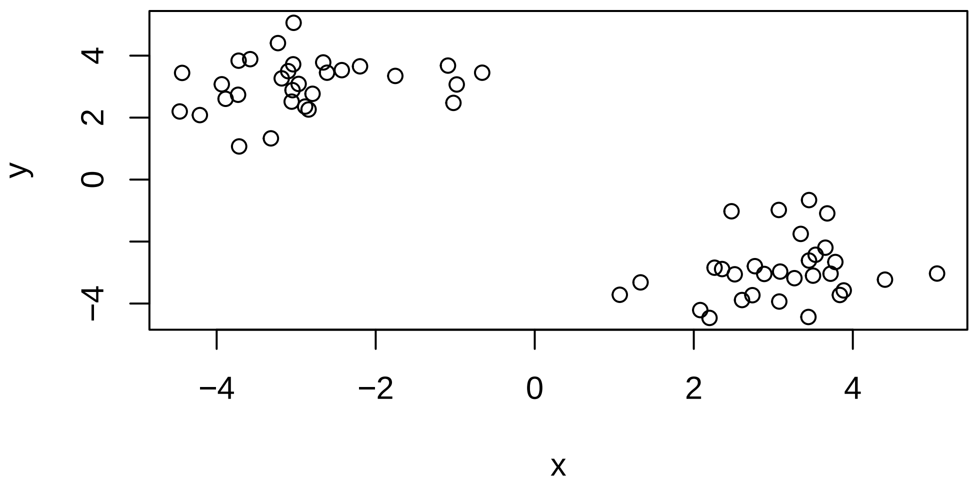
Lab 7

K-means Clustering

Let's make up some data to cluster.

```r
tmp <- c(rnorm (30, -3), rnorm(30,3))
x <- cbind(x= tmp, y= rev(tmp))
plot(x)
```



The function to do k-means clustering in base R is called 'kmeans()'.

```r
kmeans(x, centers=2)
```

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

Cluster means:
          x         y
1  3.083912 -2.919432
2 -2.919432  3.083912

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] 49.82914 49.82914
 (between_SS / total_SS =  91.6 %)

Available components:

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

```r
km <- kmeans(x, centers = 4, nstart=20)
km
```

```
K-means clustering with 4 clusters of sizes 18, 18, 12, 12

Cluster means:
          x         y
1 -3.493336  2.716752
2  2.716752 -3.493336
3  3.634653 -2.058575
4 -2.058575  3.634653

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

Within cluster sum of squares by cluster:
[1] 15.14244 15.14244 13.79891 13.79891
```

```
 (between_SS / total_SS =  95.1 %)

Available components:

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

##Q. What 'component' of your results object details - cluster size - cluster assignment/membership - cluster center

```
#will provide cluster size
km$size
```

```
[1] 18 18 12 12
```

```
#will provide cluster center
km$center
```

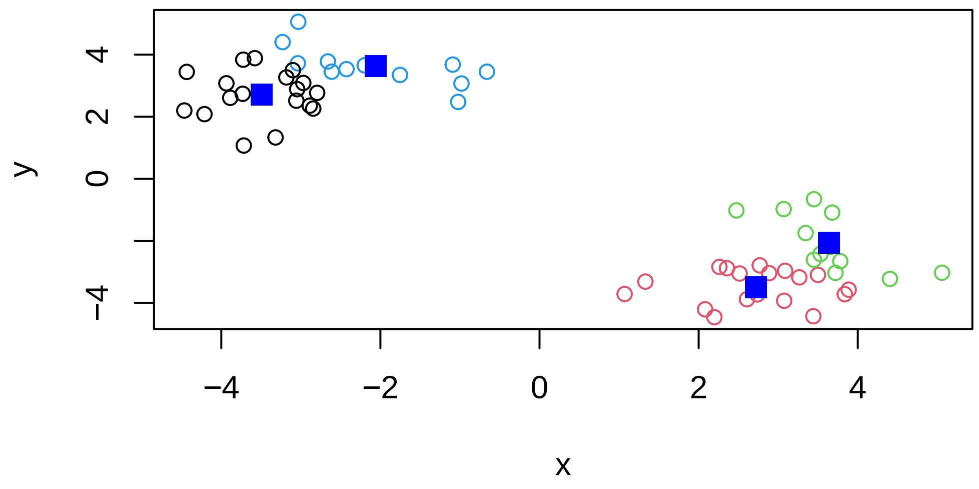
```
          x         y
1 -3.493336  2.716752
2  2.716752 -3.493336
3  3.634653 -2.058575
4 -2.058575  3.634653
```

