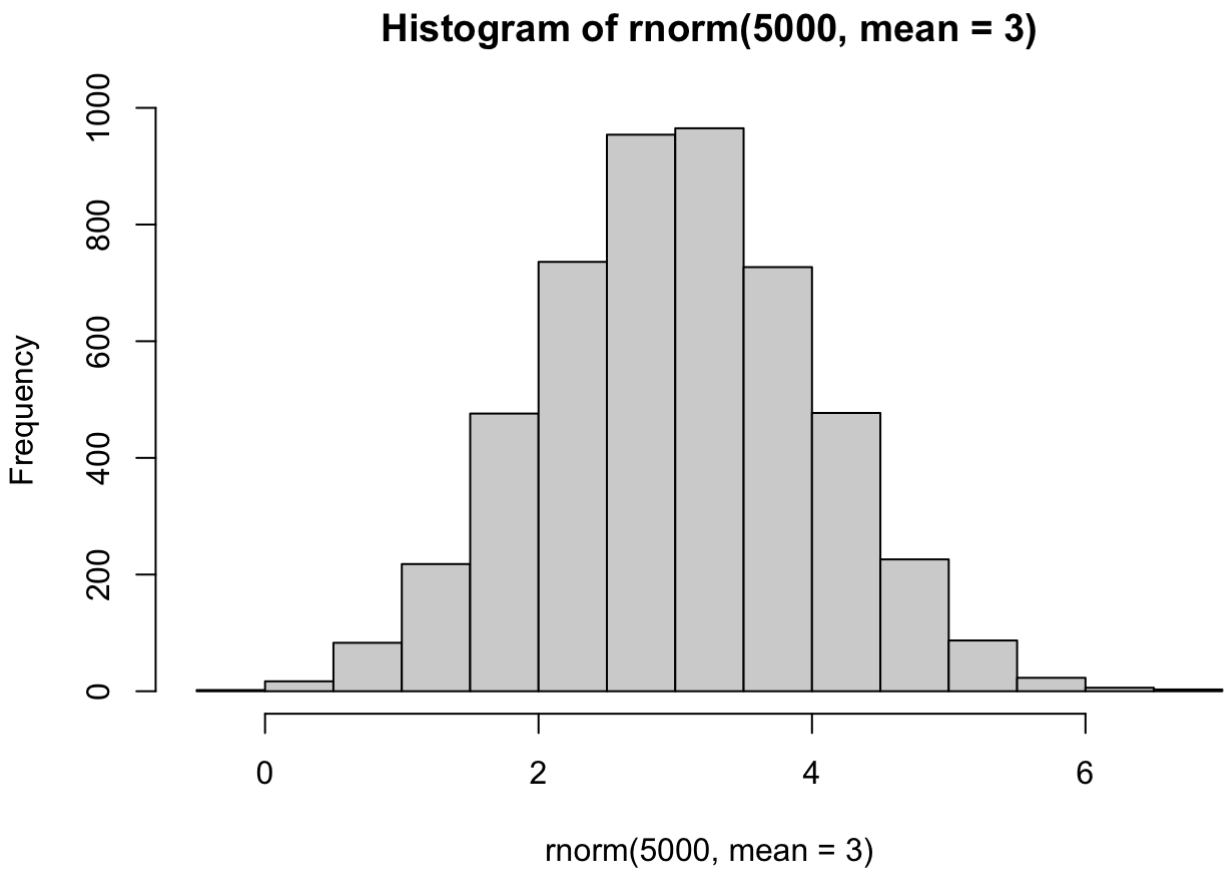
Lab 7: Clustering and PCA

Clustering

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

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

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



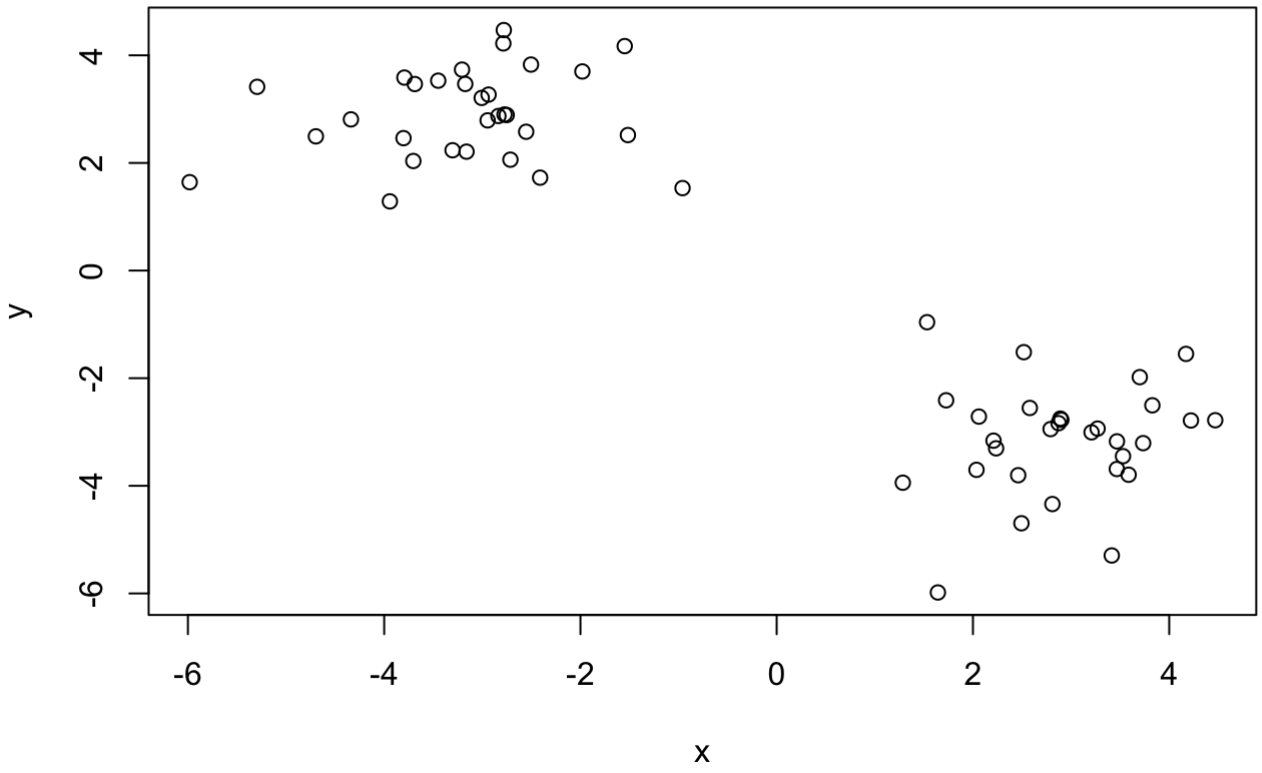
Let's get 30 points with a mean of 3.

