Class07: Clustering and PCA

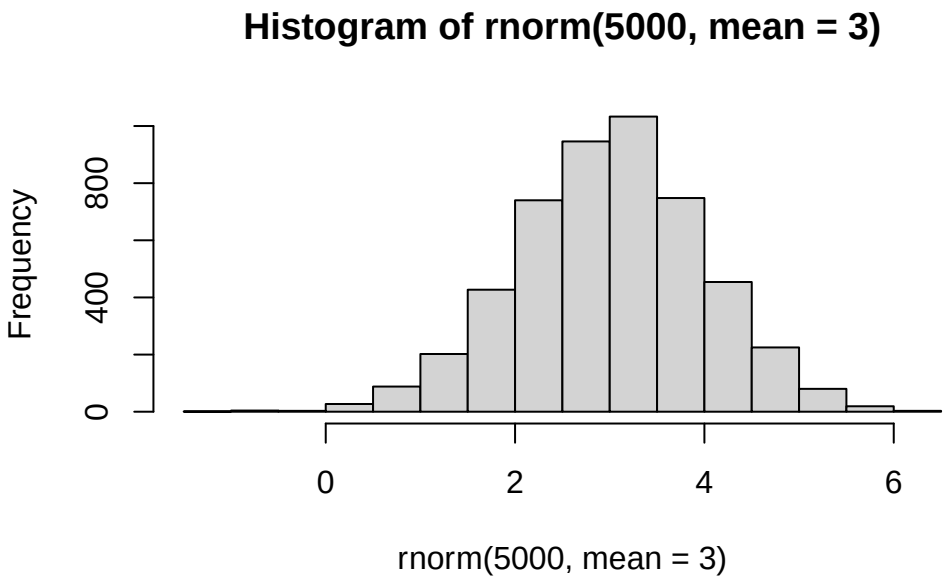
Olivia

Clustering

First let's make up some data to cluster so we can get a fell for these methods and how to work with them.

We can use the`rnorm()` function to get random numbers from a normal distribution at around a given `mean`.

```
hist( rnorm(5000, mean=3) )
```



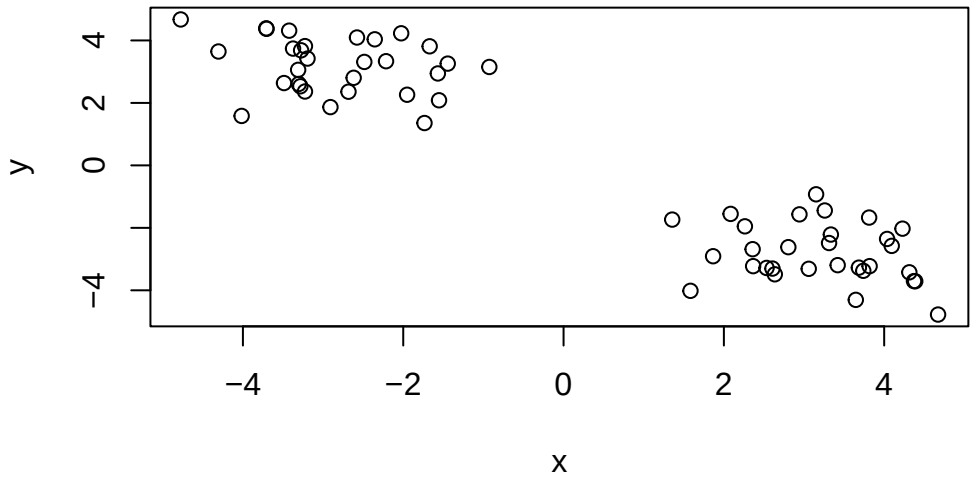
Let's get 30 points with a mean of 3

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

```
 [1]  2.8041113  3.8183700  2.6064165  4.3137529  3.0586687  1.8655621
 [7]  4.0361195  4.6722448  3.2585334  4.3716836  3.4198525  3.6841029
[13]  1.3558201  2.3649261  3.1506467  1.5833904  4.0940074  2.0840628
[19]  3.6457992  3.8122093  4.3874143  3.3123498  4.2284892  2.5325646
[25]  3.3349540  2.3586990  2.6355522  3.7395266  2.9430692  2.2621936
[31] -1.9518602 -1.5681348 -3.3748320 -3.4875991 -2.6831799 -2.2147789
[37] -3.2833402 -2.0255323 -2.4857330 -3.7092548 -1.6704349 -4.3052787
[43] -1.5534523 -2.5769982 -4.0159996 -0.9260263 -3.2259944 -1.7350639
[49] -3.2742743 -3.1942566 -3.7050400 -1.4446787 -4.7759711 -2.3553137
[55] -2.9083664 -3.3112526 -3.4231649 -3.3035706 -3.2262055 -2.6186734
```

Put two of these together

```
x <- cbind(x=tmp, y=rev(tmp))
plot(x)
```



K-means clustering.

