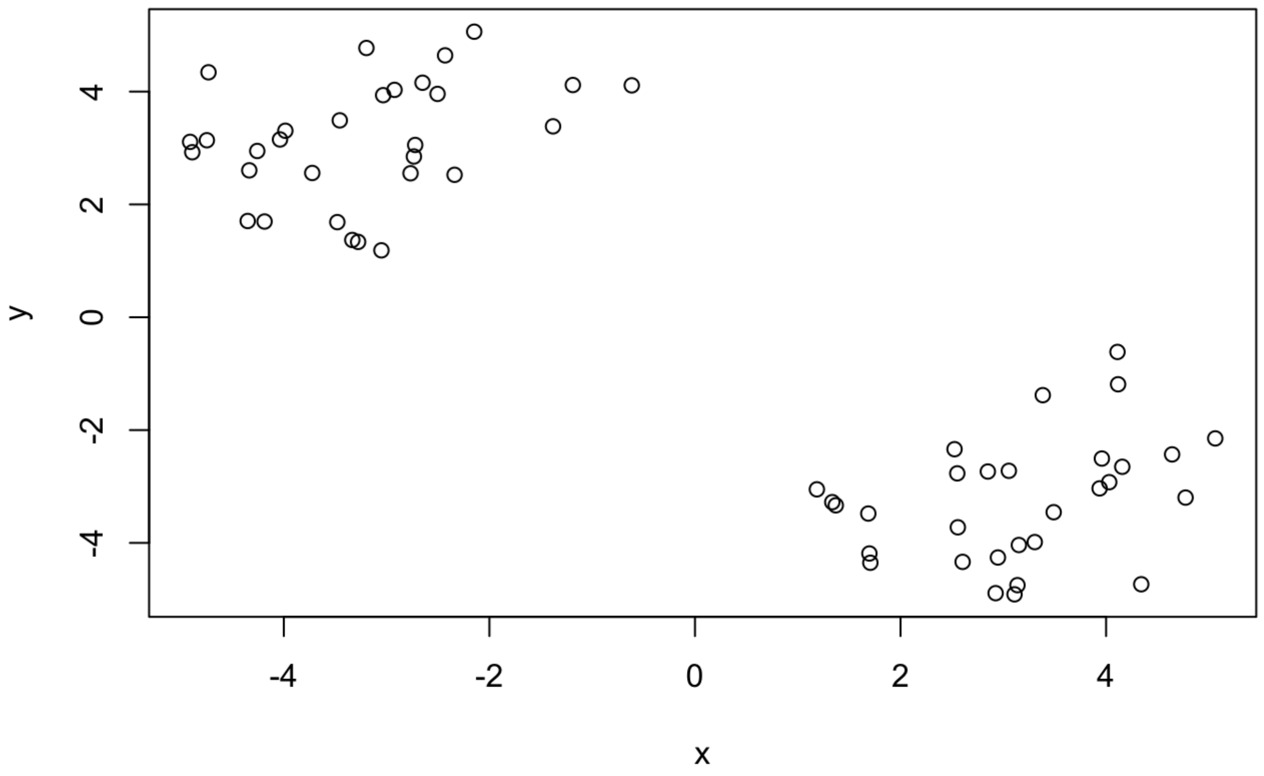
Class 7: Machine Learning

AUTHOR
Nichelle Camden

K-means Clustering

Let’s make up some data to cluster.