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

```
 [1]  3.4675540  2.2358487  1.6428127  3.5290423  2.8904724  3.5862556
 [7]  2.5181981  3.8280897  2.2097611  2.4610910  4.2214246  2.5795412
[13]  2.8726001  3.4144435  2.0606589  4.1712494  2.4939869  2.0361017
[19]  3.2086277  2.7910870  3.7343795  1.2854463  1.7265486  2.8094931
```

```
[25]  3.4661329  3.2701950  4.4689268  1.5325020  2.9008057  3.7000597
[31] -1.9803525 -2.7741061 -0.9594288 -2.7812955 -2.9365751 -3.6879142
[37] -4.3389074 -2.4110624 -3.9420554 -3.2068732 -2.9460017 -3.0062377
[43] -3.7031650 -4.6961238 -1.5488891 -2.7138970 -5.2944767 -2.8360773
[49] -2.5532545 -2.7859651 -3.8032902 -3.1611604 -2.5041169 -1.5164714
[55] -3.7941290 -2.7514340 -3.4495228 -5.9821138 -3.3033728 -3.1740984
```

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



K-means clustering.

Very popular clustering 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  2.903778 -3.151412
```

```
2 -3.151412  2.903778

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 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] 52.745 52.745
 (between_SS / total_SS =  91.2 %)

Available components:

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

> Q. What is the size?

```
km$size
```

```
[1] 30 30
```

> Q. What is the cluster/assignment?

```
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 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
```

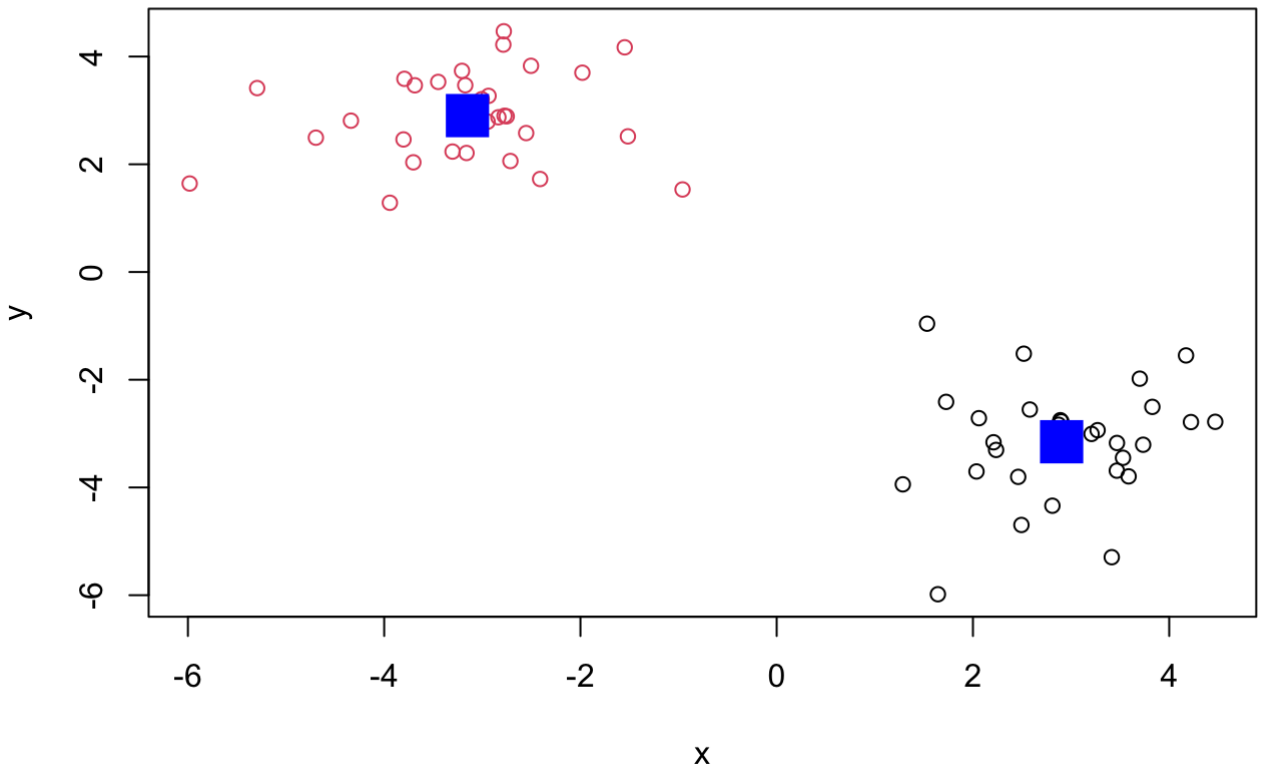
> Q. Cluster center?

```
km$centers
```

```
          x         y
1  2.903778 -3.151412
2 -3.151412  2.903778
```

