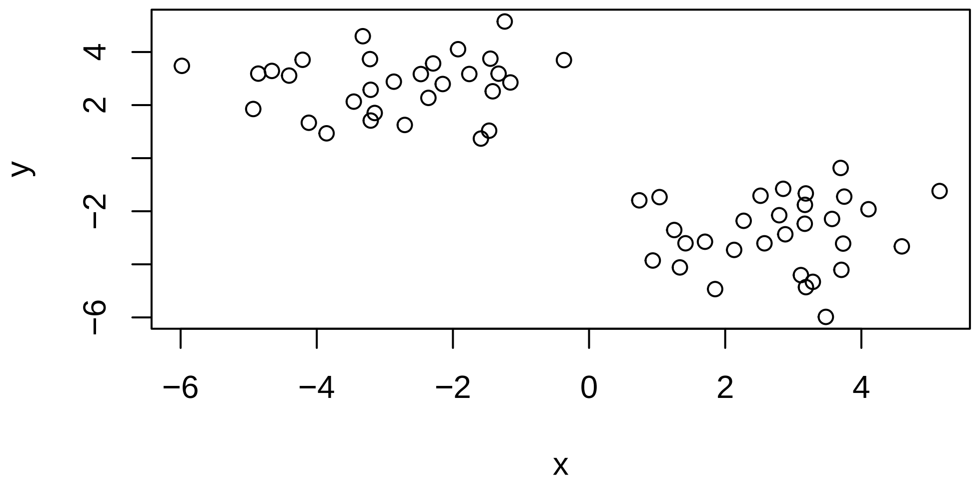
Class 7 - Machine Learning 1

Vivian Chau (A16913056)

#First up kmeans()

Demo od using kmeans() function in base R. First make up some data with a known structure.

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



Now we have some made up data in `x`. Let's see how kmeans works with this data

```r
k <- kmeans(x,centers=2,nstart=20)
k
```

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

Cluster means:
          x         y
1 -2.837711  2.772733
2  2.772733 -2.837711

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] 90.15875 90.15875
 (between_SS / total_SS =  84.0 %)

Available components:

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

Q. How many points are in each cluster?

```r
k$size
```

```
[1] 30 30
```

Q. How do we get to the cluster membership/assignment?

```r
k$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
```

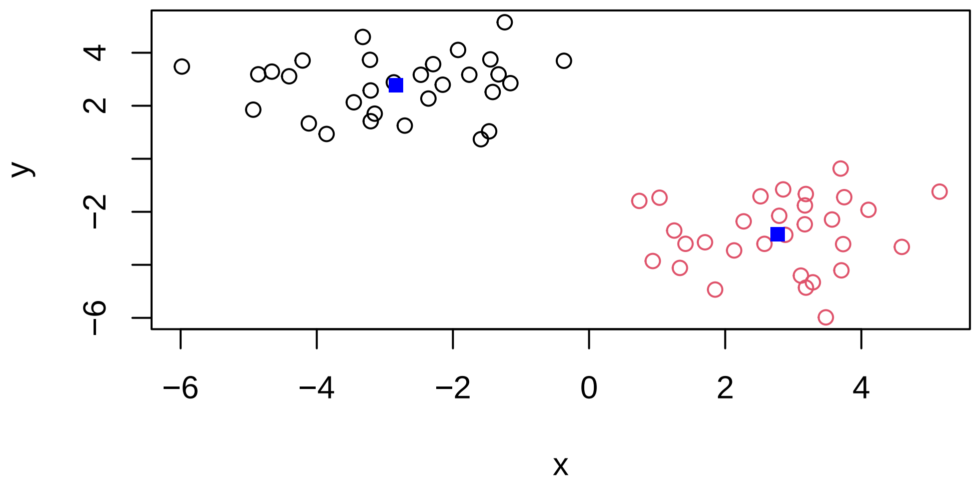
Q. What about cluster centers?

```r
k$centers
```

```
          x         y
1 -2.837711  2.772733
2  2.772733 -2.837711
```

Now we got to the main results, let's use them to plot our data with the kmeans result.

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



Now for Hierarchical Clustering

We will cluster the same data `x` with the `hclust()`. In this case `hclust()` requires a distance matrix as an output.