Very popular cluestering method 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  3.191170 -2.811142
2 -2.811142  3.191170

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] 47.86607 47.86607
 (between_SS / total_SS =  91.9 %)

Available components:

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

Q. How many points are in each cluster?

```
km$size
```

```
[1] 30 30
```

Q. cluster assignment/membership is

```
km$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
```

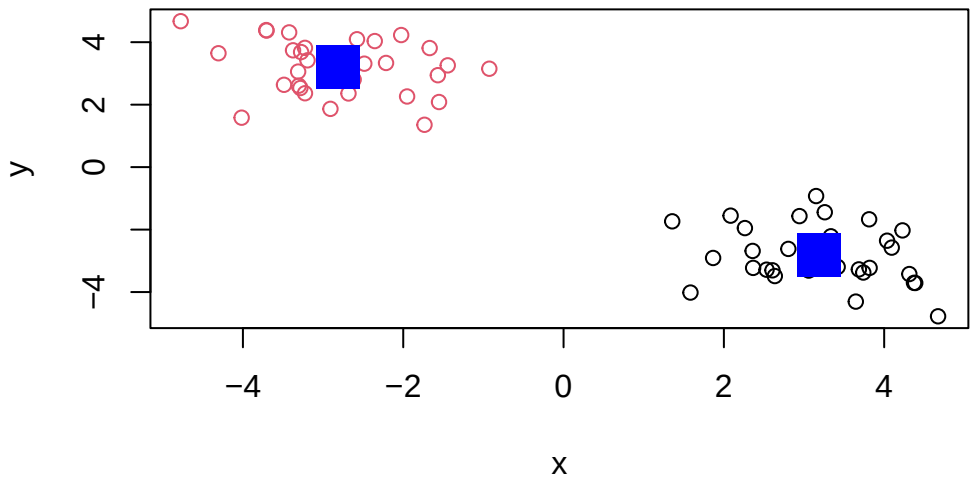
Q. cluster center is

```
km$centers
```

```
          x         y
1  3.191170 -2.811142
2 -2.811142  3.191170
```

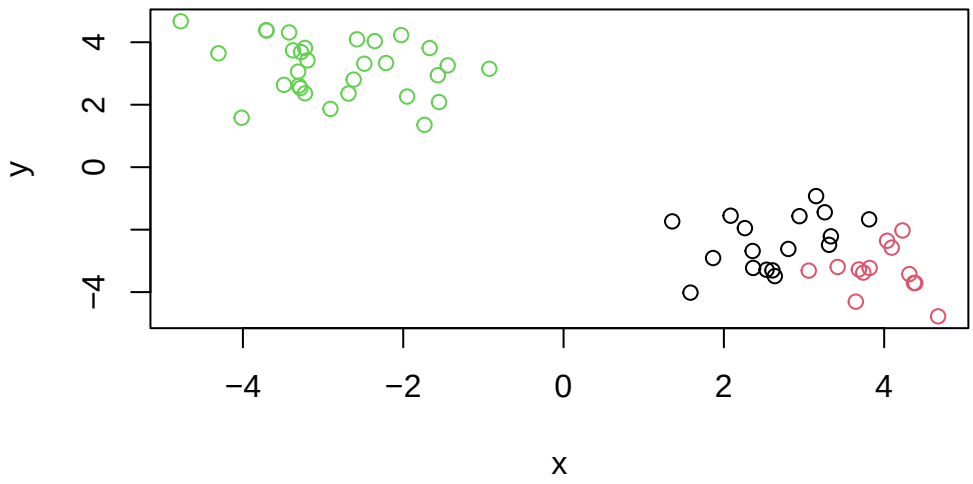
Q. Plot x colored by the means cluster assignment and add cluster centers as blue points