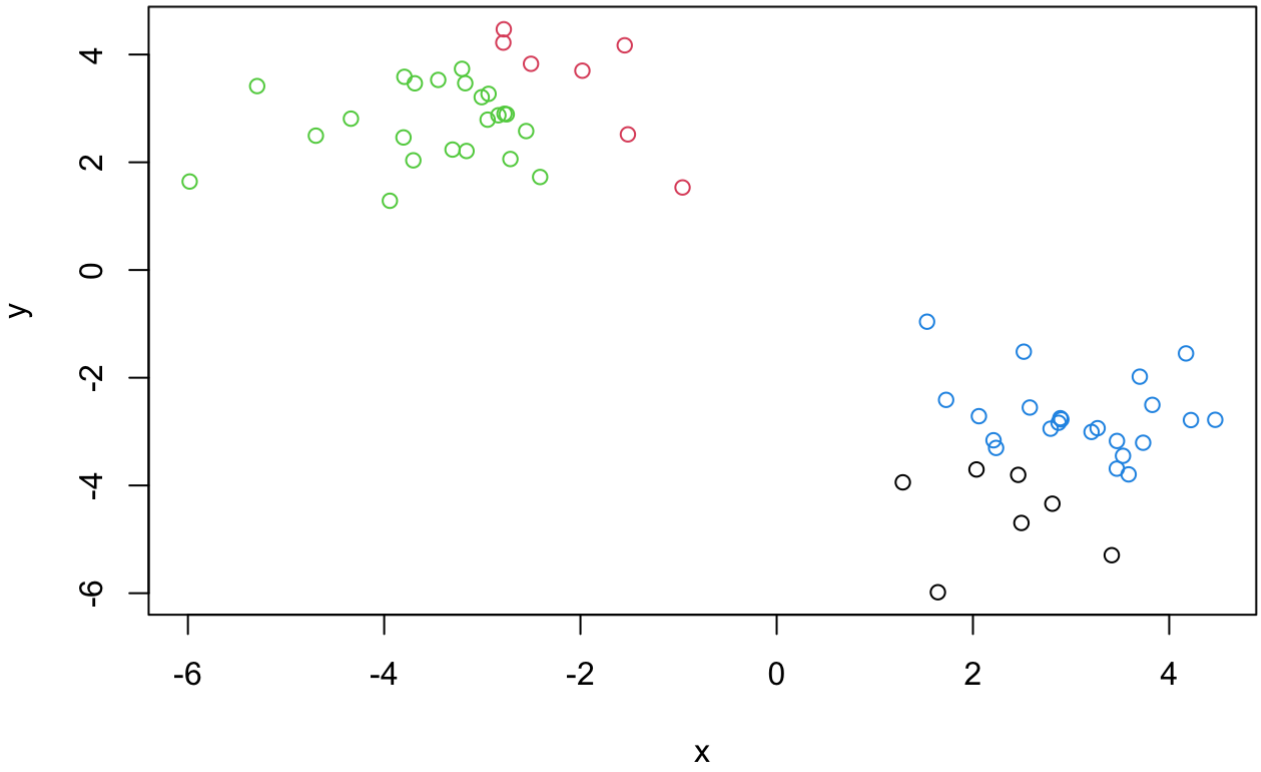
> Q.

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



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

```
km <- kmeans(x, centers =4)
plot(x, col=km$cluster)
```



#Hirerarchial 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"