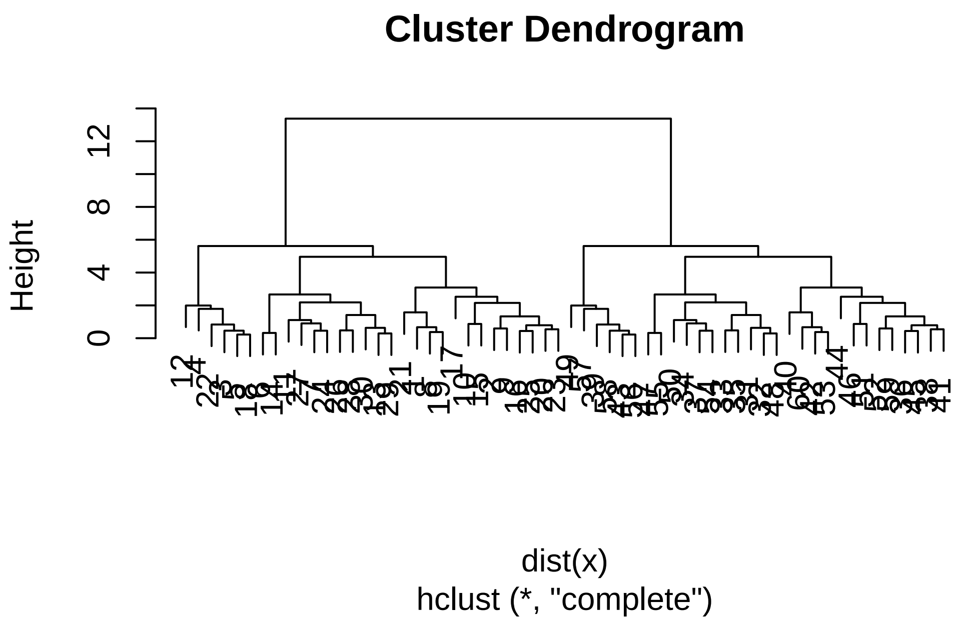
```
hc<-hclust(dist(x))
hc
```

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

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

Let's plot our hclust result

```
plot(hc)
```



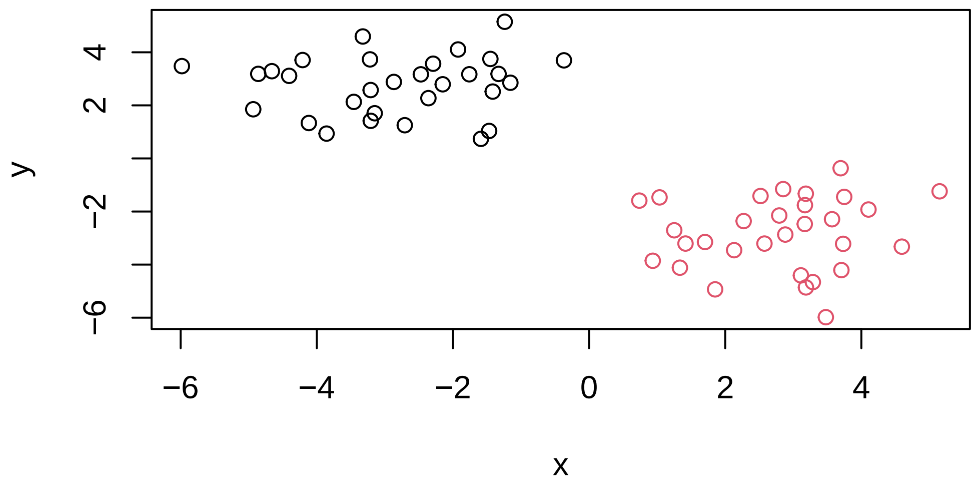
To get our clutster membership vector we need to "cut" the tree with the `cutree()`

```
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 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
```

Now we can plot our data with the hclust() results.

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



Prinicpal Component Analysis (PCA)

PCA of UK food dat

We will read data from the class website and then try a few visualizations.

```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 data frame named x? What R functions could you use to answer this questions? There are 17 rows and 4 columns in this data frame.