```
#Membership vector
km$cluster
```

```
 [1] 4 1 4 1 1 4 1 4 1 1 4 4 4 1 1 4 1 1 4 1 1 1 4 1 1 1 1 4 1 4 3 2 3 2 2 2 2 3
[39] 2 2 2 3 2 2 3 2 2 3 3 3 2 2 3 2 3 2 2 3 2 3
```

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

```
plot(x, col=km$cluster)
points(km$centers, col="blue", pch=15, cex=1.5)
```



hclust()

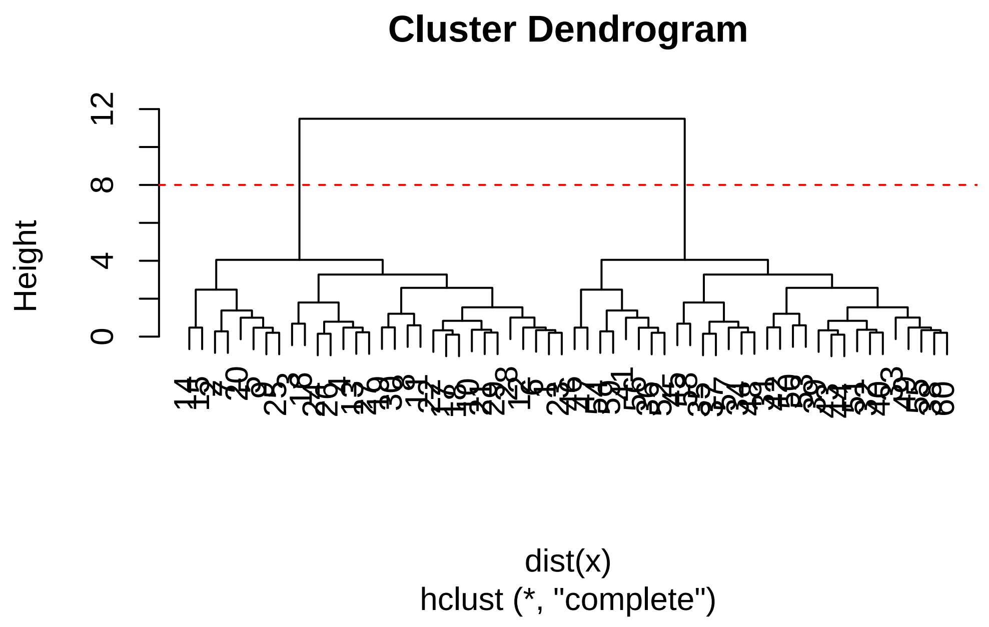
Hierarchical Clustering

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

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

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

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



To get my "main" result (cluster membership), I want to

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

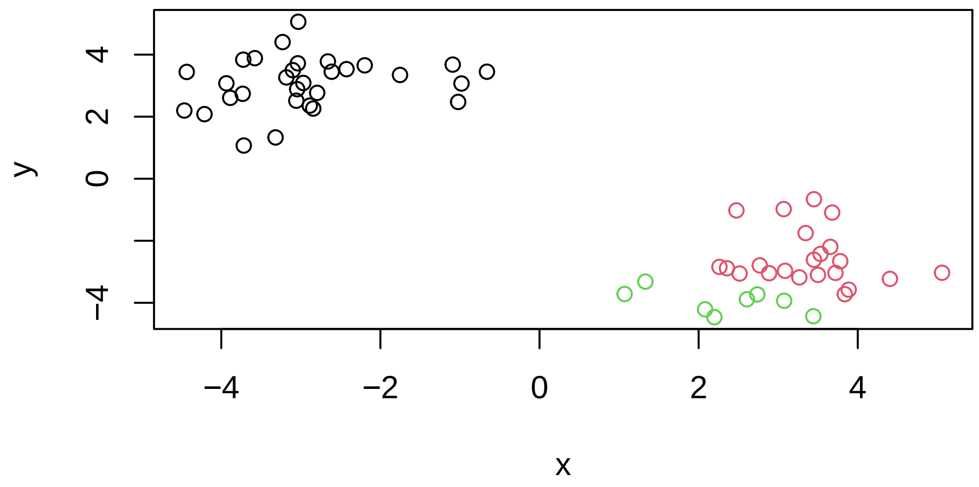
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

More often we will use 'cutree()' with k=2 for example