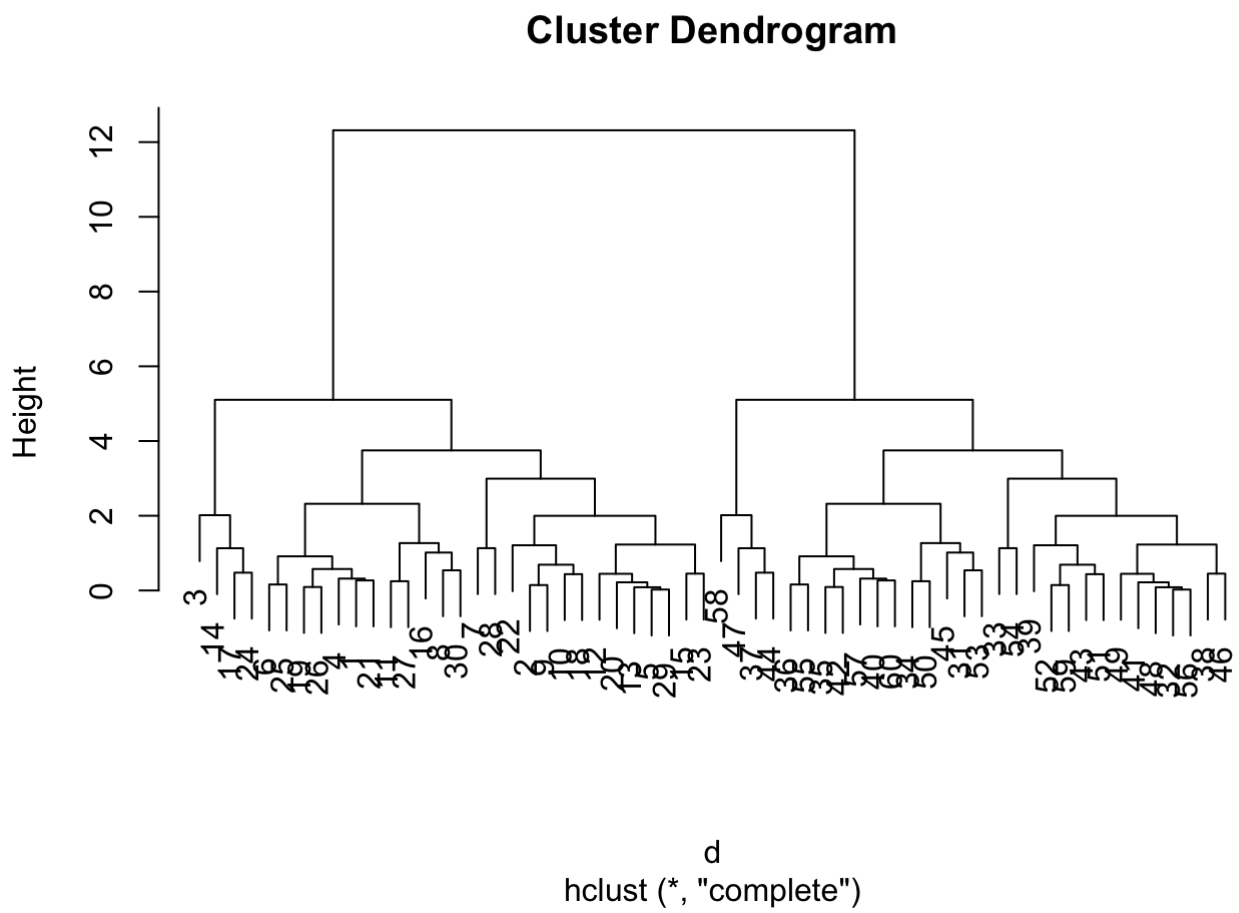
We will use the `dist()` function to start with.

```
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 cluster membership vector.

```
grps <- cutree(hc, h = 8)
grps
```

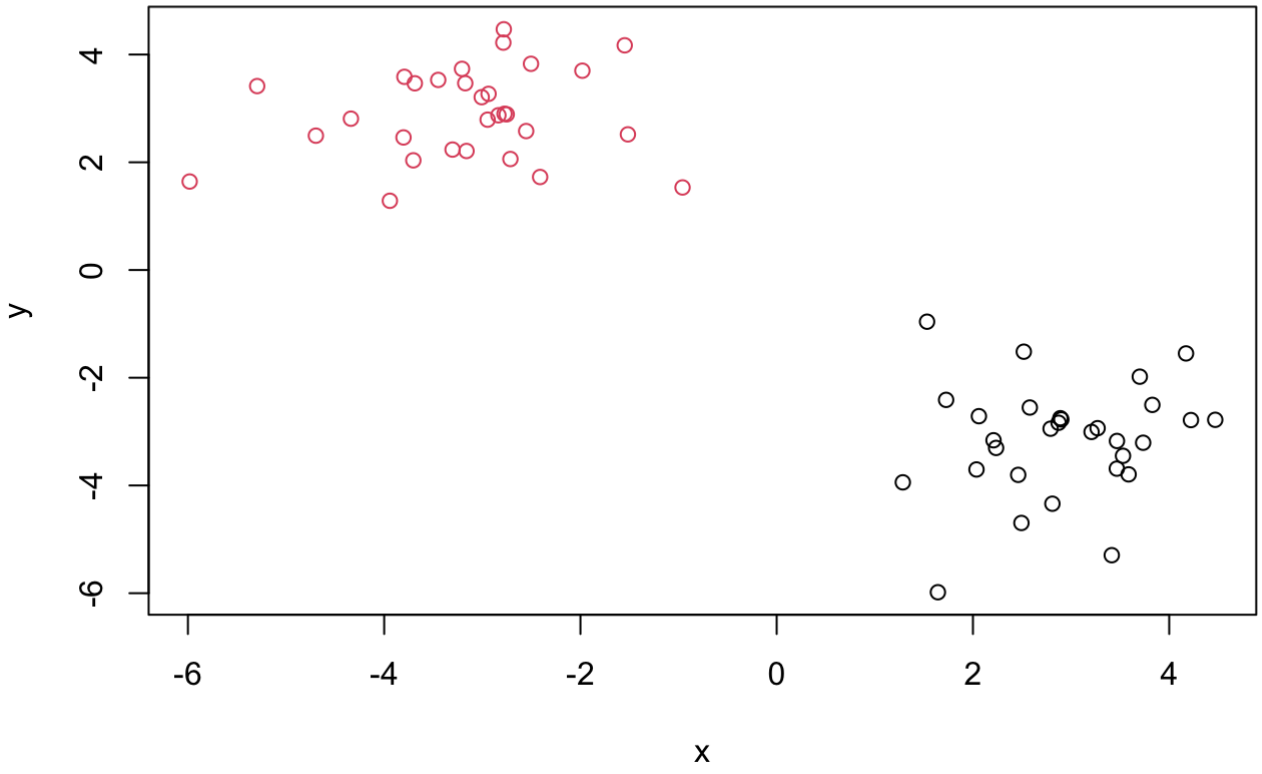
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

You can also tell `cutree()` to cut where it yields “k” groups.