```
nrow(x)
```

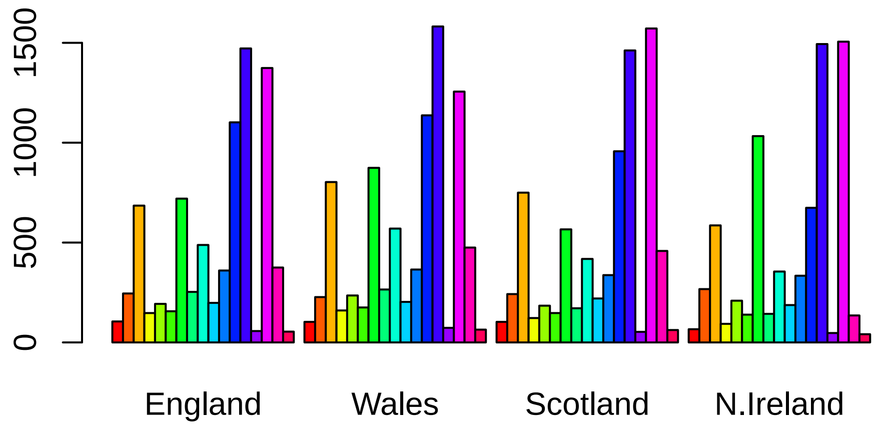
```
[1] 17
```

```
ncol(x)
```

```
[1] 4
```

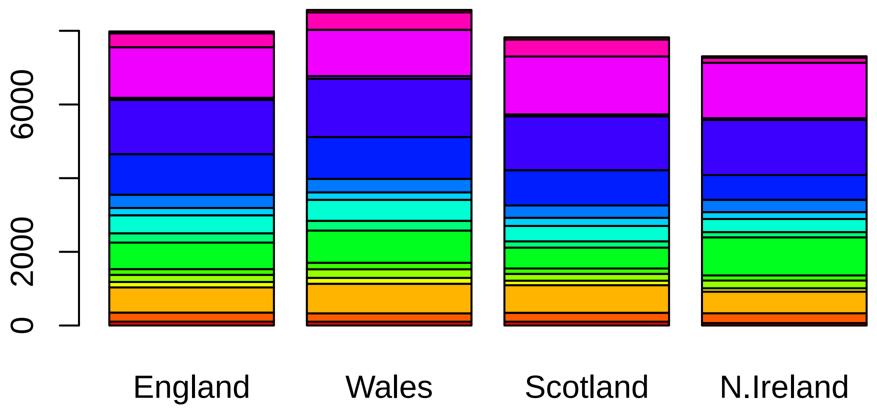
> Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances? Using the row.names=1 is prefered because it is easier to use and properly starts the rows where desired.

```
cols<-rainbow(nrow(x))
barplot(as.matrix(x),col=cols, beside=TRUE)
```



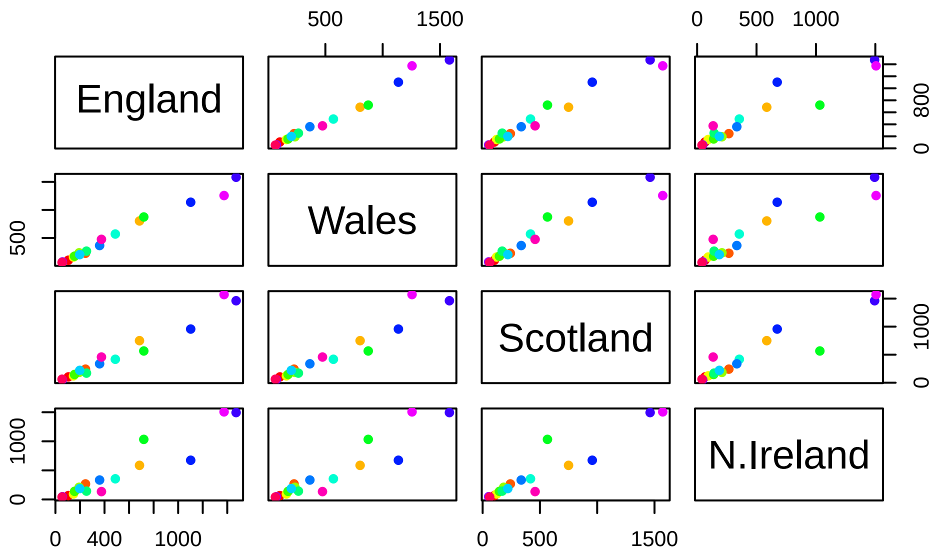
Q3: Changing what optional argument in the above barplot() function results in the following plot? Changing the beside=TRUE to beside=false results in the following plot.

