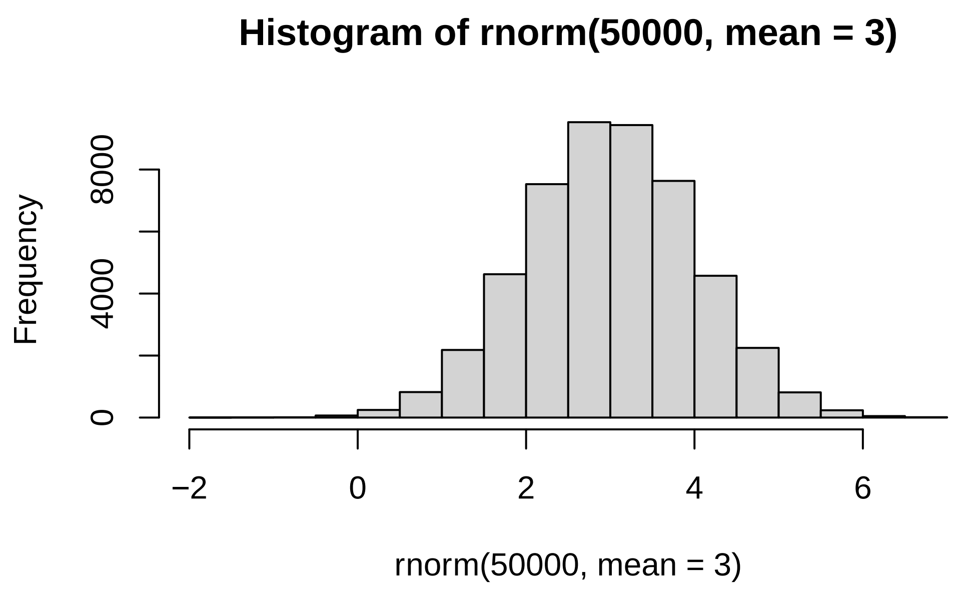
inclass07

Clustering

First, lets make up some data to cluster so we can get a feel for these methodsandd how to work with them.

We can use the "rnorm" function to get random numbers from a normal distribution around a given "mean"

```r
hist(rnorm(50000, mean=3))
```