```
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 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)

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

> Q1

Complete the following code to find out how many rows and columns are in x?

```
dim(ukfoods)
```

```
[1] 17  5
```

Preview the first 6 rows

```
View(ukfoods)
```

```
# Note how the minus indexing works
rownames(ukfoods) <- ukfoods[,1]
ukfoods <- ukfoods[,-1]
head(ukfoods)
```

```
              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
```

```
dim(ukfoods)
```

```
[1] 17  4
```

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

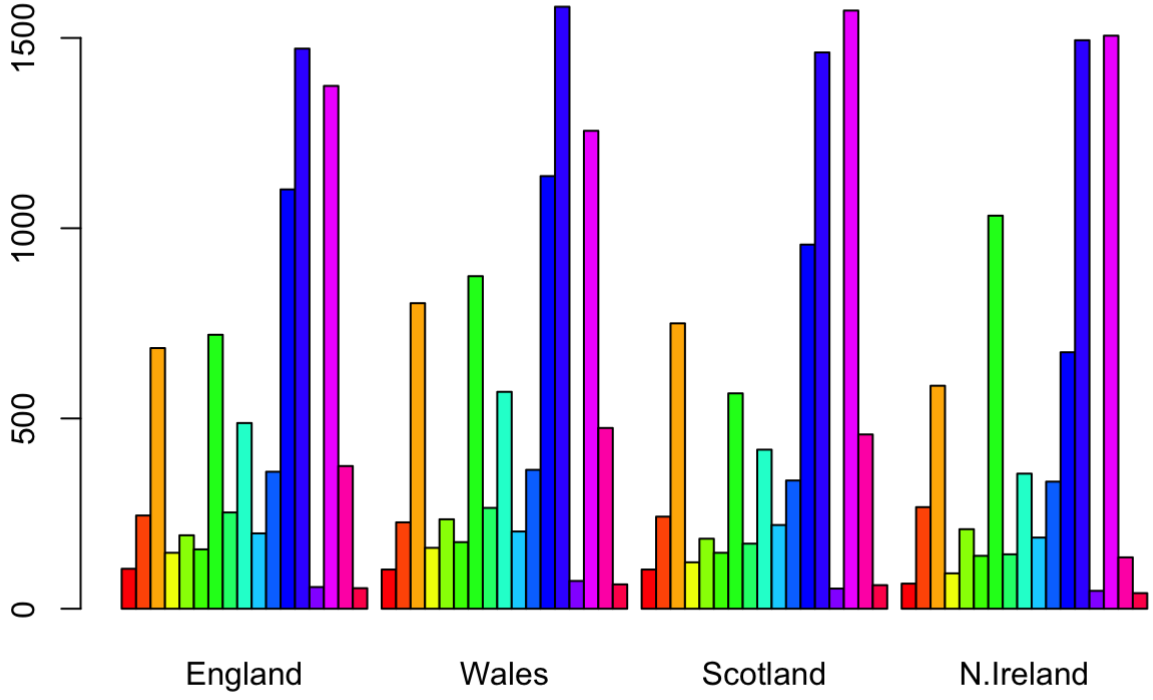
```
              England Wales Scotland N.Ireland
Cheese            105   103      103        66
Carcass_meat      245   227      242       267