```
cols<-rainbow(nrow(x))
barplot(as.matrix(x),col=cols, beside=FALSE)
```



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? The following plots allow for the comparison of different food categories consumed between two countries. If a given point lies on the diagonal for a given plot, this means that people in a particular country ate the same amount of the specific food category as the people in the other country being compared to.

```r
pairs(x,col=cols,pch=16)
```



Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set? N. Ireland appears to be the most different from the other countries, as many of the dots skew away from the diagonal line. In this case, the food category represented by the blue dot is consumed more in the other countries than N. Ireland.

PCA to the rescue! The main base R PCA function is called `prcomp()`and we will need to give it the transpose of our input data.

```
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
```

```
attributes(pca)
```

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

```
$class
[1] "prcomp"
```

To make our new PCA plot (a.k.a. PCA score plot) we access `pca$x`

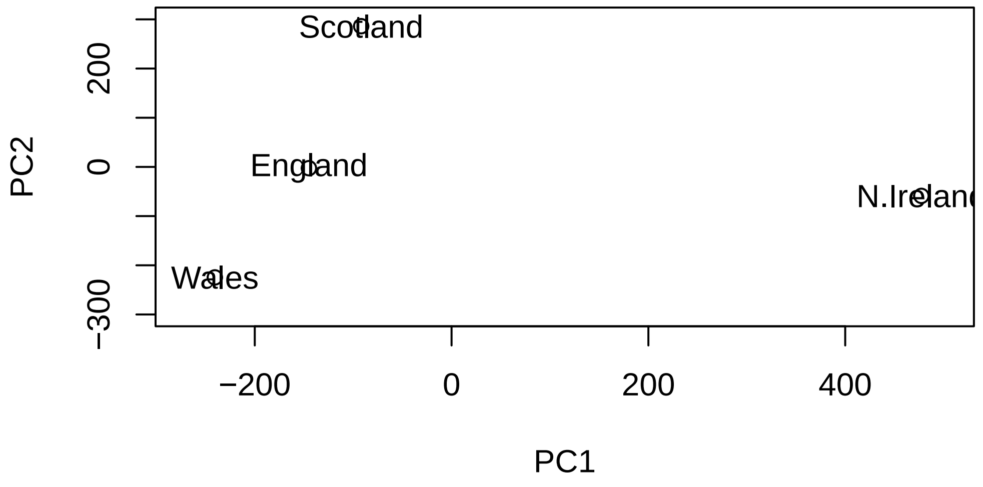
```
head(pca$x)
```

```
                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.

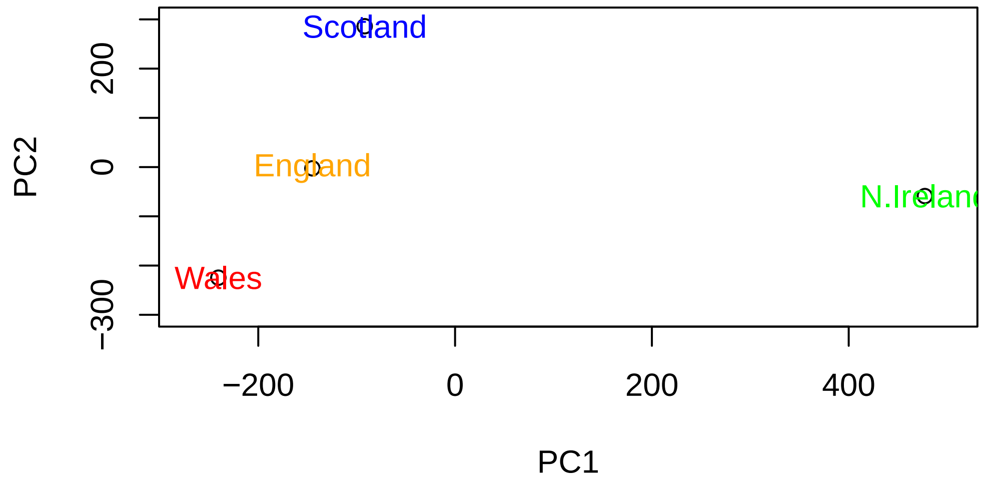
Let's plot our main results:

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500), ylim=c(-300,300))
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.

```
country_cols<-c("orange","red","blue","green")
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500), ylim=c(-300,300))
text(pca$x[,1], pca$x[,2], colnames(x),col=country_cols)
```



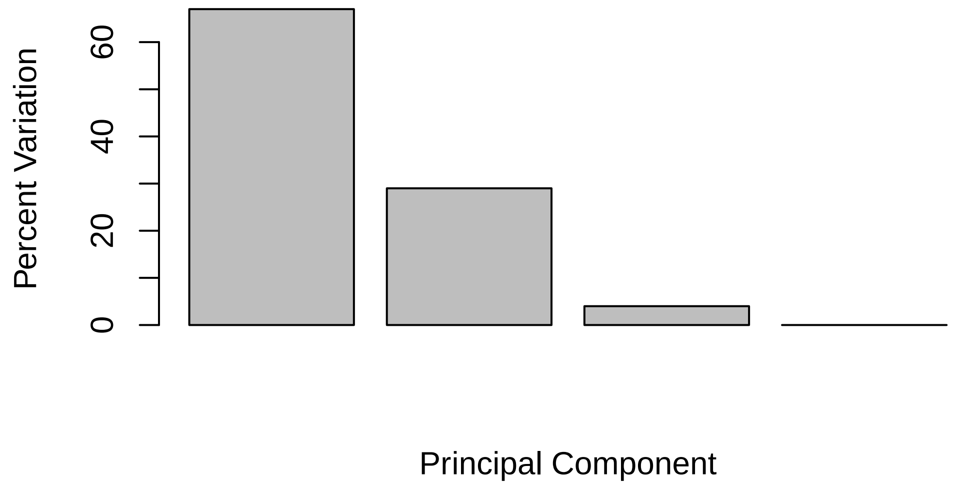
```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

```
z <- summary(pca)
z$importance
```

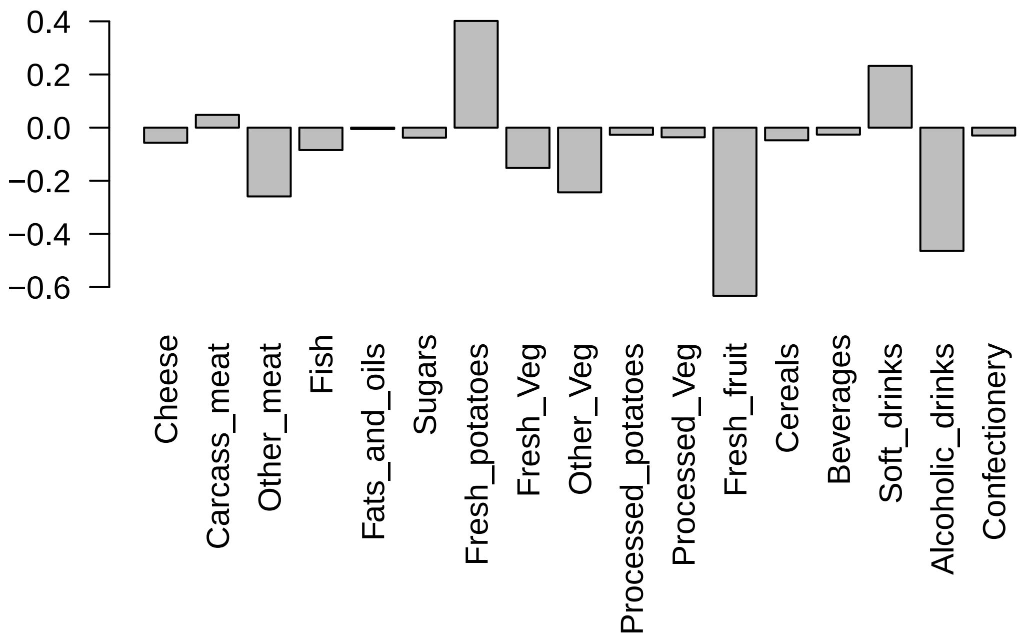
```
                             PC1       PC2      PC3          PC4
Standard deviation     324.15019 212.74780 73.87622 3.175833e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