Lets gets 30 points with a mean of 3

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

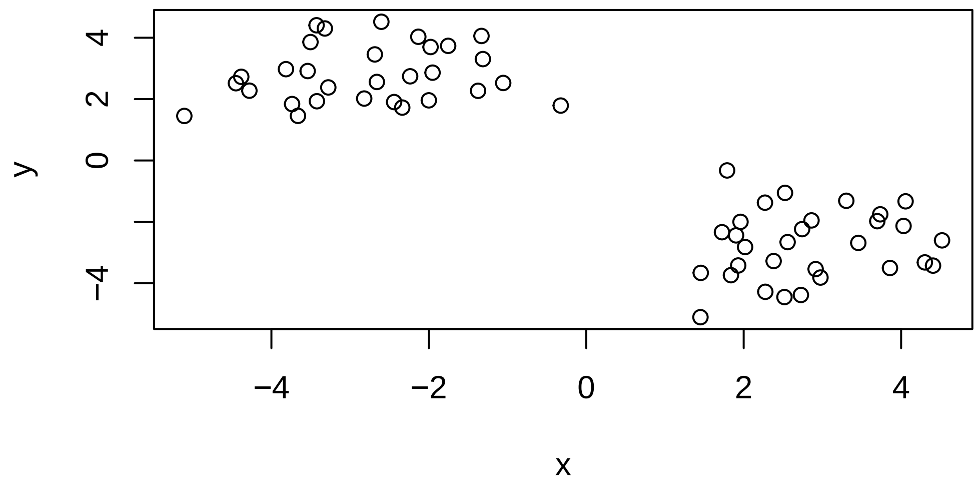
put two of these together

```r
x <- cbind(x=temp, y=rev(temp))
x
```

```
              x         y
 [1,]  2.742507 -2.237313
 [2,]  1.930372 -3.421799
 [3,]  3.697294 -1.977868
 [4,]  4.056811 -1.331041
 [5,]  2.558193 -2.660298
 [6,]  1.454726 -3.662458
 [7,]  4.301499 -3.320913
 [8,]  4.405483 -3.426306
 [9,]  2.270497 -1.374508
[10,]  2.380468 -3.277130
[11,]  4.520089 -2.603031
[12,]  1.789649 -0.324164
[13,]  3.303600 -1.313204
[14,]  2.726551 -4.382811
[15,]  4.030920 -2.133516
[16,]  1.724160 -2.337513
[17,]  1.450845 -5.106174
[18,]  2.517830 -4.450796
[19,]  2.274701 -4.279307
[20,]  2.017993 -2.820837
[21,]  3.858287 -3.503523
[22,]  2.914180 -3.539410
[23,]  2.975645 -3.815064
[24,]  3.456224 -2.686058
[25,]  2.861493 -1.952562
[26,]  1.959717 -2.000008
[27,]  3.734885 -1.754357
[28,]  2.525061 -1.055127
[29,]  1.903224 -2.441048
[30,]  1.838105 -3.737298
[31,] -3.737298  1.838105
[32,] -2.441048  1.903224
[33,] -1.055127  2.525061
```

```
[34,] -1.754357  3.734885
[35,] -2.000008  1.959717
[36,] -1.952562  2.861493
[37,] -2.686058  3.456224
[38,] -3.815064  2.975645
[39,] -3.539410  2.914180
[40,] -3.503523  3.858287
[41,] -2.820837  2.017993
[42,] -4.279307  2.274701
[43,] -4.450796  2.517830
[44,] -5.106174  1.450845
[45,] -2.337513  1.724160
[46,] -2.133516  4.030920
[47,] -4.382811  2.726551
[48,] -1.313204  3.303600
[49,] -0.324164  1.789649
[50,] -2.603031  4.520089
[51,] -3.277130  2.380468
[52,] -1.374508  2.270497
[53,] -3.426306  4.405483
[54,] -3.320913  4.301499
[55,] -3.662458  1.454726
[56,] -2.660298  2.558193
[57,] -1.331041  4.056811
[58,] -1.977868  3.697294
[59,] -3.421799  1.930372
[60,] -2.237313  2.742507
```

```
plot(x)
```



##K-means clustering

Very popular clustering mehtods, especially for big datasets,that we can use with the "kmeans()" function in base R.

```
km <- kmeans(x, centers=2)
km
```

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

Cluster means:
          x         y
1 -2.764181  2.806034
2  2.806034 -2.764181

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 1 1

Within cluster sum of squares by cluster:
[1] 62.53017 62.53017
 (between_SS / total_SS =  88.2 %)
```

```
Available components:

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

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

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

Cluster means:
          x         y
1 -2.764181  2.806034
2  2.806034 -2.764181

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] 62.53017 62.53017
 (between_SS / total_SS =  88.2 %)

Available components:

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

Q: What component of your result objects deatils

-cluster size?

```r
km$size
```

```
[1] 30 30
```

-cluster assignemnt/membership?

```r
km$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 1 1
```

cluster center?

