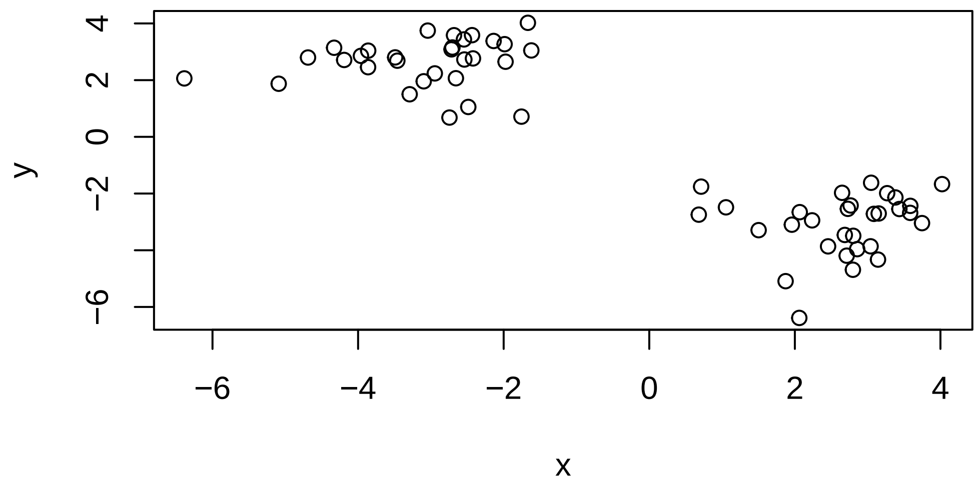
Class 07 Lab

Darby Patterson

Example of K-means clustering

First, we make up some data with known structure, so we know the suspected answer.

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



Now we have structured data in x. Lets see if k-means is able to identify the groups.

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

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

Cluster means:
          x         y
1 -3.092537  2.636453
2  2.636453 -3.092537

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] 55.36825 55.36825
 (between_SS / total_SS =  89.9 %)

Available components:

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

Now we will explore k.

```r
k$size
```

```
[1] 30 30
```

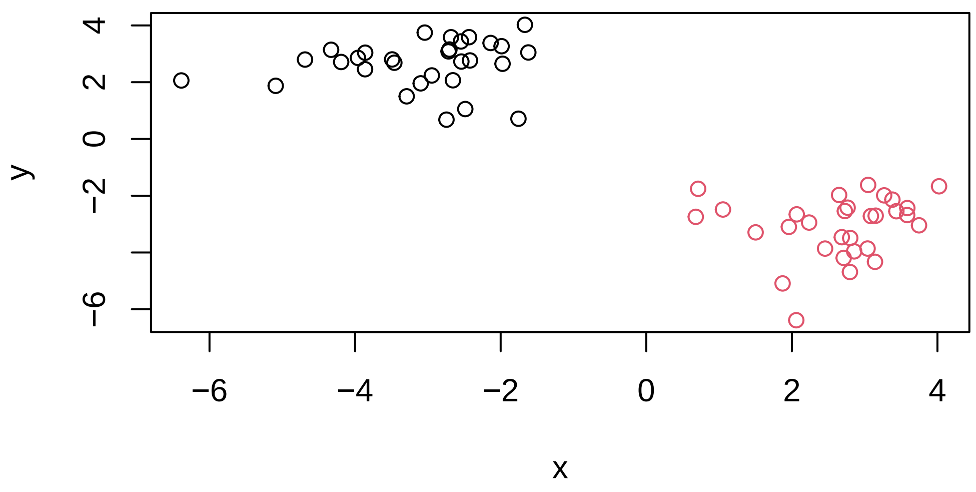
```r
k$centers
```

```
          x         y
1 -3.092537  2.636453
2  2.636453 -3.092537
```

```r
k$cluster
```

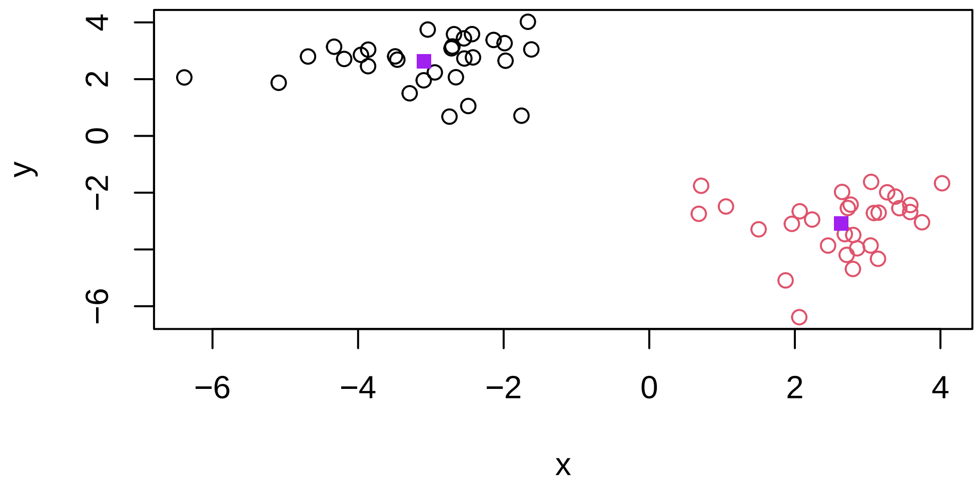
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
plot(x, col= k$cluster)
```



Now we add the cluster centers:

```
plot(x, col= k$cluster)
points (k$centers, col= 'purple', pch=15)
```



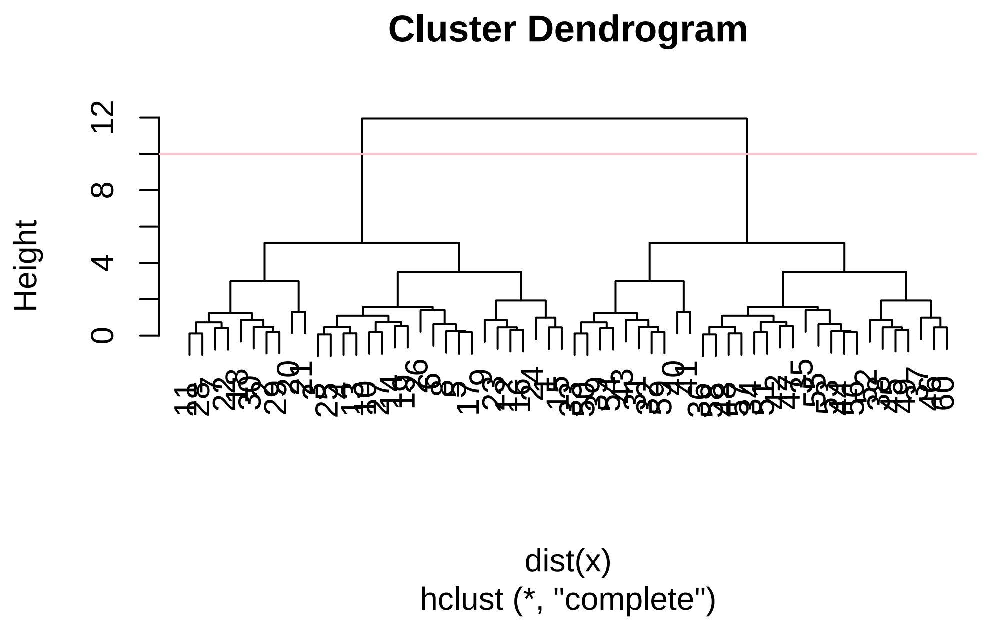
Hierarchical Clustering

Lets use the same data stored in `x` and using the `hclust()` function.

```r
clustering <- hclust(dist(x))
clustering
```

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

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

```r
plot(clustering)
abline(h= 10, col= 'pink')
```



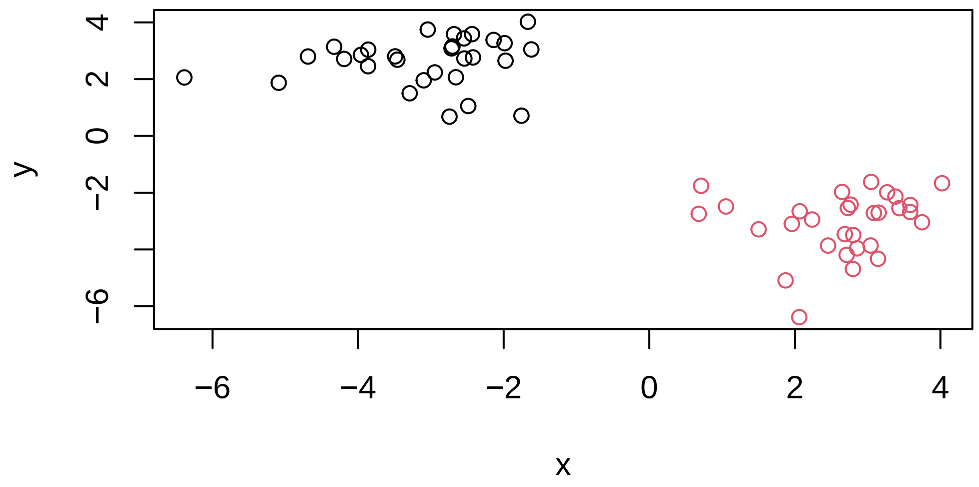
To get our results (membership vector), we need to "cut" the tree. using `cutree()`.