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



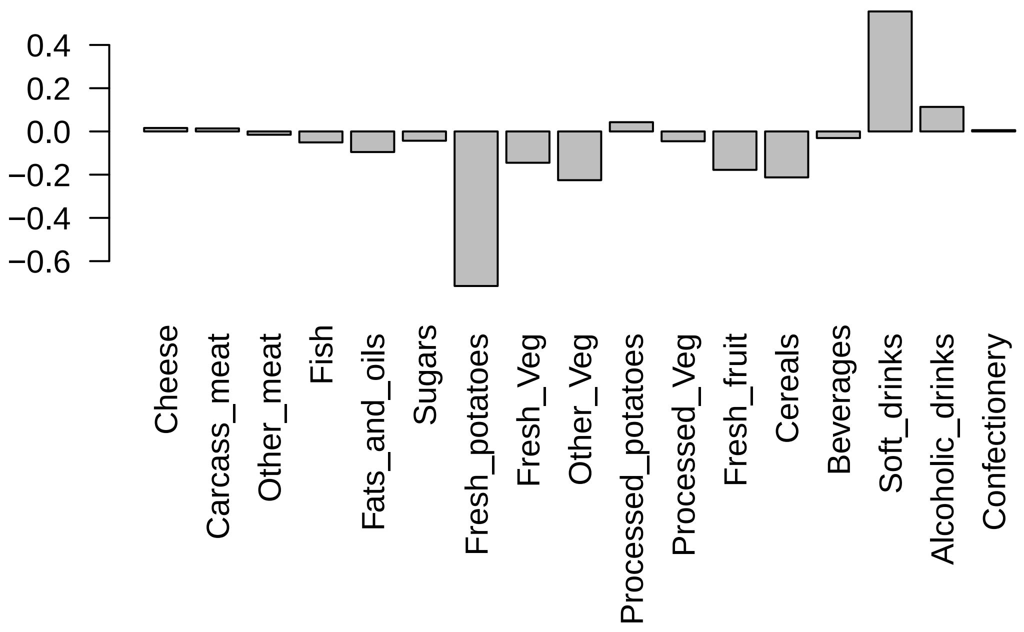
Digging deeper (variable loadings)

```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 )
```



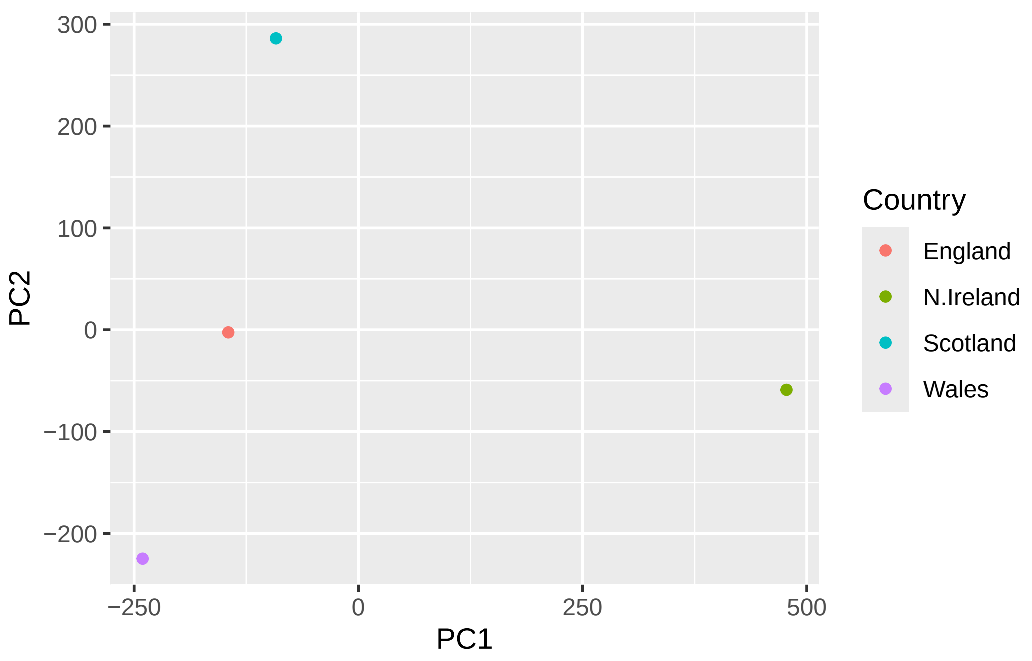
Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about? Fresh potatatoes has a large negative score while soft drinks has a high positive loading score. PC2 mainly tells us about the second most important differentce within the data, that differs from the primary trend.