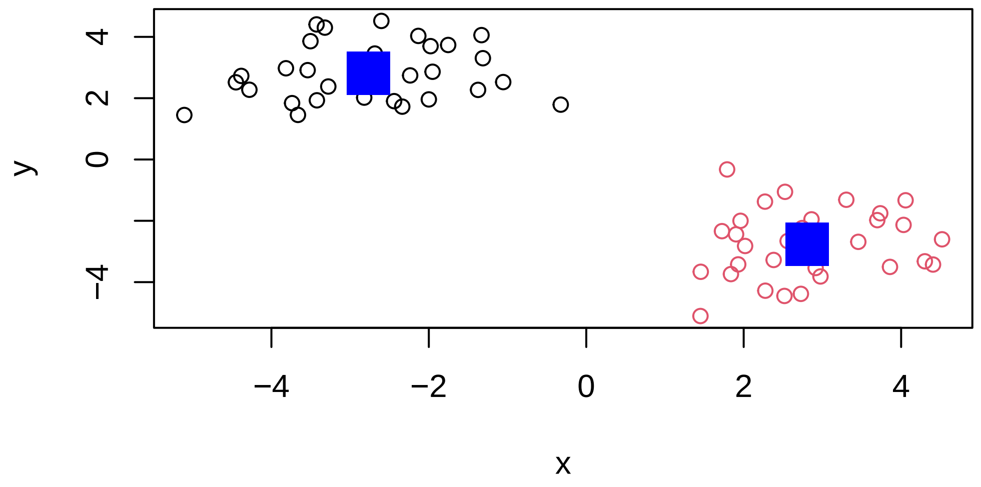
```
km$centers
```

```
          x         y
1 -2.764181  2.806034
2  2.806034 -2.764181
```

> Q. Plot x colored by the kmeans cluster assignement and add cluster points as blue points

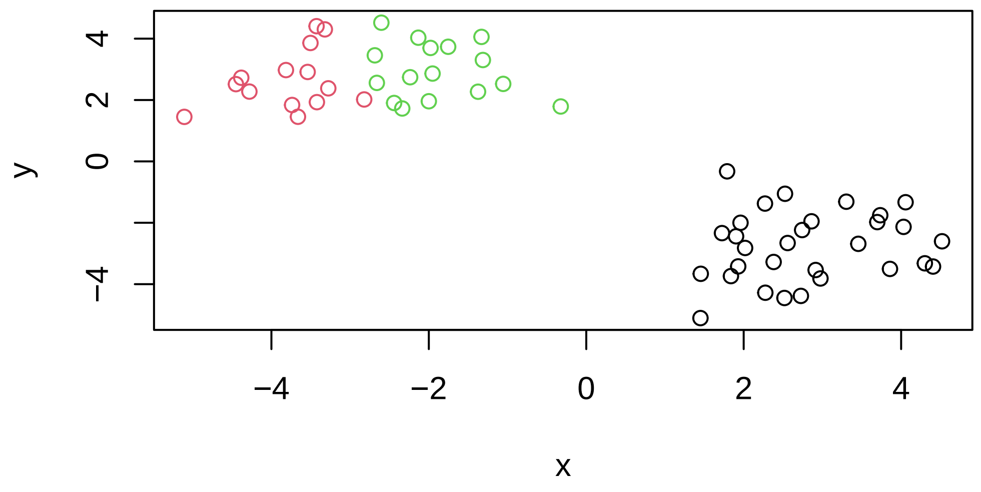
#pch made it a square, cex made the square bigger

```
plot(x, col=km$cluster)
points(km$centers, col="blue", pch=15, cex=3)
```



Q. Lets cluster into 3 groups or same "x" data and make a plot

```r
km <- kmeans(x,3)
plot(x, col=km$cluster)
```



#Hierarchical clustering

we can use the ‘hclust()’ function for Heriarchical Clustering. Unlike ‘kmeans()’ where we could just pass in our data as input, we need to give ‘hclust()’ a distance matrix.