```
subgroups <- cutree(clustering, h=10)
subgroups
```

```
 [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
```

Plotting this…

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



You can also "cut" your tree with number of desired clusters:

```
cutree(clustering, k = 2)
```

```
 [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
```

Principle Component Analysis (PCA)

PCA of U.K food

First we will read the provided `UK_foods.csv` input file (note we can read this directly from the following tinyurl short link: "https://tinyurl.com/UK-foods".

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

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

```
#dim(x)
dim(x)
```

```
[1] 17  4
```

```
#we could also use nrow(x); ncol(x)
```

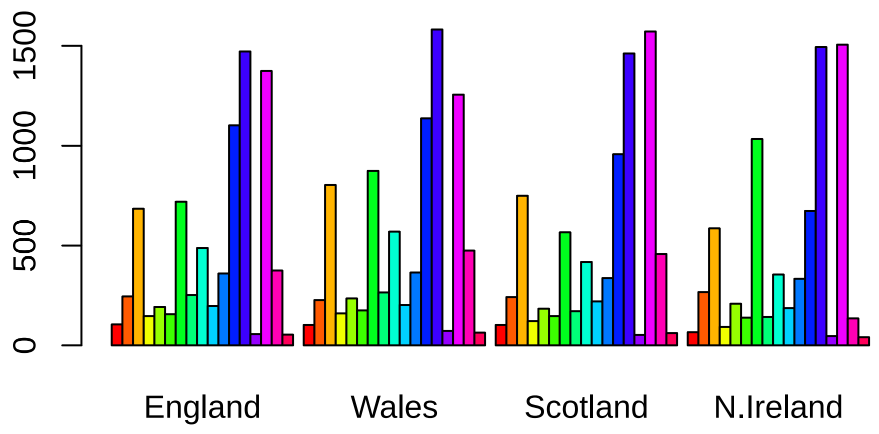
We can use `dim(x)` to see how many rows and columns are in our data set.

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?

making the function row.names equal to 1 is more efficient and allows us and RStudio to read the data properly.

Now we can generate some basic visuals

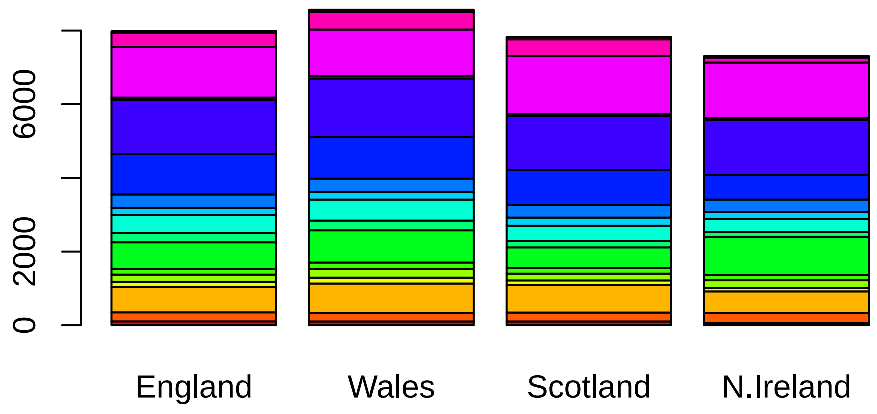
```
barplot(as.matrix(x), col= rainbow(nrow(x)), beside=T)
```



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

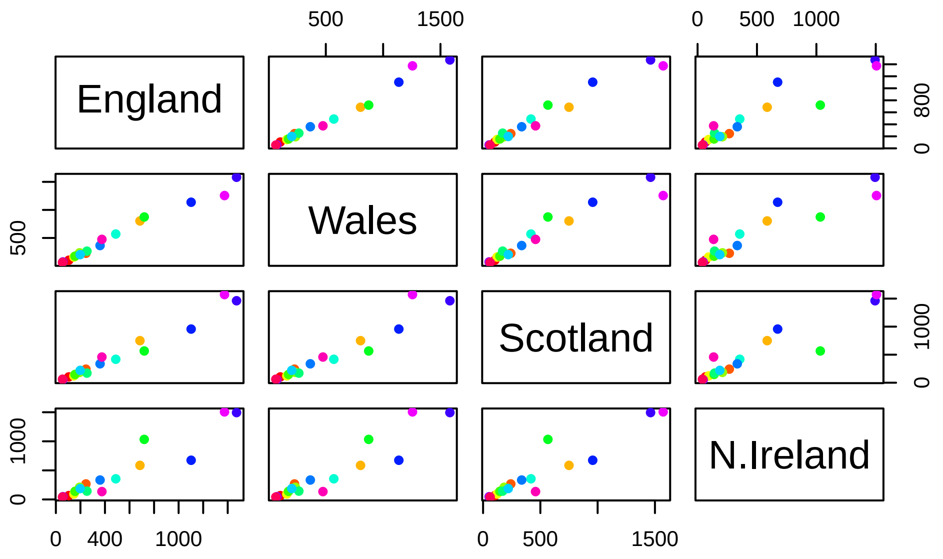
Setting `beside=FALSE` would make it so the data is not side by side but instead vertically plotted like below.

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



Let's refine our barplot...

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



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 diagonal line represents the distribution of the variable, that the variable plotted on x is the same as the variable plotted on y.

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

N. Ireland seems to consume the most food overall compared to the three other listed nations.

Applying PCA

Let's apply PCA using the command `prcomp()`. This function expects the transpose of our data.

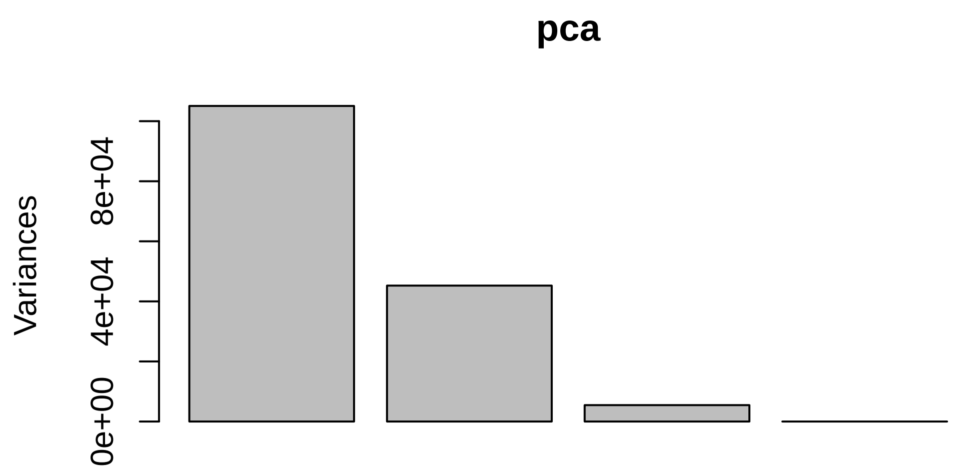
```
pca <- prcomp (t(x))
summary(pca)
```