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



The function to do k-means clustering in base R is called `kmeans()`. We give this our input data for clustering and the number of clusters we want `centers`.

```
km <- kmeans(x, centers = 4, nstart = 20)
km
```

```
K-means clustering with 4 clusters of sizes 17, 13, 17, 13

Cluster means:
```

```
          x         y
1  2.535453 -3.972393
2 -2.297487  3.892280
3 -3.972393  2.535453
4  3.892280 -2.297487

Clustering vector:
 [1] 3 2 2 2 2 3 3 2 2 2 2 3 3 2 3 2 3 2 3 3 3 2 3 3 3 3 3 3 2 3 1 4 1 1 1 1 1 1
[39] 4 1 1 1 4 1 4 1 4 1 1 4 4 4 4 1 1 4 4 4 4 1

Within cluster sum of squares by cluster:
[1] 19.97669 14.04267 19.97669 14.04267
 (between_SS / total_SS =  95.0 %)

Available components:

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

```
km$cluster
```

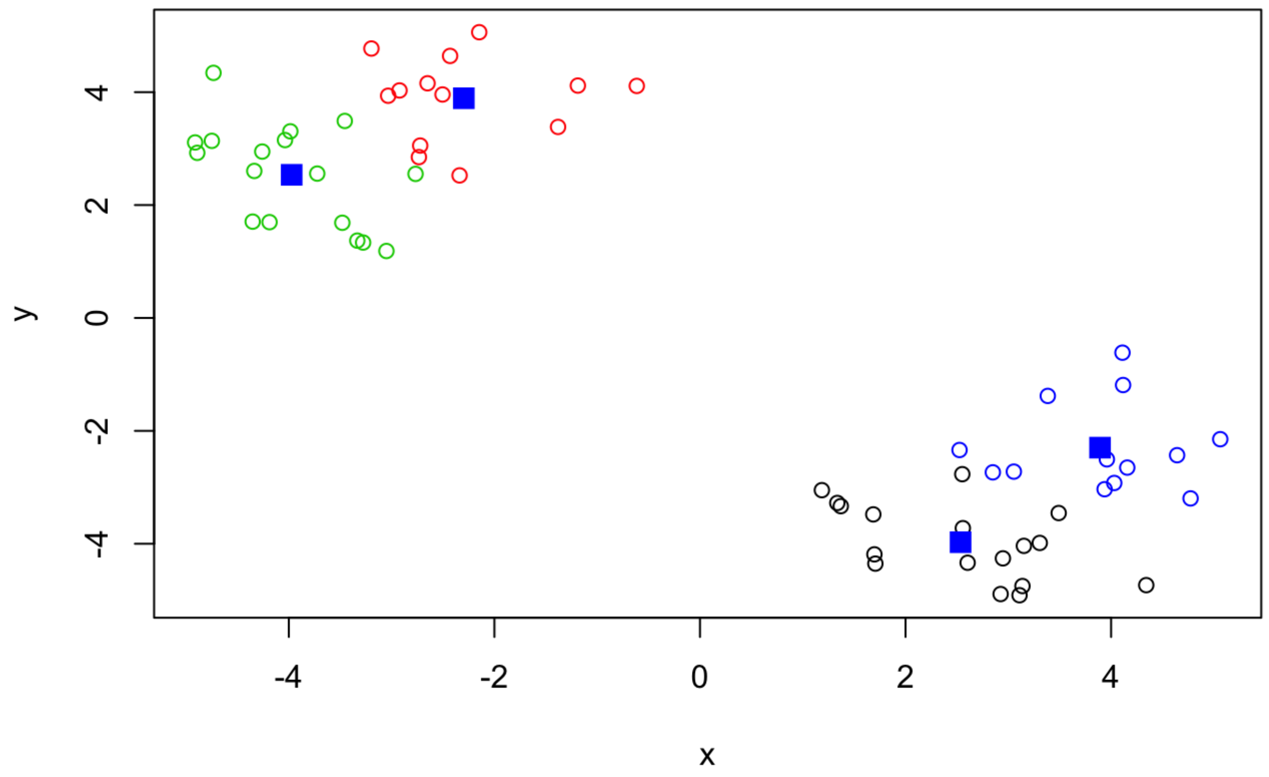
```
 [1] 3 2 2 2 2 3 3 2 2 2 2 3 3 2 3 2 3 2 3 3 3 2 3 3 3 3 3 3 2 3 1 4 1 1 1 1 1 1
[39] 4 1 1 1 4 1 4 1 4 1 1 4 4 4 4 1 1 4 4 4 4 1
```