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



Q. Let's cluster into 3 groups for same x data and make a plot.

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



Hierarchical Clustering

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

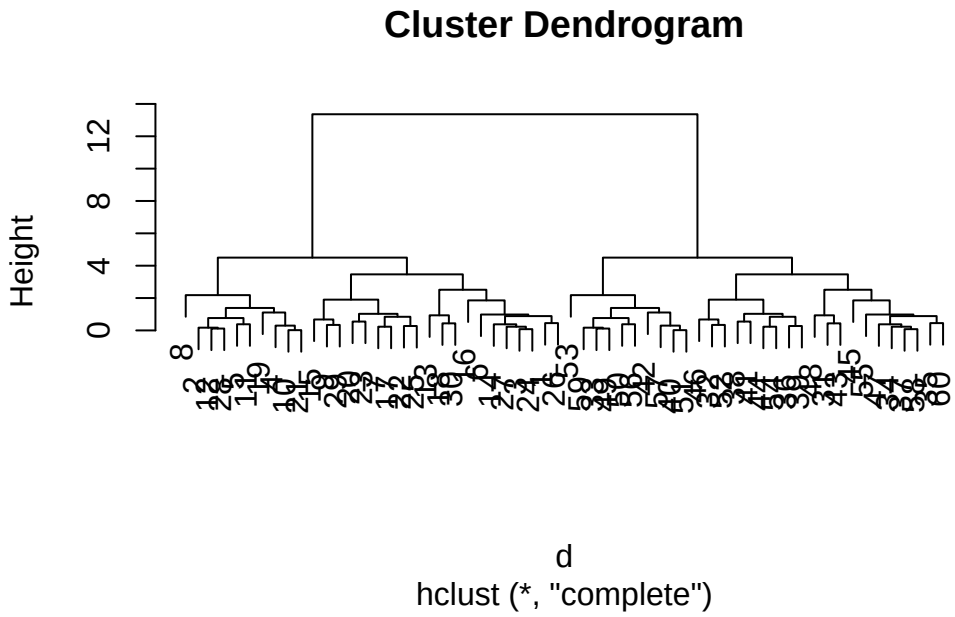
We will use the `dist()` function to start with.

```
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 the `cutree()`to yield a culster membership vector.

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

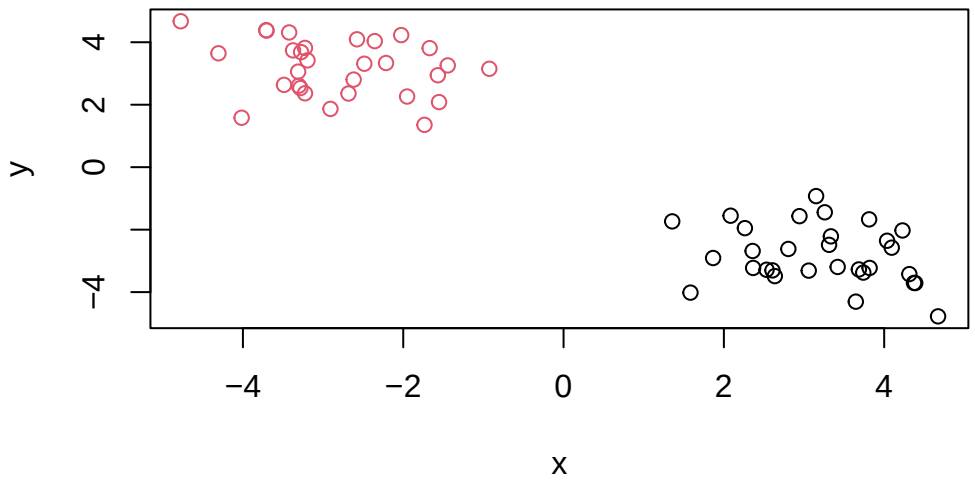
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

You can also tell `cutree` to cut where it yields "k" groups.