```
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
```

Let's plot the PCA results

```r
plot(pca)
```



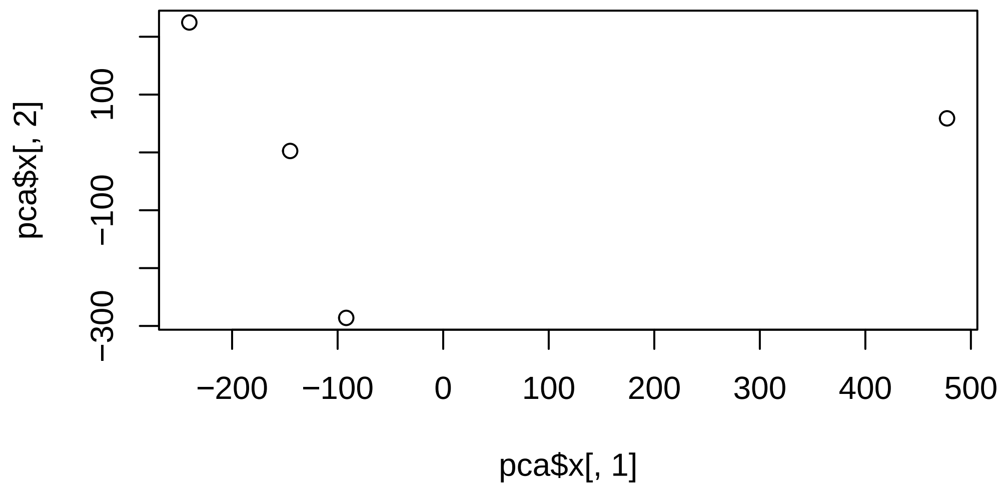
We need to access the results of the PCA exploring the dataframe.

```r
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
```

Plotting:

```
plot( x=pca$x[,1], y=pca$x[,2] )
```

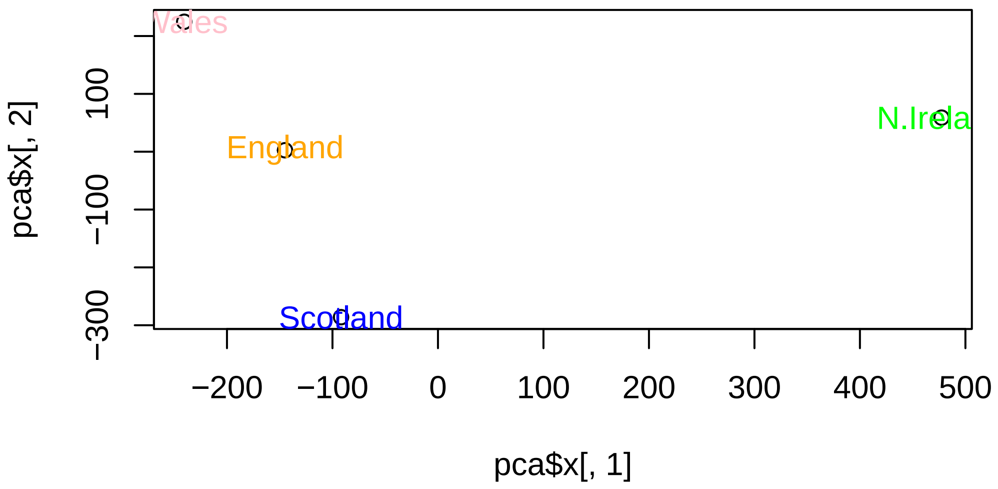


Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

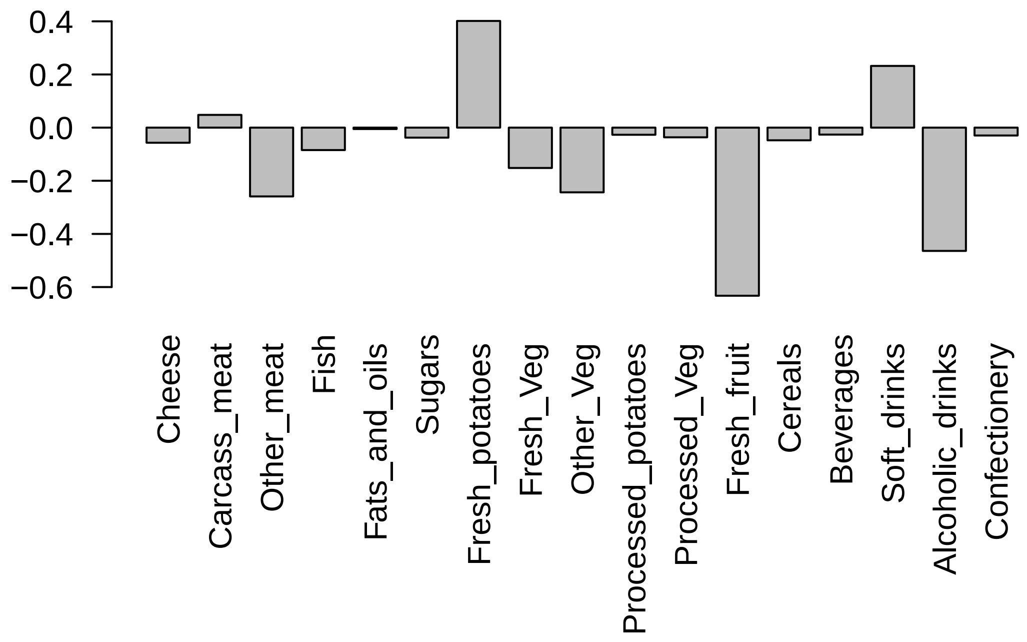
Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

Plotted below:

```
plot( x=pca$x[,1], y=pca$x[,2])
countrycolor <- c("orange", "pink","blue","green" )
text( x=pca$x[,1], y=pca$x[,2], colnames(x), col= countrycolor )
```

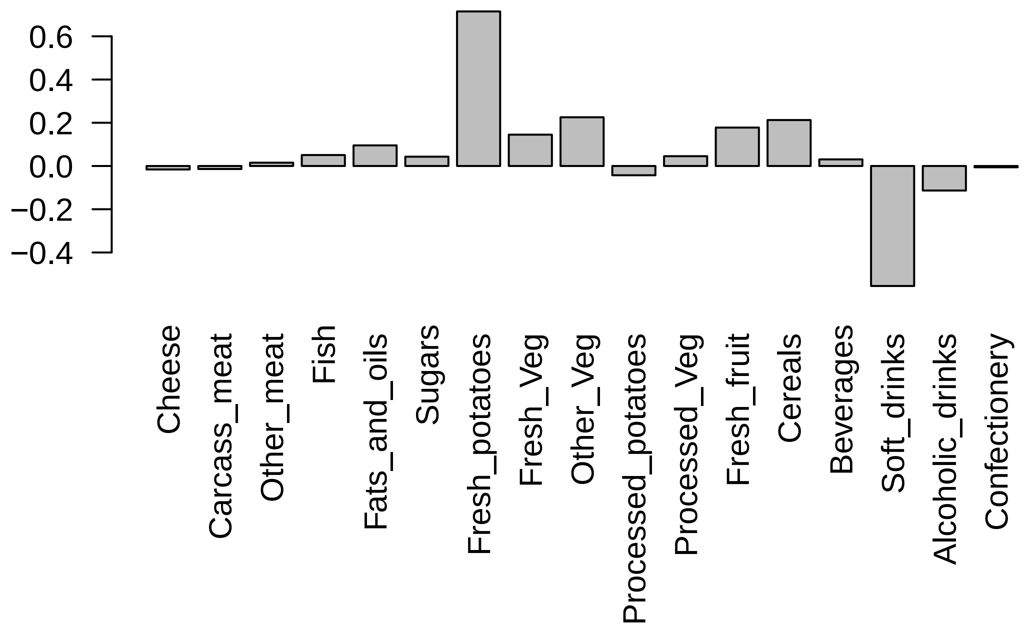


```r
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



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

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



The most prominent groups in PCA 2 were fresh potatoes and soft drink, which PCA 2 tells us about the second largest amount of variability second to PCA 1.