```
km$centers
```

```
          x         y
1  2.535453 -3.972393
2 -2.297487  3.892280
3 -3.972393  2.535453
4  3.892280 -2.297487
```

> Q. plot x colored by the kmeans cluster assignment and add cluster centers as blue points

```
plot(x, col=km$cluster)
points(km$centers, col="blue", pch=15, cex=1.5)
```



Hierarchical Clusters

The `hclust()` function performs hierarchical clustering. The big advantage here is that I don't need to tell it "k" the number of clusters..

To run `hclust()` I need to provide a distance matrix as input (not the original data).

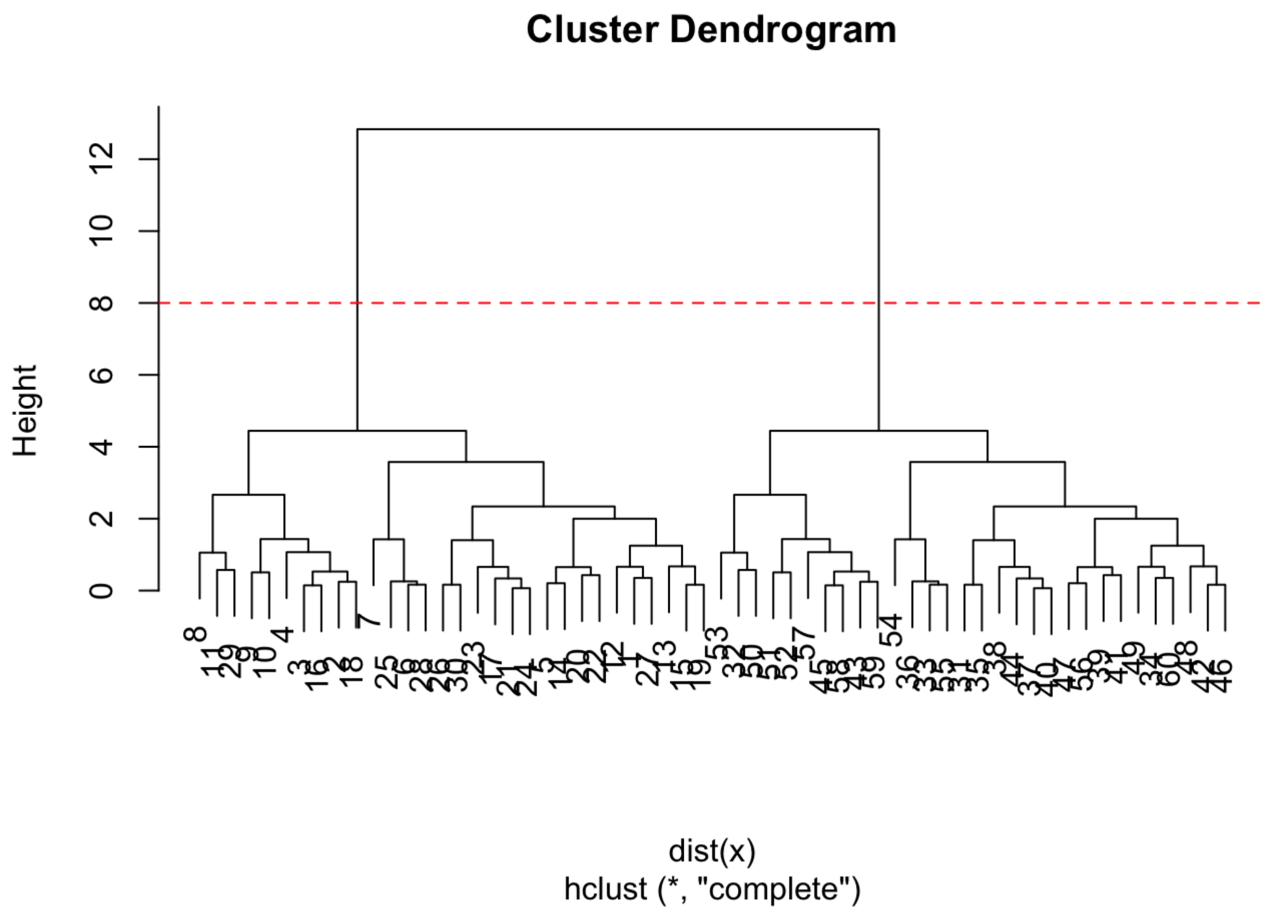
```
hc <- hclust( dist(x))
hc
```