We will use the ‘dist()’ fucntion to start with

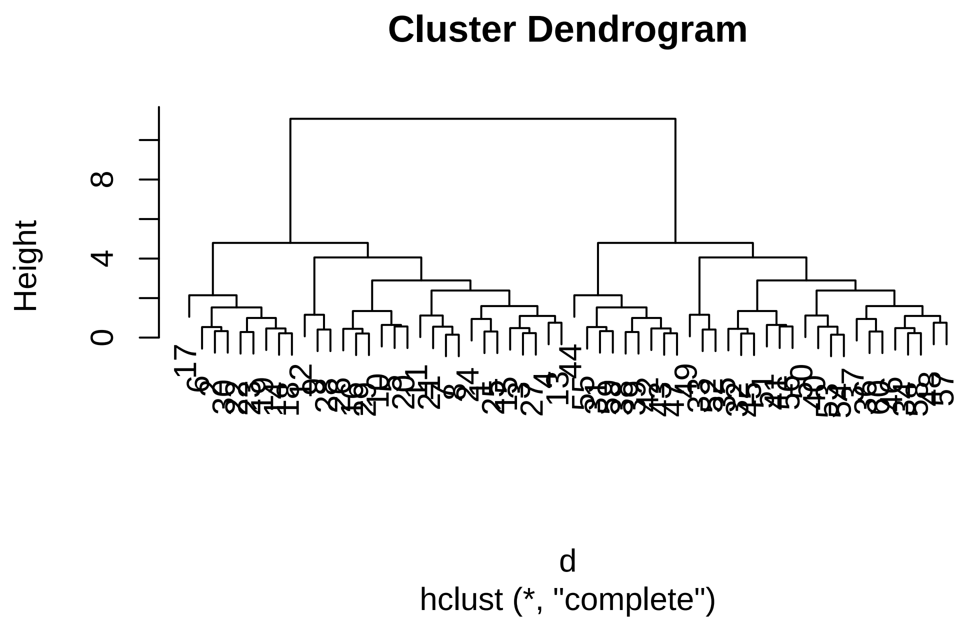
```r
d <- dist(x)
hc <- hclust(d)
hc
```

```

Call:
hclust(d = d)

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

```
plot(hc)
```

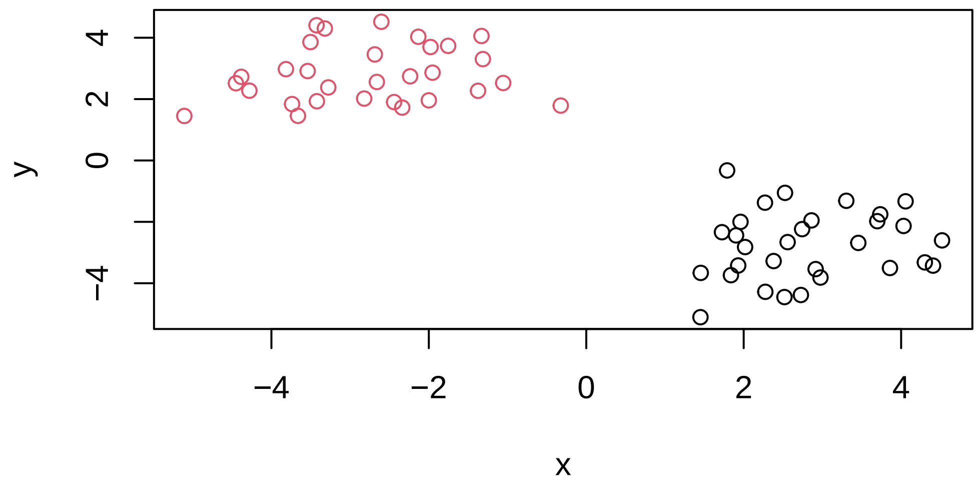


I can now 'cut' my tree with "cutree()' to yield a clister membership vector

```
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 2 2
```

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



You can also tell "cutree()" to cut where it yields k groups

```
cutree(hc, 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
```

#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?

find the number of rows and columns

```
dim(x)
```

```
[1] 17  5
```

Preview the first 6 rows

```r
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
```

fix the labeling of the rows, they are set as the first column of our x dataframe instead of proper row names

```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
#now re check number of rows and columns to show it's corrected
dim(x)
```