PCA of RNA- Sequence dataset

We first load in the file:

```
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
```

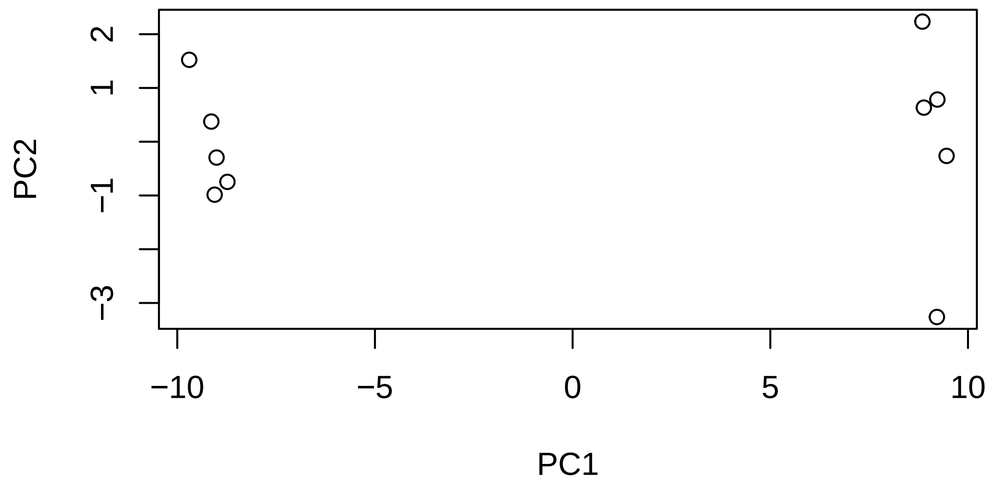
Q10: How many genes and samples are in this data set?

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

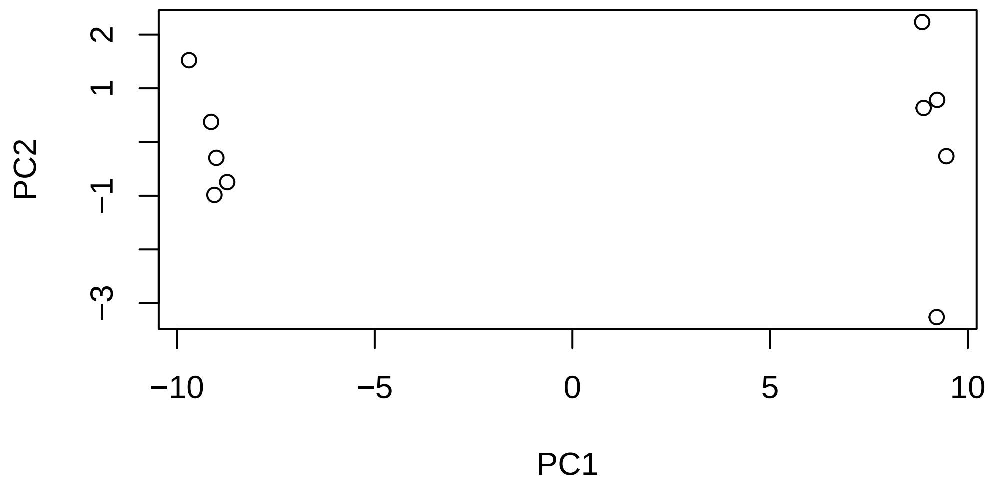
```
[1] 100  10
```

There are 100 genes, and 10 samples.

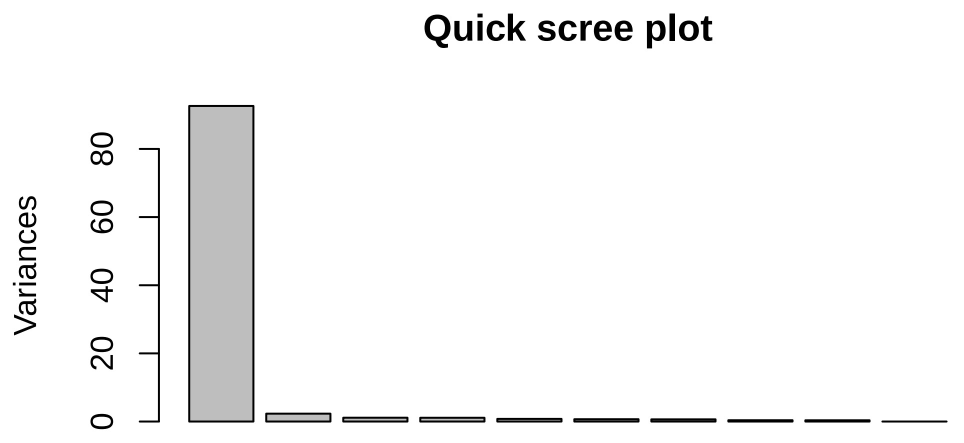
```r
## Again we have to take the transpose of our data
pca_rna <- prcomp(t(rna.data), scale=TRUE)

## Simple un polished plot of pc1 and pc2
plot(pca_rna$x[,1], pca_rna$x[,2], xlab="PC1", ylab="PC2")
```