```

Call:
hclust(d = dist(x))

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

```
plot(hc)
abline(h=8, col="red", lty= 2)
```



To get my “main” result (cluster membership) I want to “cut” this tree to yield “branches” whos “leaves” are the members of the cluster.

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

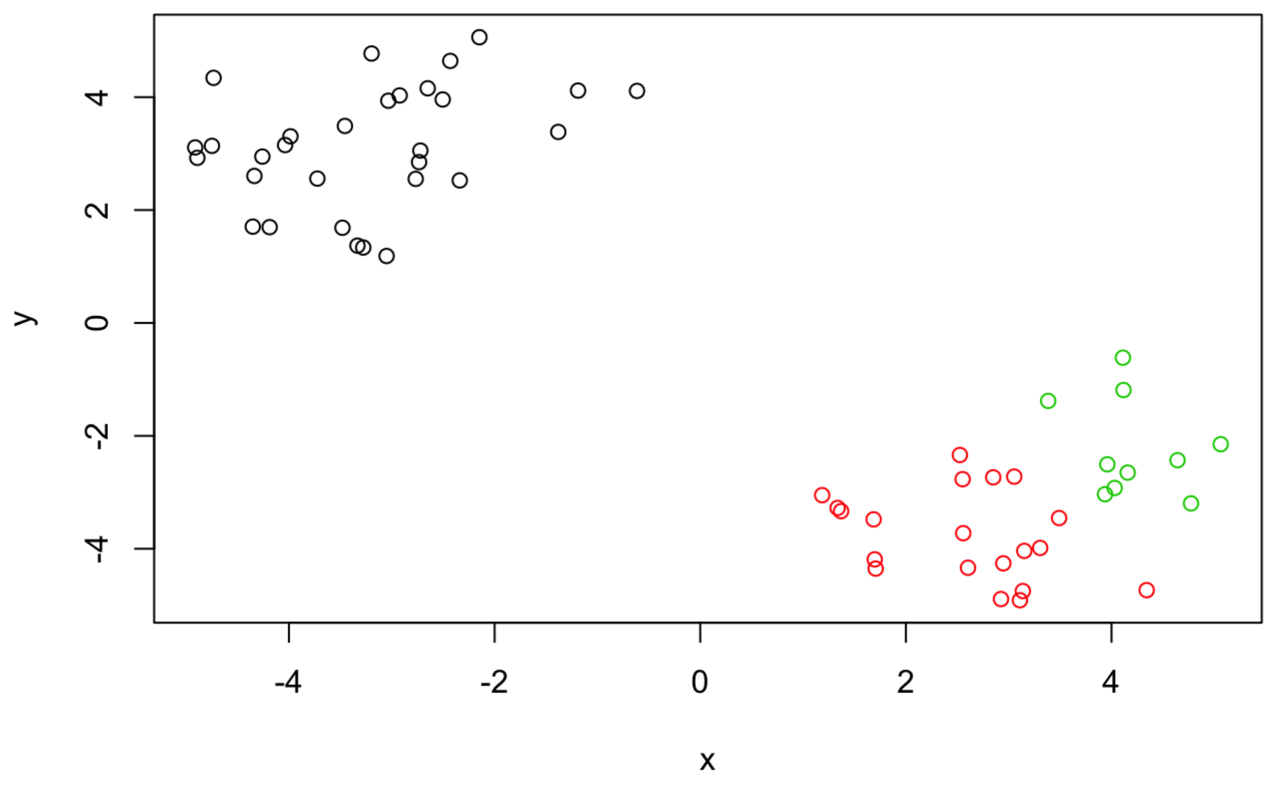
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