```
grps <- cutree(hc, k=2)
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 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)
```



Principal Component Analysis (PCA)

New low dimensional acis (or surfaces) closest to the observations. Can be compared to best-fit lines. PC1 goes from left to right PC2 goes from up to down

Class 7

Data import

```
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?

```
#number of rows = 17
nrow(x)
```

```
[1] 17
```

```
#number of columnns = 5
ncol(x)
```

```
[1] 5
```

```
#number of rows and columns
dim(x)
```

```
[1] 17  5
```

Examining data

```
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
```

```
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
```

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

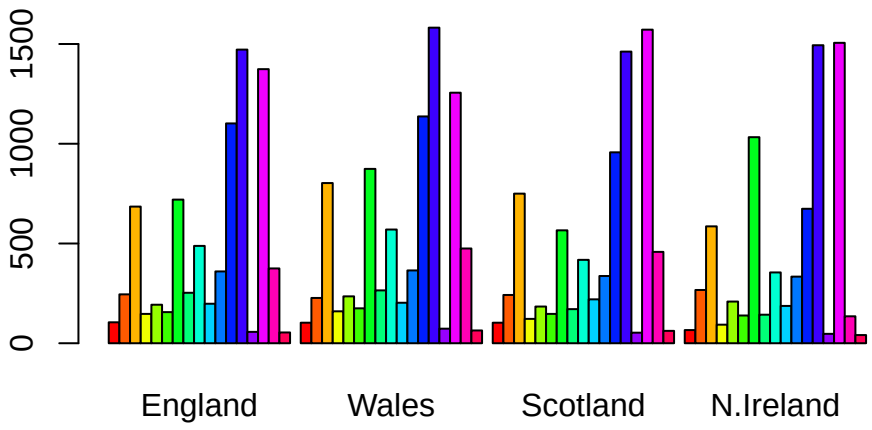
	England	Wales	Scotland	N.Ireland
Cheese	105	103	103	66
Carcass_meat	245	227	242	267
Other_meat	685	803	750	586
Fish	147	160	122	93
Fats_and_oils	193	235	184	209
Sugars	156	175	147	139

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?

I prefer using `x <- read.csv(url, row.names=1) head(x)` because it doesn't mix up the amount of rows and columns.

Exploratory analysis

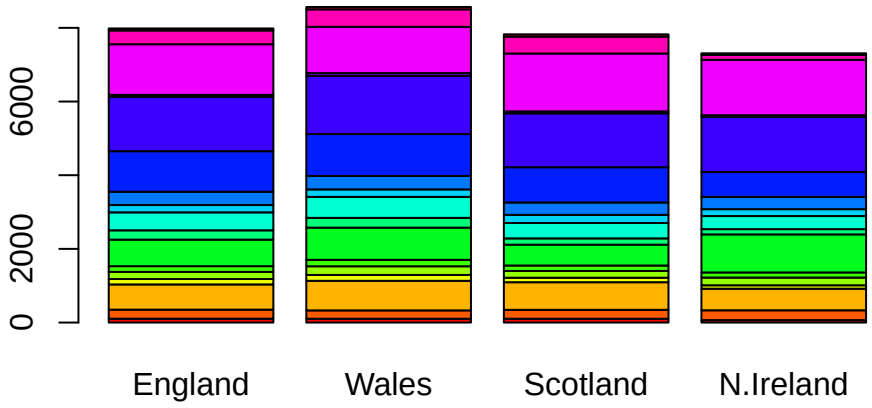
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