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

## Simple un polished plot of pc1 and pc2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```



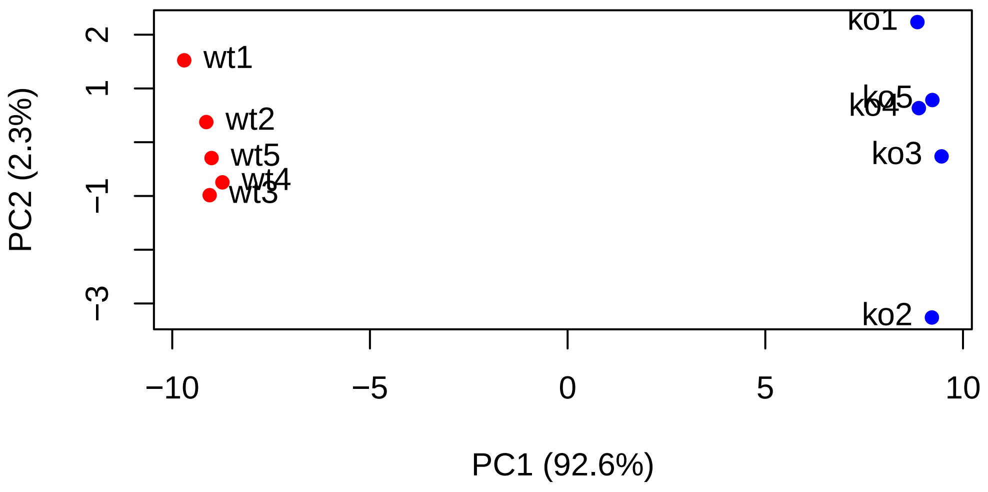
```
plot(pca, main="Quick scree plot")
```



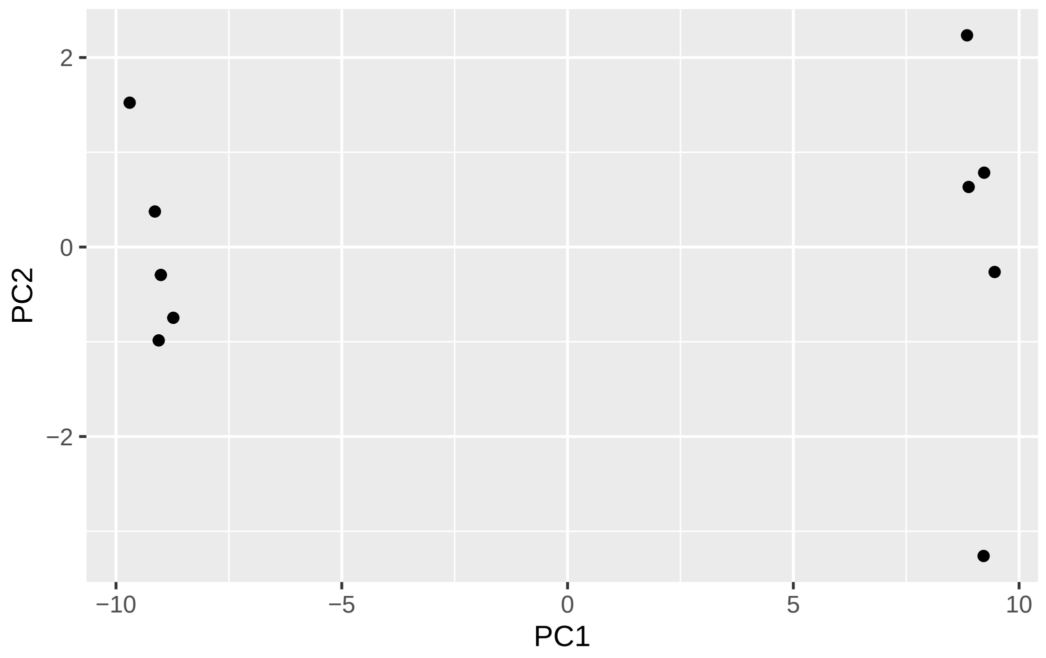
```
## Variance captured per PC
pca.var <- pca$sdev^2

## 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
```

