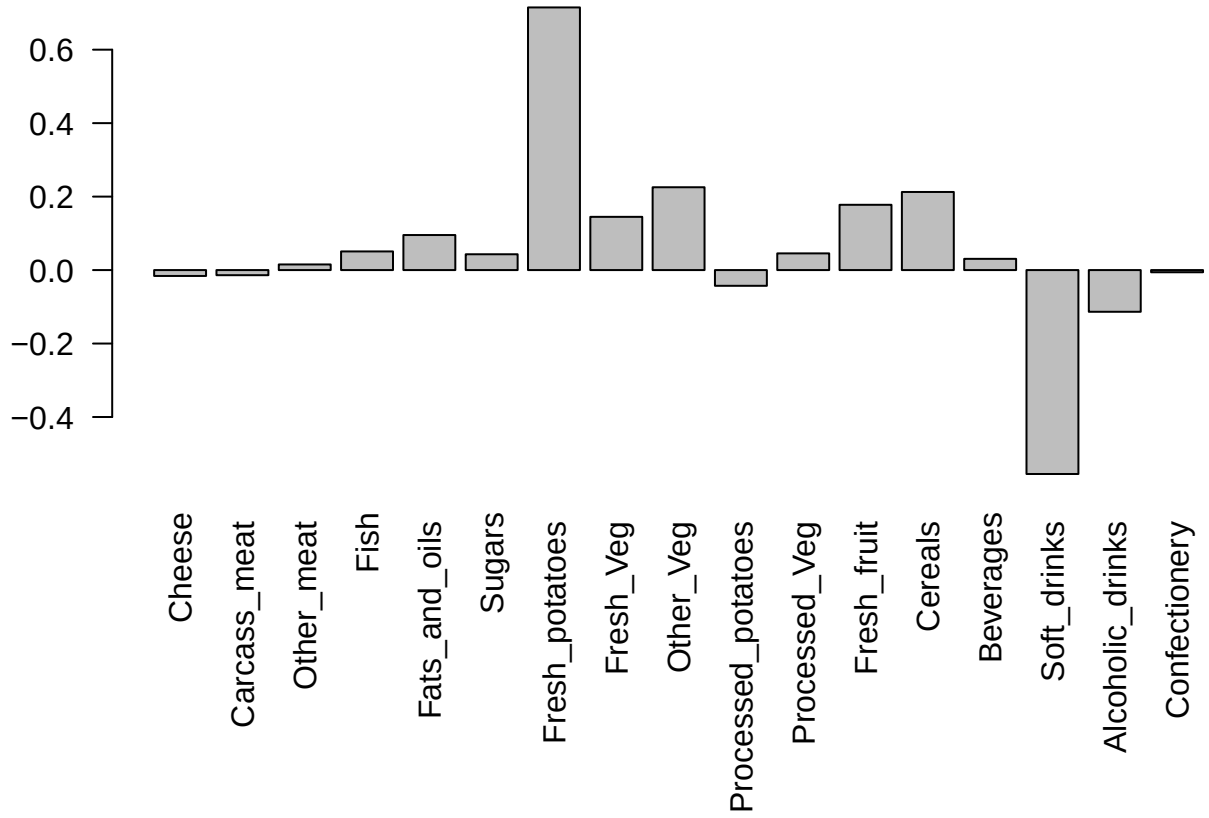
```
#Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot(pca$rotation[, 1], las=2)
```



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

PC2 mainly tells us about differences between Wales, Engalnd, and Scottland. Wales has a high feature count for processed potatoes VS Scottland which has a high feature count for soft drinks.

```
# Plot for PC2
par(mar=c(10, 3, 0.35, 0))
barplot(pca$rotation[, 2], las=2)
```



PCA of RNA-seq data

```
url2 <- "https://tinyurl.com/expression-CSV"
rna.data <- read.csv(url2, row.names=1)
head(rna.data)
```

```
##        wt1 wt2  wt3  wt4 wt5 ko1 ko2 ko3 ko4 ko5
## gene1  439 458  408  429 420  90  88  86  90  93
## gene2  219 200  204  210 187 427 423 434 433 426
## gene3 1006 989 1030 1017 973 252 237 238 226 210
## gene4  783 792  829  856 760 849 856 835 885 894
## gene5  181 249  204  244 225 277 305 272 270 279
## gene6  460 502  491  491 493 612 594 577 618 638
```

Q10: How many genes and samples are in this data set?