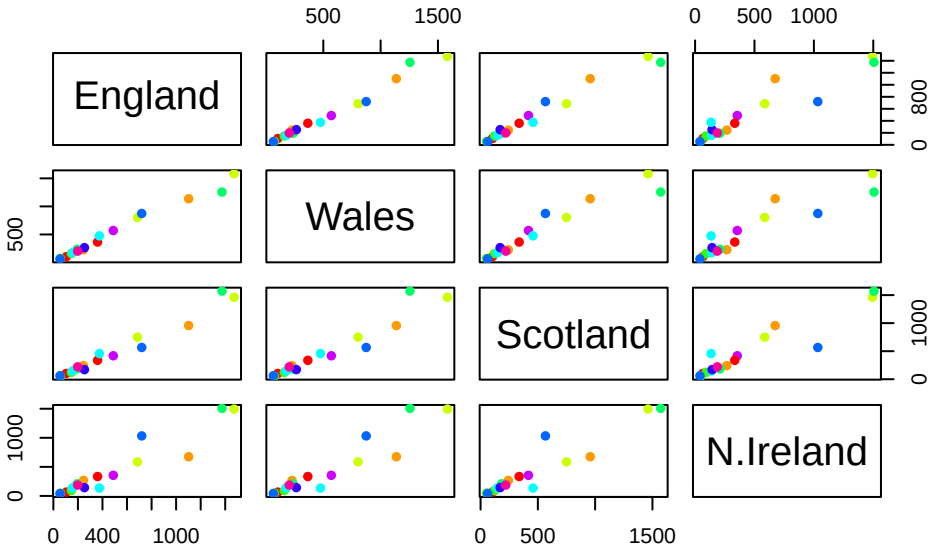
Changing argument `beside=T` to `beside=F`

```
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?

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



The x-axis and y-axis changes depending on the plot that is being examined. If a point lies in the diagonal for a given plot, it means that one of the countries eats more or less of one of the foods than the other.

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

Ireland seems to differ on the amount of food corresponding by the blue and orange data points since these are further from the other points.

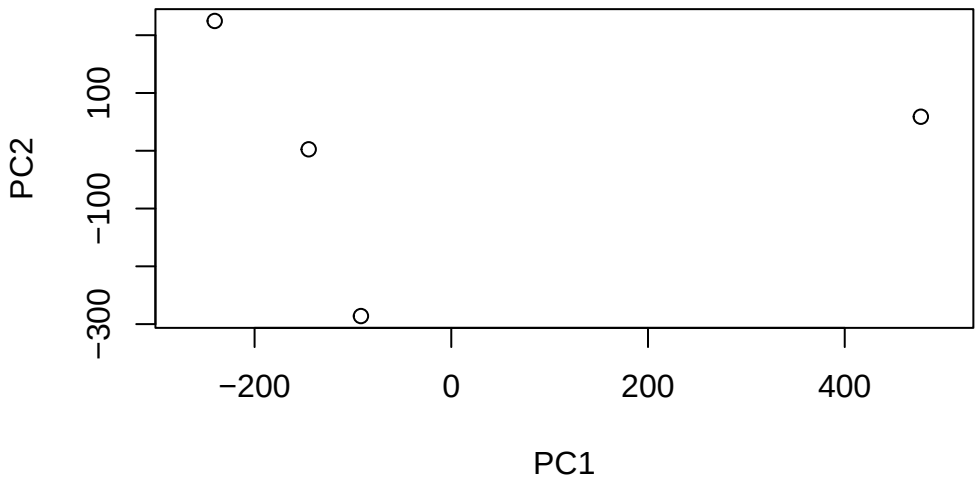
The main PCA function in base R is called prcomp() it expects the transpose of our data.

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

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 5.552e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

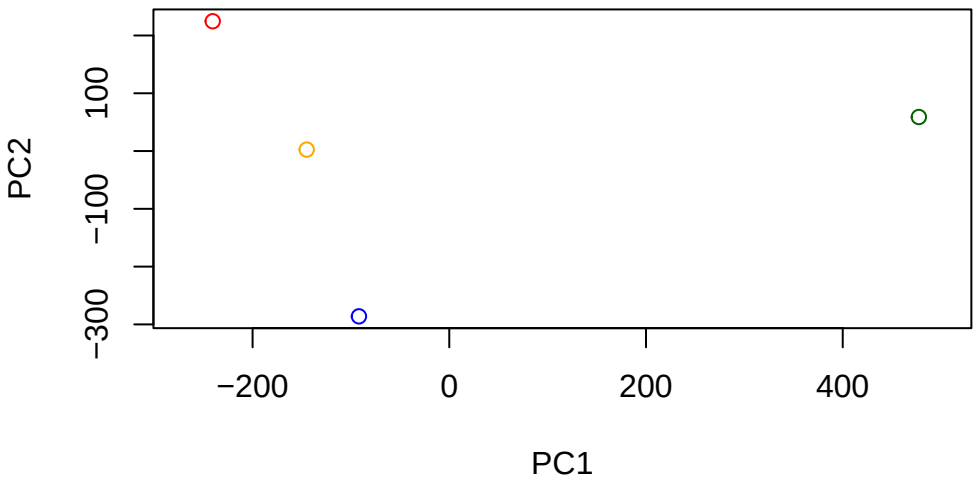
> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

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



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.

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


PC2
100
−100
−300
−200
0
200
400
PC1