Other_meat        685   803      750       586
Fish              147   160      122        93
Fats_and_oils     193   235      184       209
Sugars        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

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

Less destructive

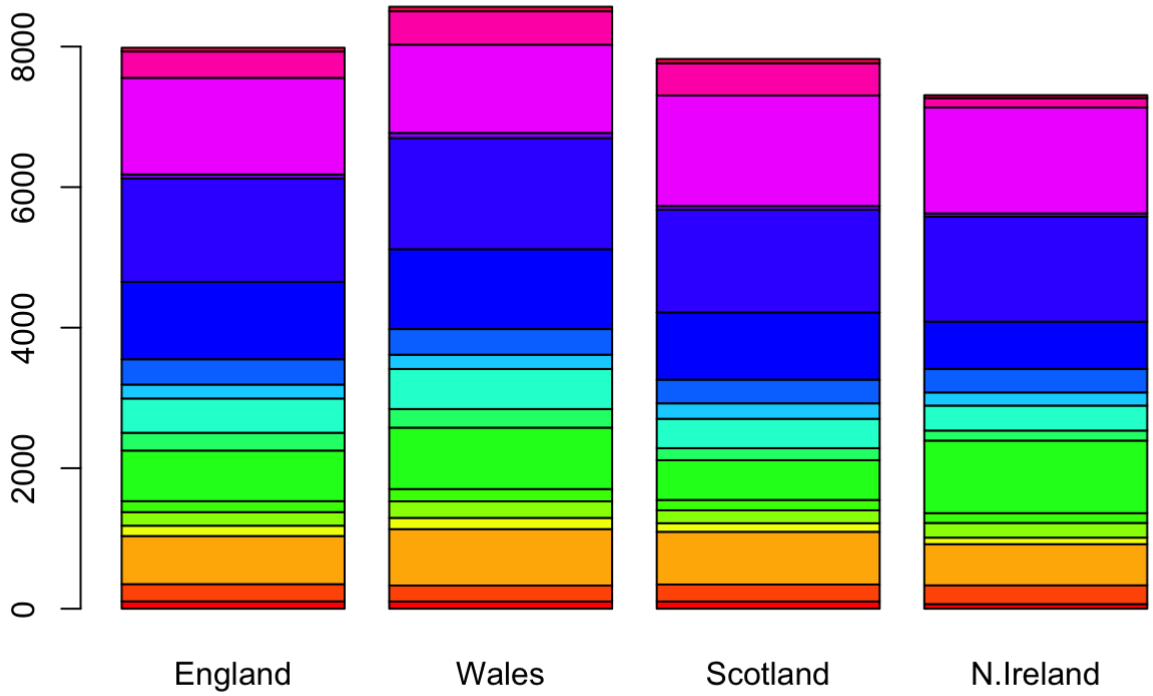
```
barplot(as.matrix(ukfoods), beside=T, col=rainbow(nrow(ukfoods)))
```



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

Changing beside = TRUE to FALSE

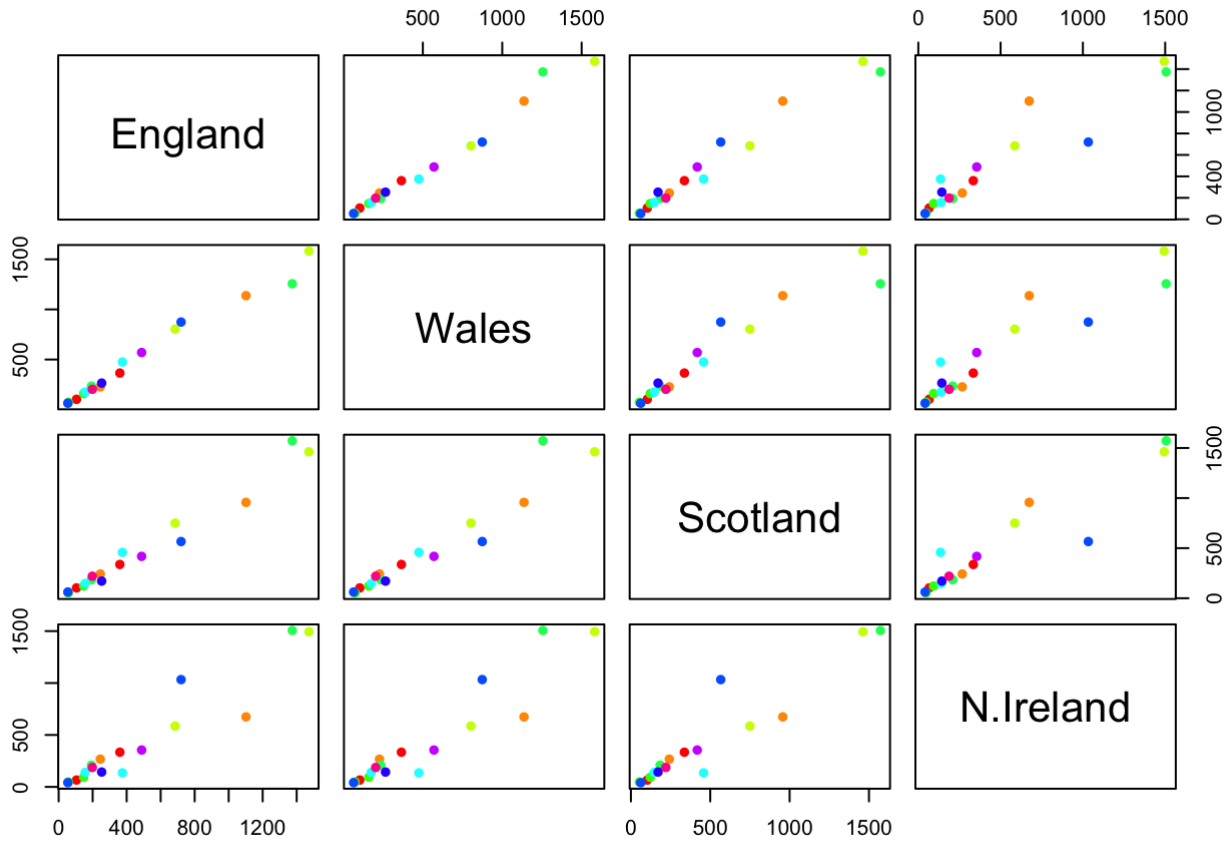
```
barplot(as.matrix(ukfoods), beside=F, col=rainbow(nrow(ukfoods)))
```



> 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?

If a given point falls on the diagonal, this would suggest some sort of correlation in tendency to consume a given product compared between two countries.