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



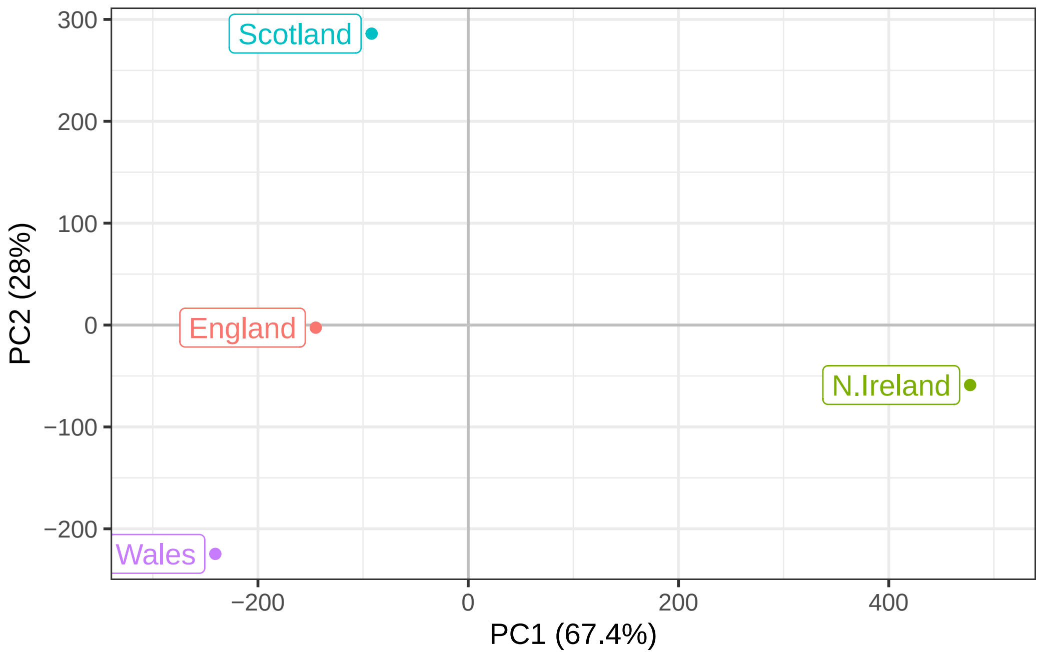
Using ggplot for these figures:

```
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()
```



To make this graph 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()
```



PCA of RNA-seq data:

Reading data from class website

```r
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? There are 10 samples and 100 genes in this data set.

```r
nrow(rna.data)
```

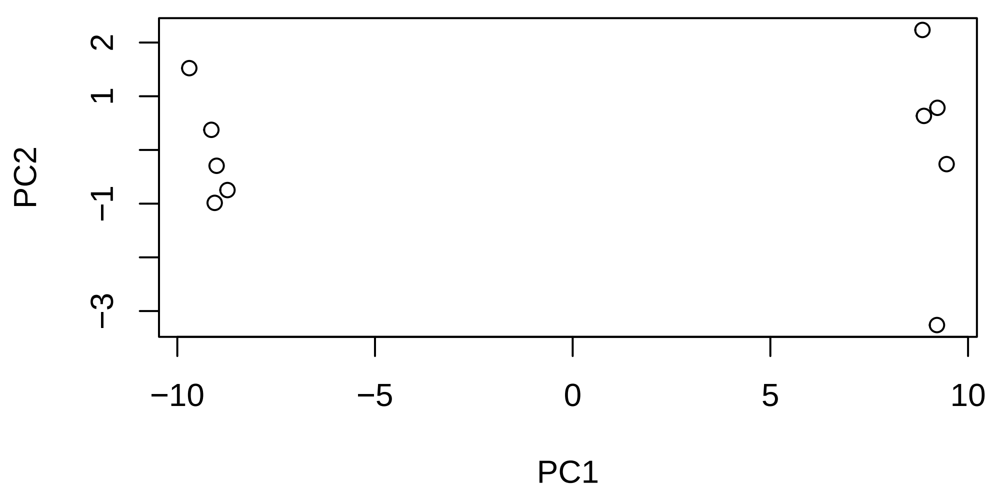
```
[1] 100
```

```r
ncol(rna.data)
```

```
[1] 10
```

```r
## Again we have to take the transpose of our data
pca<-prcomp(t(rna.data),scale=TRUE)
```

```r
## Here's a simple plot of pc1 and pc2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```

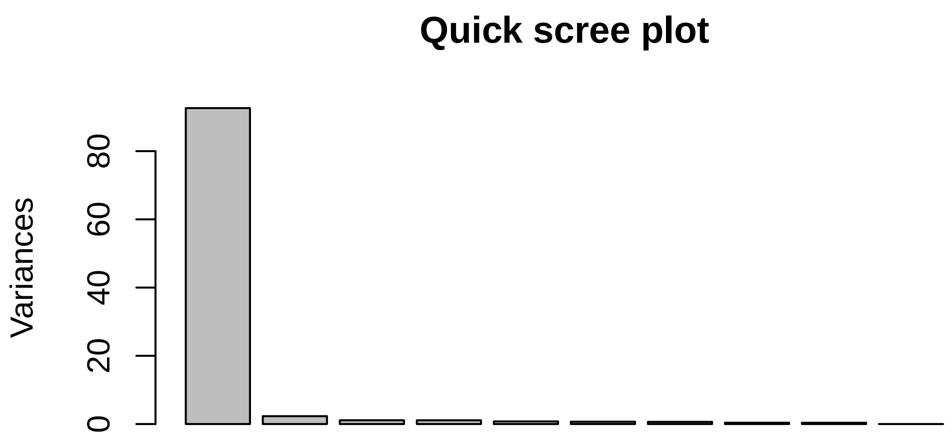