```
[1] 17  4
```

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

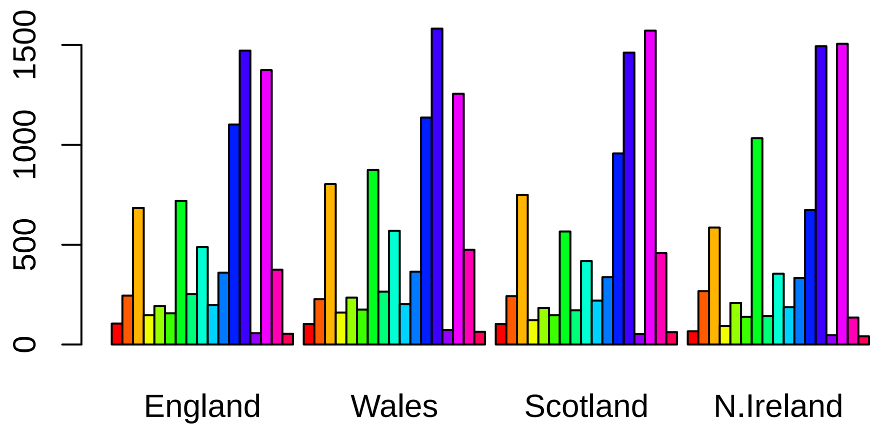
```
               England Wales Scotland N.Ireland
Cheese             105   103      103        66
Carcass_meat       245   227      242       267
```

```
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 is prefered because if you run the code x<-x[,-1] multiple times, it will keep shifting the columns

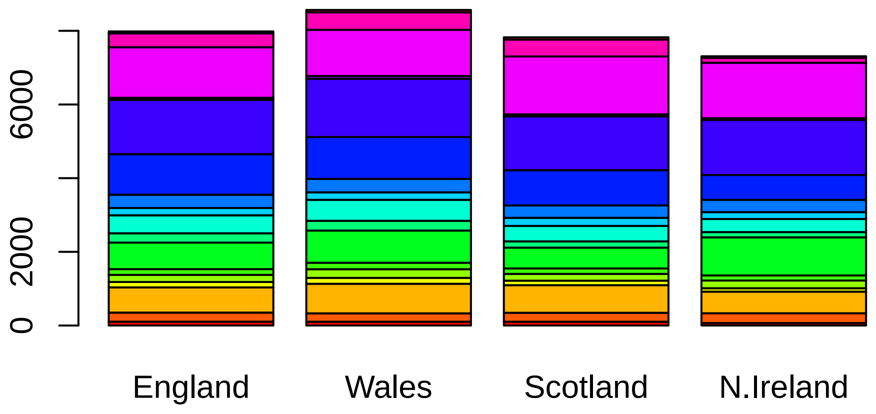
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

can change the beside factor to make it stacked

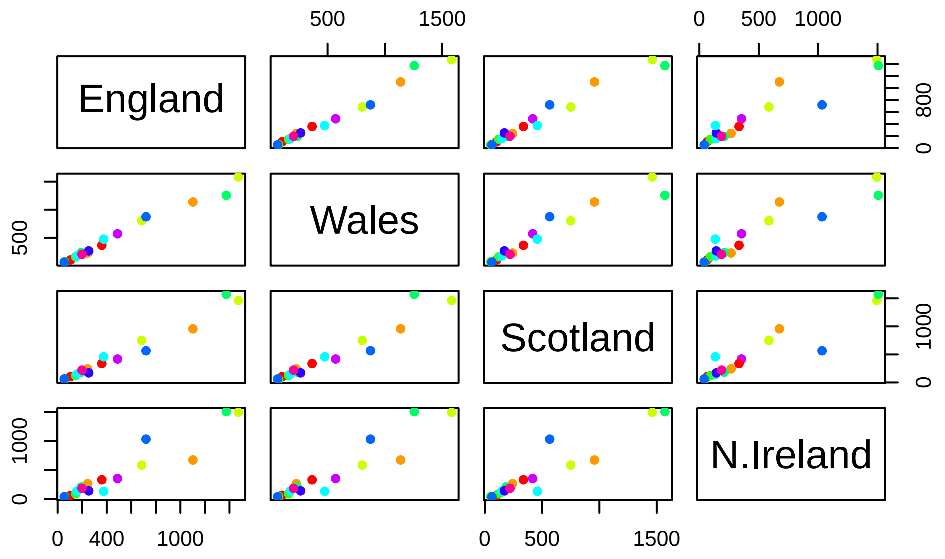
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



> 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?