```
grps = cutree(hc, k=3)
```

Make a plot of our 'hclust()' results i.e. our data colored by cluster assignment

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



Principal Component Analysis (PCA)

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url)
```

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

```
nrow(x)
```

```
[1] 17
```

```
ncol(x)
```

```
[1] 5
```

There are 17 rows and 5 columns.

```r
View(x)
head(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
```

```r
tail(x)
```

```
                   X England Wales Scotland N.Ireland
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
```

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

```r
dim(x)
```

```
[1] 17  4
```

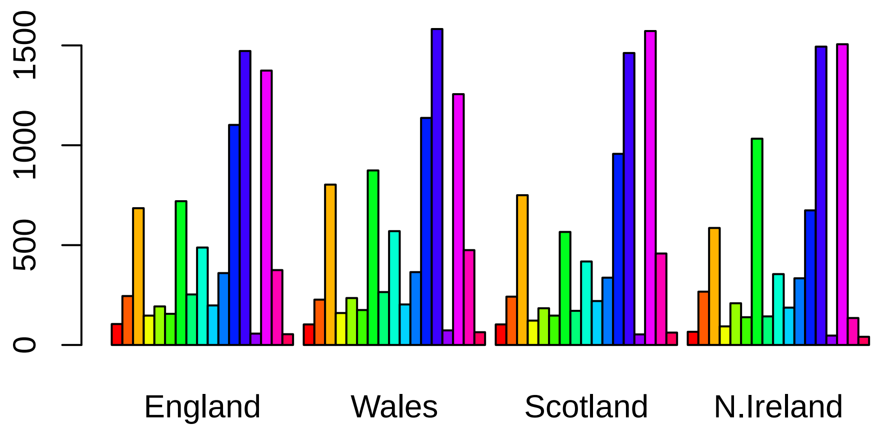
```
x <- read.csv(url, row.names=1)
head(x)
```

```
               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
```

##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 (x <- read.csv(url, row.names=1)) because it is a single step and easier to remember. They are both equally robust if the dataset is the same.

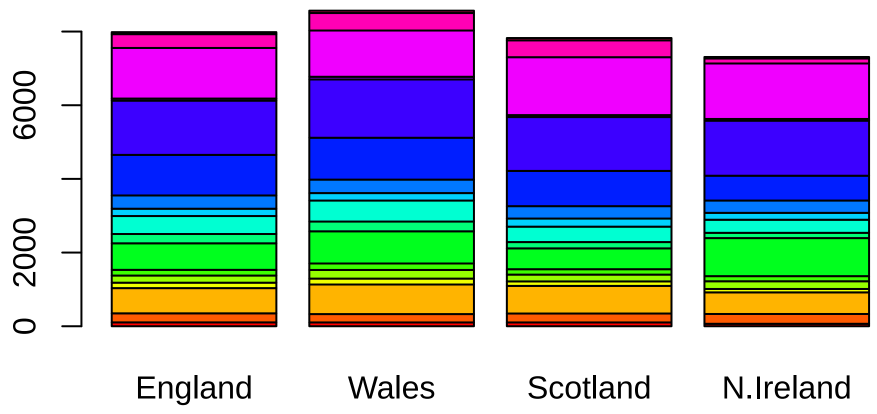
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

Changing beside=TRUE to beside=FALSE will provide the following plot.