```r
summary(pca)
```

```
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 3.457e-15
Proportion of Variance 0.00385 0.00364 0.000e+00
Cumulative Proportion  0.99636 1.00000 1.000e+00
```

A quick barplot summary of this Proportion of Variance

```r
plot(pca, main="Quick scree plot")
```



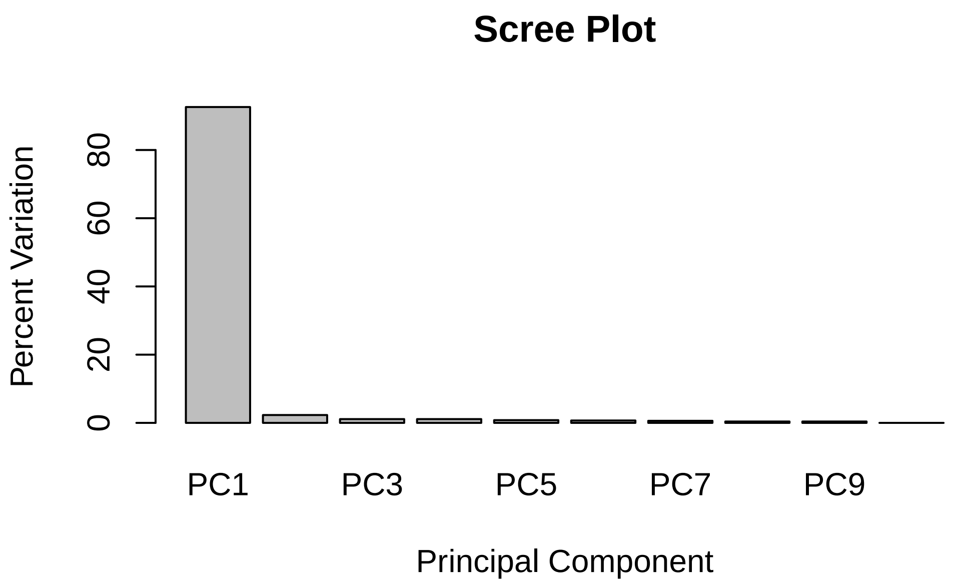
We can calculate how much variation in the original data each PC accounts for:

```r
## Variance captured per PC
pca.var <- pca$sdev^2

## Percent variance is often more informative to look at
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)
pca.var.per
```

```
 [1] 92.6  2.3  1.1  1.1  0.8  0.7  0.6  0.4  0.4  0.0
```

And we can generate our own scree-plot like:

```r
barplot(pca.var.per, main="Scree Plot",
        names.arg = paste0("PC", 1:10),
        xlab="Principal Component", ylab="Percent Variation")
```


Scree Plot
Percent Variation
0
20
40
60
80
PC1
PC3
PC5
PC7
PC9
Principal Component