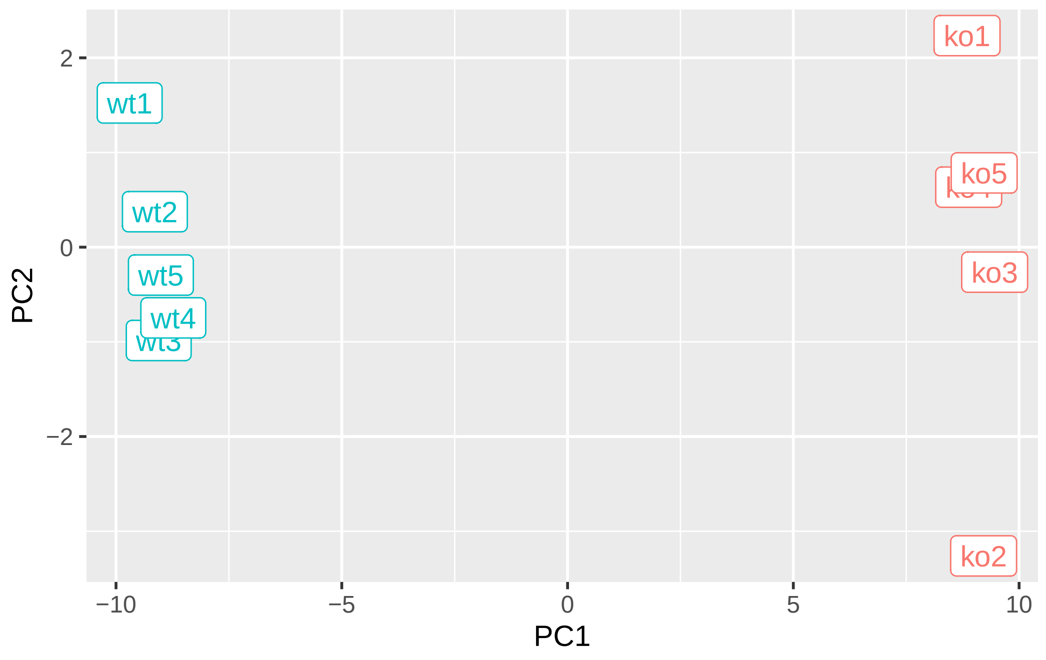
```
## A vector of colors for wt and ko samples
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

plot(pca$x[,1], pca$x[,2], col=colvec, pch=16,
     xlab=paste0("PC1 (", pca.var.per[1], "%)"),
     ylab=paste0("PC2 (", pca.var.per[2], "%)"))

text(pca$x[,1], pca$x[,2], labels = colnames(rna.data), pos=c(rep(4,5), rep(2,5)))
```



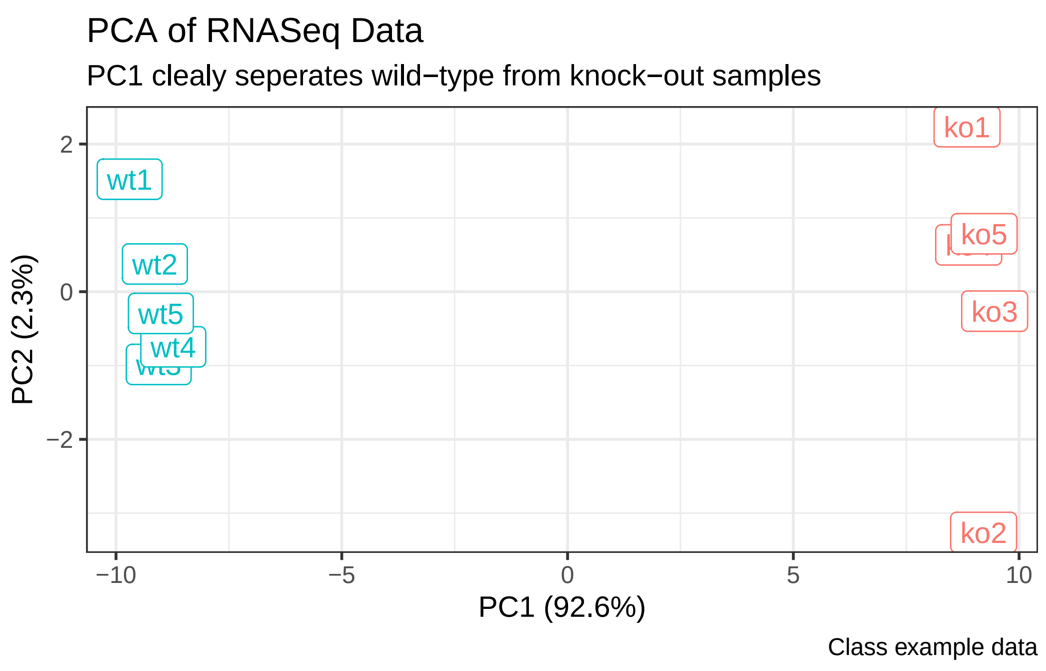
```r
library(ggplot2)

df <- as.data.frame(pca$x)

# Our first basic plot
ggplot(df) +
  aes(PC1, PC2) +
  geom_point()
```



```r
# Add a 'wt' and 'ko' "condition" column
df$samples <- colnames(rna.data)
df$condition <- substr(colnames(rna.data),1,2)

p <- ggplot(df) +
        aes(PC1, PC2, label=samples, col=condition) +
        geom_label(show.legend = FALSE)
p
```



```
p + labs(title="PCA of RNASeq Data",
       subtitle = "PC1 clealy seperates wild-type from knock-out samples",
       x=paste0("PC1 (", pca.var.per[1], "%)"),
       y=paste0("PC2 (", pca.var.per[2], "%)"),
       caption="Class example data") +
     theme_bw()
```


PCA of RNASeq Data
PC1 clealy seperates wild−type from knock−out samples
ko1
wt1
ko5
wt2
ko3
wt5
wt4
ko2
2
0
−2
−10
−5
0
5
10
PC1 (92.6%)
PC2 (2.3%)
Class example data