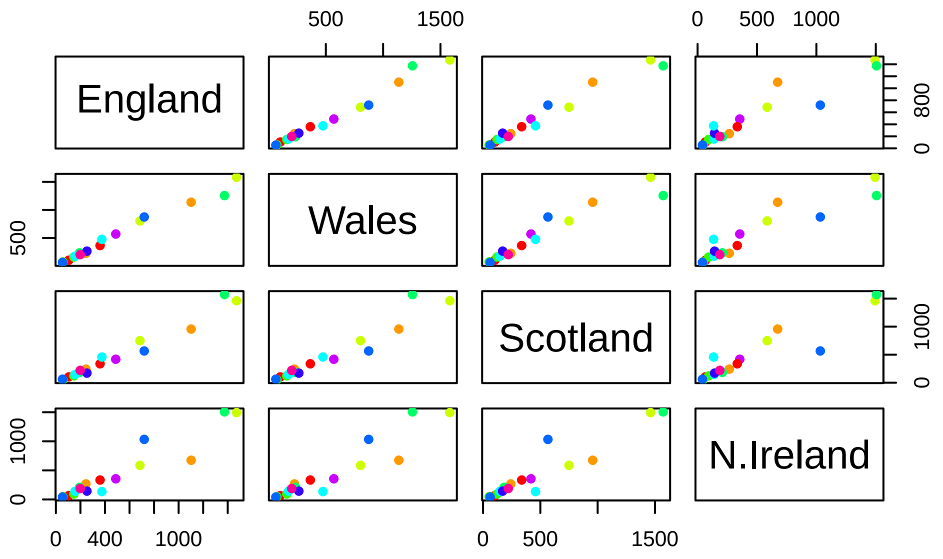
```
#changing beside=TRUE to beside=FALSE will provide the following plot
barplot(as.matrix(x), beside=FALSE, col=rainbow(nrow(x)))
```



##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?

Yes, we are generating pairwise plots by using (pairs()) using X dataset and indicating that the colors of each datapoint will be based off the rainbow colors(multiple colors instead of one color) and the pch=16 just sets the shape of the datapoint on the plot. The resulting pairwise plots provides plots of each column of the dataset (x) against each other (countries) and the named countries in the empty boxes indicate the y-axis and x-axis of each plot. For example all the plots in the England column has England's data as the x-axis and the plots in the England row has England's data as the y-axis. If a given point lies on the diagonal for a given plot it means that the value is the same for the two countries (x-axis and y-axis).

```
pairs(x, col=rainbow(10), pch=16)
```



PCA to the rescue!

The main function in base R to do PCA is called 'prcomp()' One issue with the 'prcomp()' function is that it expects the transpose of our data as input.

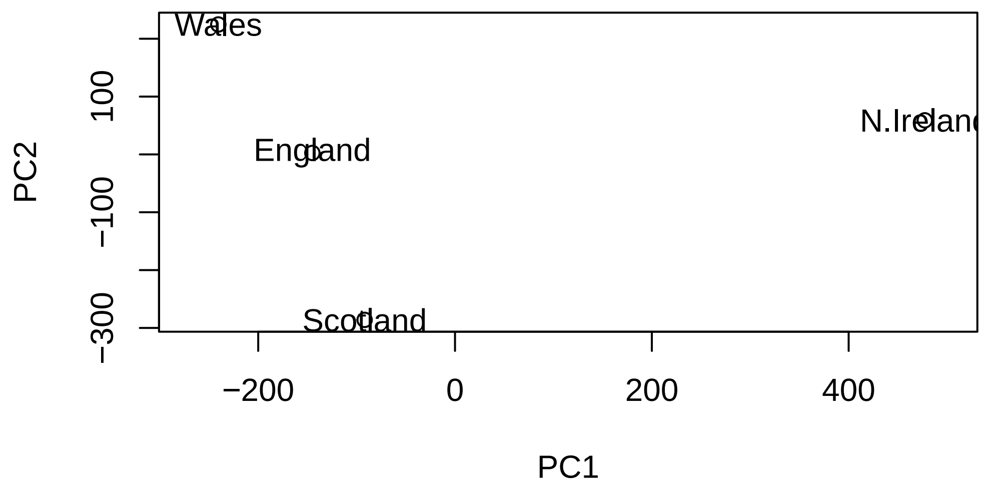
##Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set? A main difference between N. Ireland and other countries is that the data has some outliers more outliers compared to the rest.

```
# Use the prcomp() PCA function to return a list object.
pca <- prcomp( t(x) )
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
```