More often, we will use `cutree()` with k=2 for example

```
grps <- cutree(hc, k=3)
```

Make a plot of our `hclust` results, i.e. our data colored by cluster assignment

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



Principal Component Analysis (PCA)

Read data for UK food trends from online

```
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?

```
ncol(x)
```

```
[1] 4
```

```
nrow(x)
```

```
[1] 17
```

There are 5 columns and 17 rows. The food column counts as a column, in addition to the four countries. (before arguing with the url and changing the row names)

```
#View(x)
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
```

```
x[,-1] #shows everything except the first row (food)
```

```
                    Wales Scotland N.Ireland
Cheese                103      103        66
Carcass_meat          227      242       267
Other_meat            803      750       586
Fish                  160      122        93
Fats_and_oils         235      184       209
Sugars                175      147       139
Fresh_potatoes        874      566      1033
Fresh_Veg             265      171       143
Other_Veg             570      418       355
Processed_potatoes    203      220       187
Processed_Veg         365      337       334
Fresh_fruit          1137      957       674
Cereals              1582     1462      1494
Beverages              73       53        47
Soft_drinks          1256     1572      1506
```

```
Alcoholic_drinks      475      458       135
Confectionery          64       62        41
```

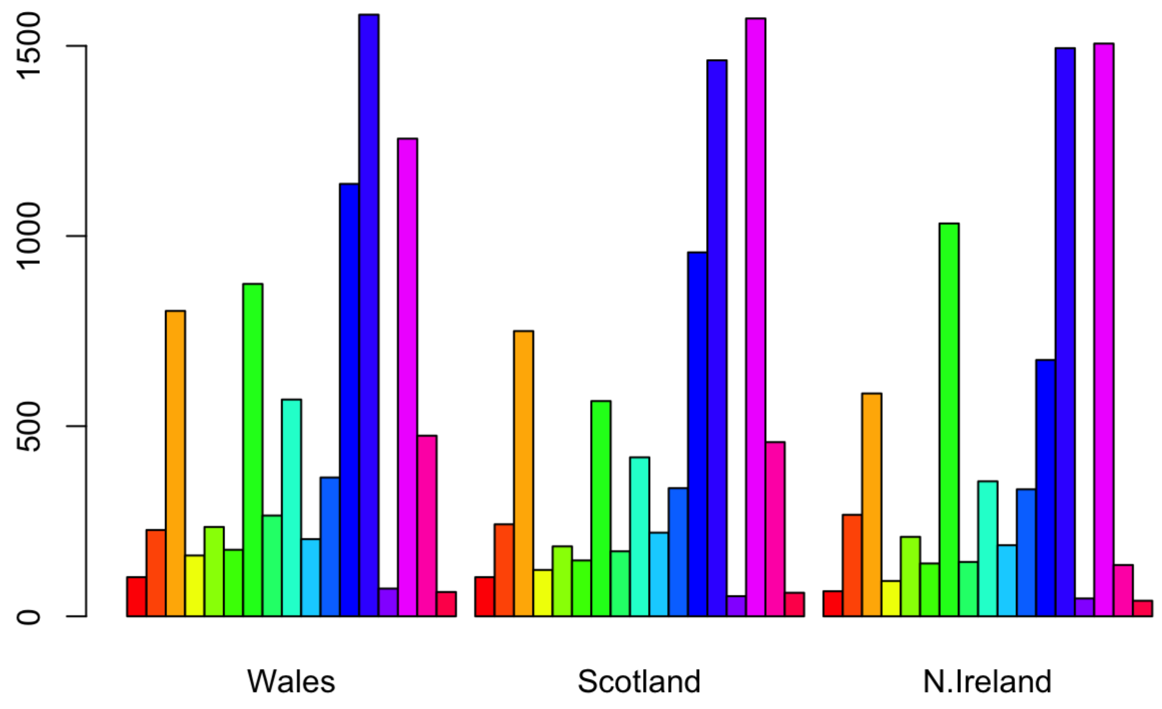
```
rownames(x) <- x[,1]
x <- x[,-1]
head(x)
```