This figure shows all countries compared separately ind eparate plots, and you can mathc up their x and y axis. If a given point lies on the diagonal for a given plot, it means that at that point, both countries have the same value/correlate.

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



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

Northern Ireland has lower values on average than other countries of the UK

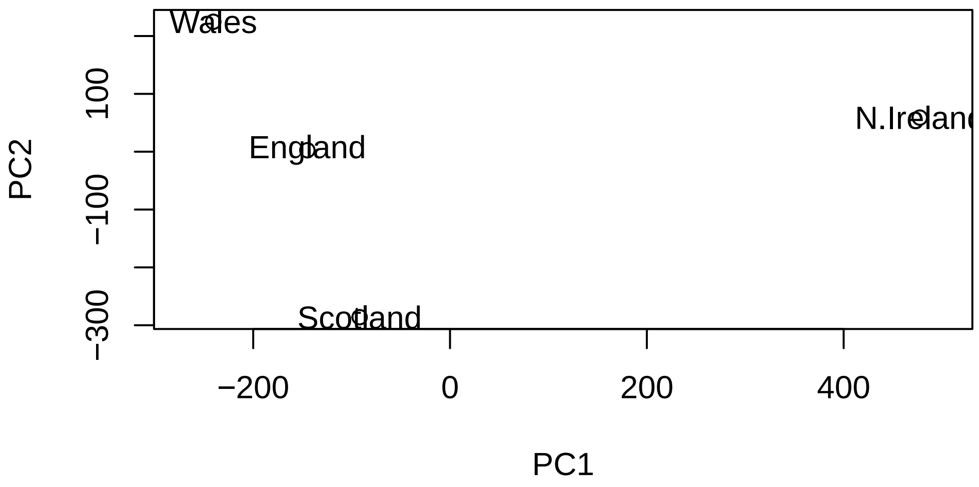
#Lets use PCA to make this plot easier

```
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
```

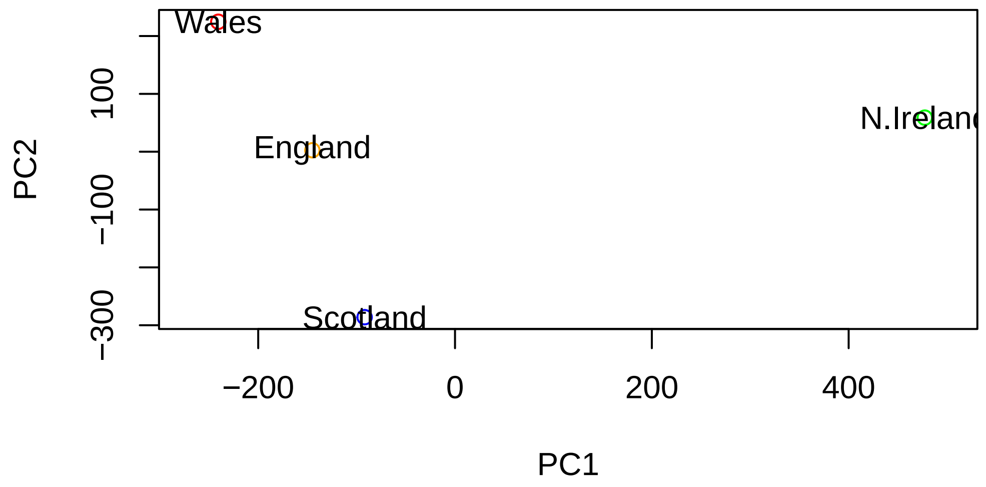
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

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



add colors

```
plot(pca$x[,1], pca$x[,2], col= c("orange", "red", "blue", "green"), xlab="PC1", ylab="PC2
text(pca$x[,1], pca$x[,2], colnames(x))
```


Wales
England
N.Irelanc
Scotland
PC1
PC2
−200
0
200
400
100
−100
−300