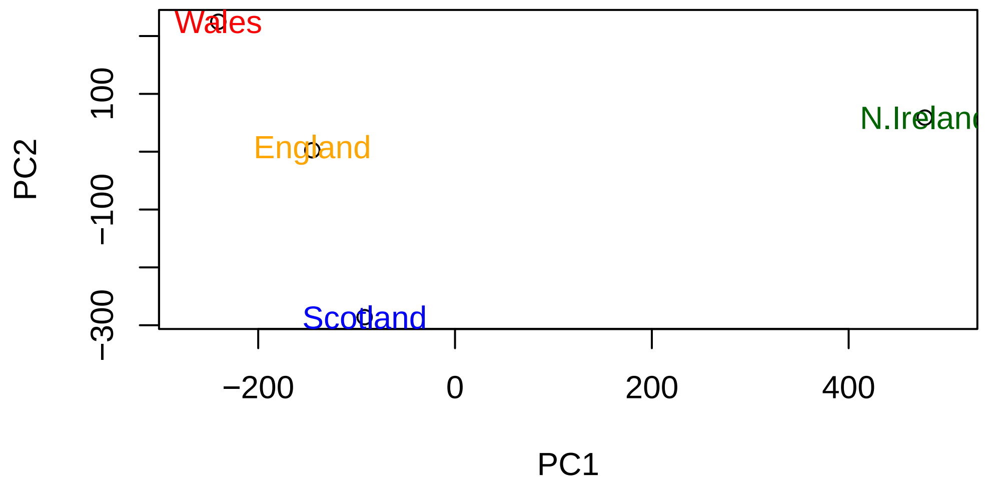
##Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```r
# Plot PC1 vs PC2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x))
```



##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.

```r
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x), col=c("orange", "red", "blue", "darkgreen"))
```



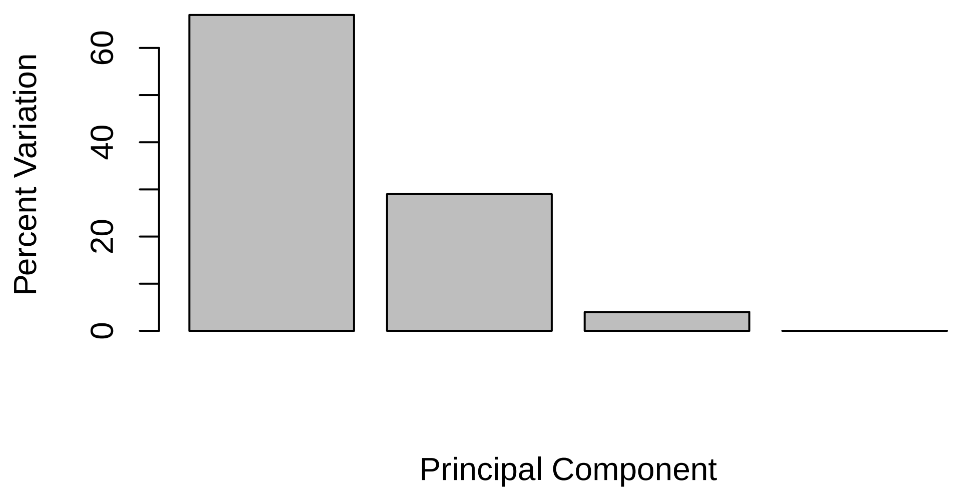
```
#Code below: Use square of pca$sdev , which stands for "standard deviation", to calculate
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