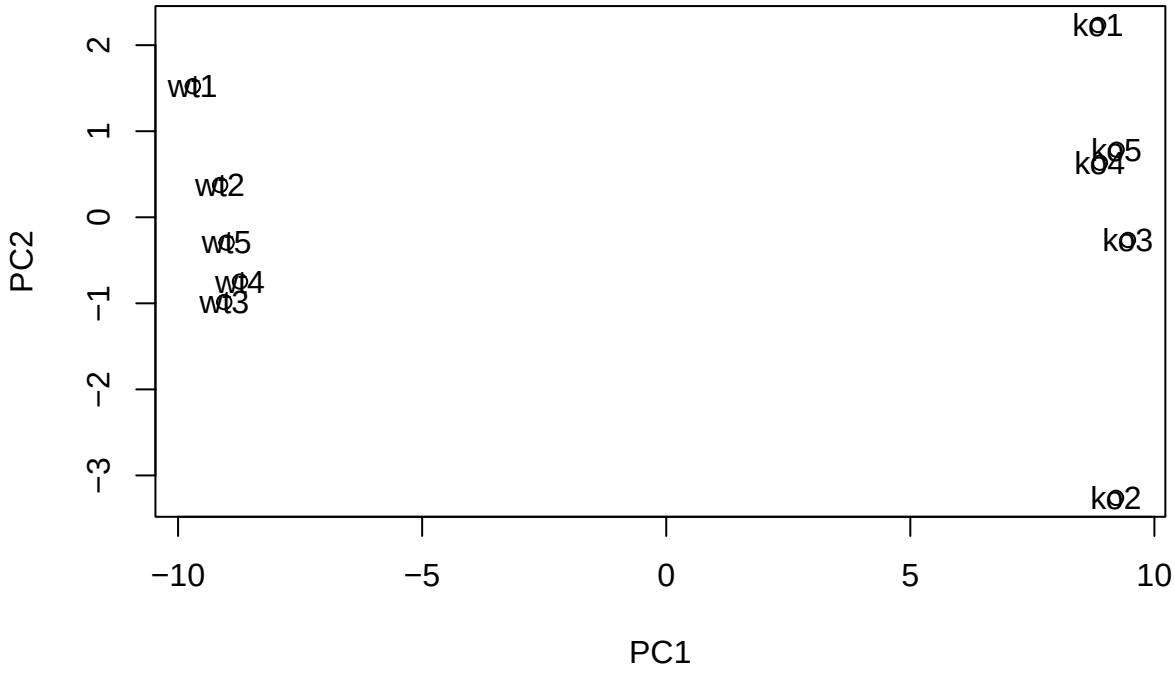
100 genes and 10 samples

```
dim(rna.data)
```

```
## [1] 100  10
```

```
# Generate PCA
pca_rna <- prcomp(t(rna.data), scale=TRUE) #transpose data before running PCA analysis

# Simple un polished plot of pc1 and pc2
plot(pca_rna$x[,1], pca_rna$x[,2], xlab="PC1", ylab="PC2")
text(pca_rna$x[, 1:2], labels= colnames(rna.data))
```




```
#Investigate what proportion of variance PC1 explains
summary(pca_rna)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     9.6237 1.5198 1.05787 1.05203 0.88062 0.82545 0.80111
## Proportion of Variance 0.9262 0.0231 0.01119 0.01107 0.00775 0.00681 0.00642
## Cumulative Proportion  0.9262 0.9493 0.96045 0.97152 0.97928 0.98609 0.99251
##                           PC8     PC9      PC10
## Standard deviation     0.62065 0.60342 3.348e-15
## Proportion of Variance 0.00385 0.00364 0.000e+00
## Cumulative Proportion  0.99636 1.00000 1.000e+00
```