```
pairs(ukfoods, 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?

The blue and orange points differ the most compared the other countries.

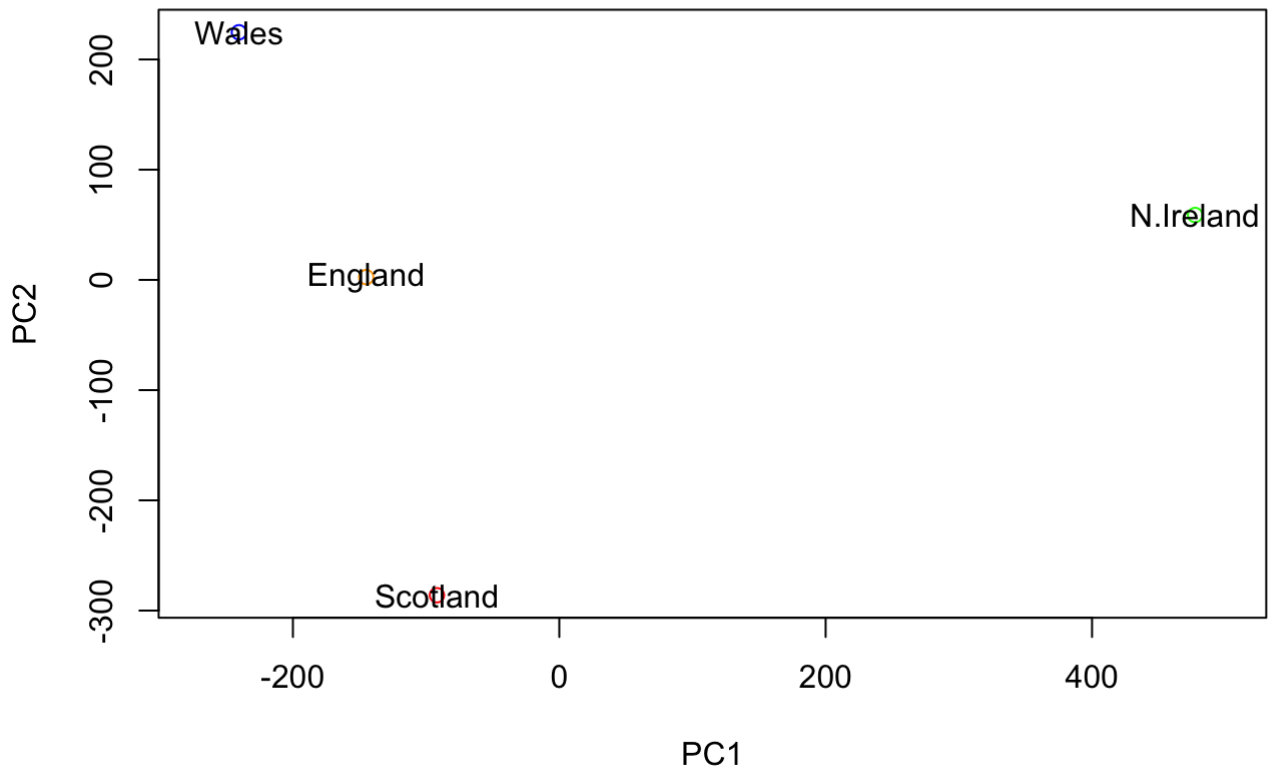
```
# Use the prcomp() PCA function
pca <- prcomp( t(ukfoods) )
summary(pca)
```

```
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
```

```
pca$x
```