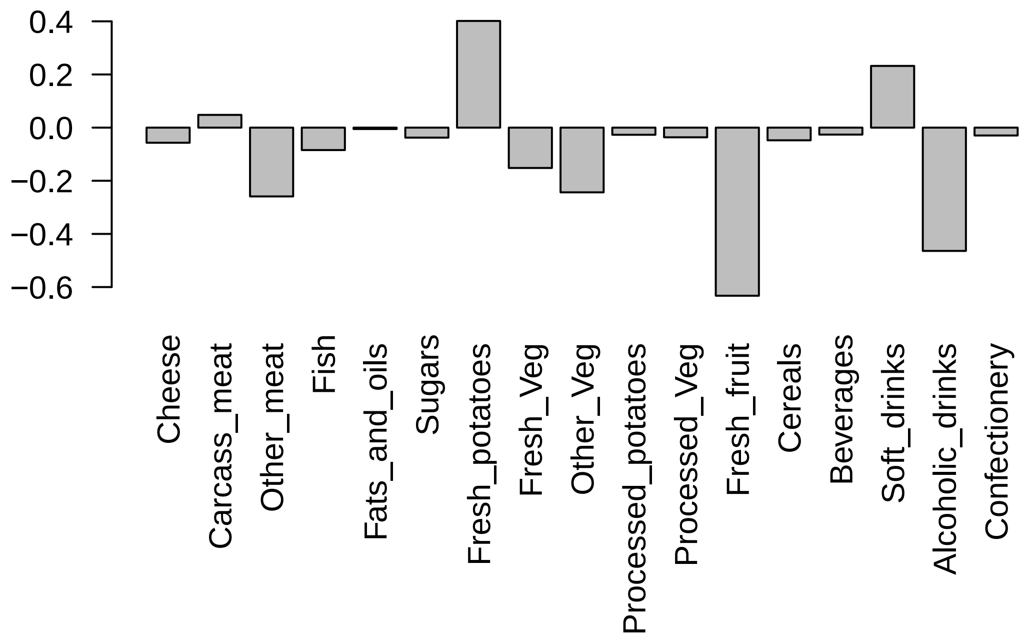
```
#Alternative method to calculate how much variation in the original data
z <- summary(pca)
z$importance
```

```
                             PC1       PC2      PC3          PC4
Standard deviation     324.15019 212.74780 73.87622 4.188568e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

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



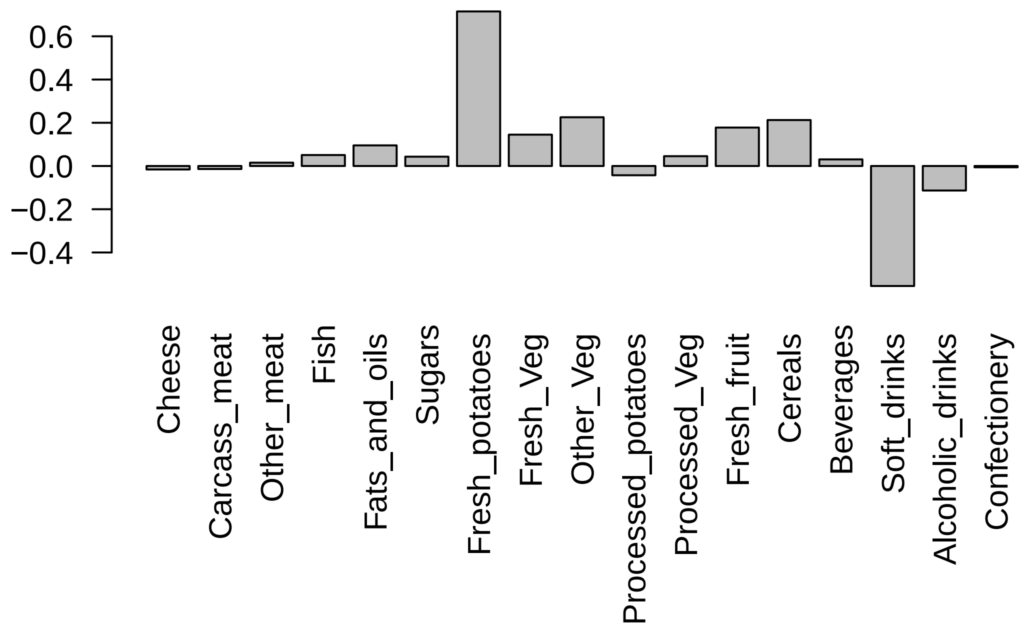
```r
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



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

Fresh potatoes and Soft Drinks. PC2 plot is explaining how each variable is contributing/affecting PC2.

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


0.6
0.4
0.2
0.0
−0.2
−0.4
Cheese
Carcass_meat
Other_meat
Fish
Fats_and_oils
Sugars
Fresh_potatoes
Fresh_Veg
Other_Veg
Processed_potatoes
Processed_Veg
Fresh_fruit
Cereals
Beverages
Soft_drinks
Alcoholic_drinks
Confectionery