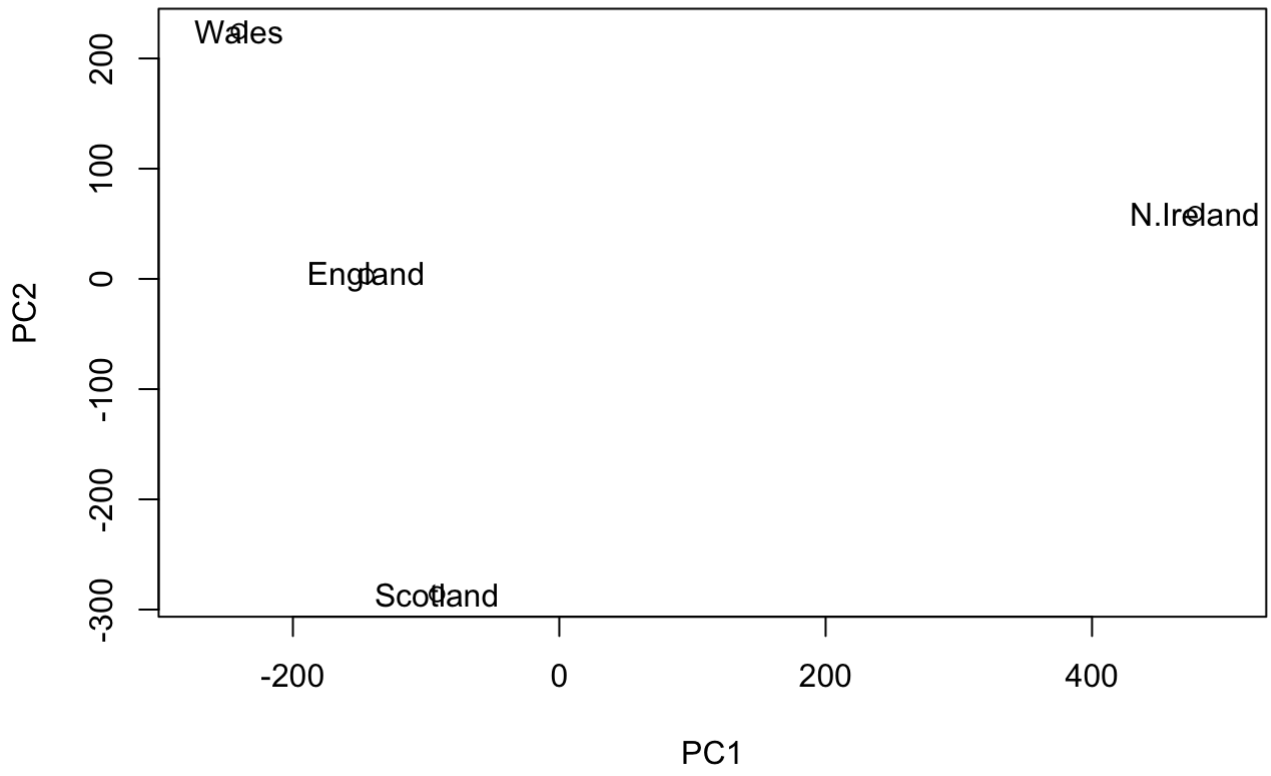
```
                PC1         PC2         PC3           PC4
England    -144.99315    2.532999 -105.768945  1.042460e-14
Wales      -240.52915  224.646925   56.475555  9.556806e-13
Scotland    -91.86934 -286.081786   44.415495 -1.257152e-12
N.Ireland   477.39164   58.901862    4.877895  2.872787e-13
```

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



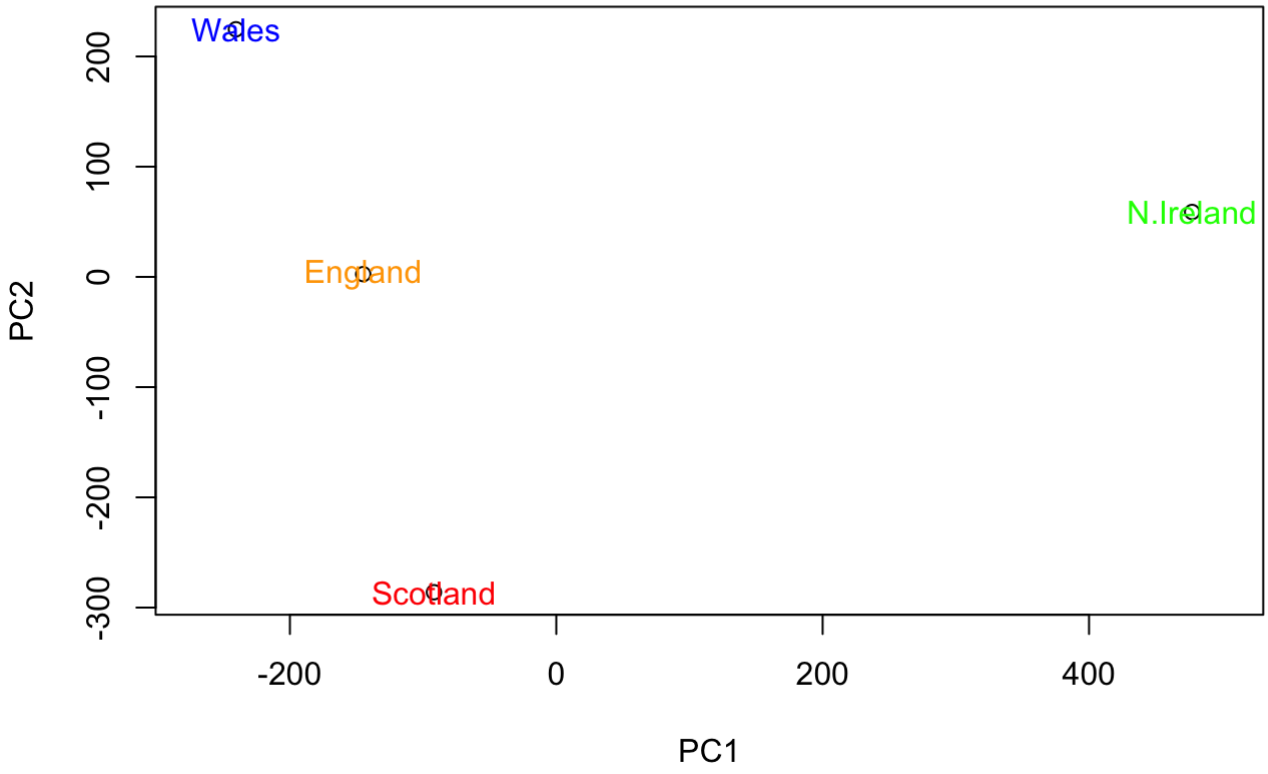
> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
# 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(ukfoods))
```



> 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 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(ukfoods), col= c("orange", "blue", "red", "green"))
```


Wales
England
N.Ireland
Scotland
PC2
PC1
200
100
0
-100
-200
-300
-200
0
200
400