```
    Wales Scotland N.Ireland
105   103      103        66
245   227      242       267
685   803      750       586
147   160      122        93
193   235      184       209
156   175      147       139
```

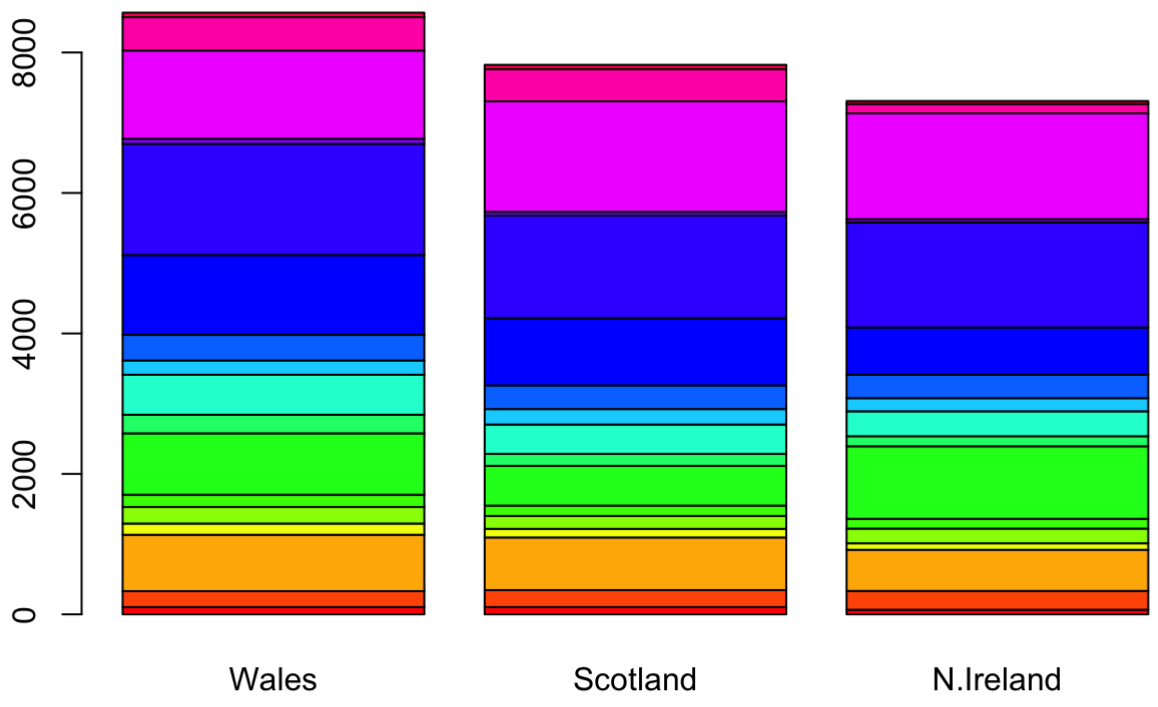
Q2. Which approach to solving the ‘row-names problem’ mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

I prefer to add a row.names argument when calling in the url so that the first row doesn’t continually get deleted every time that chunk of code is run.

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



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



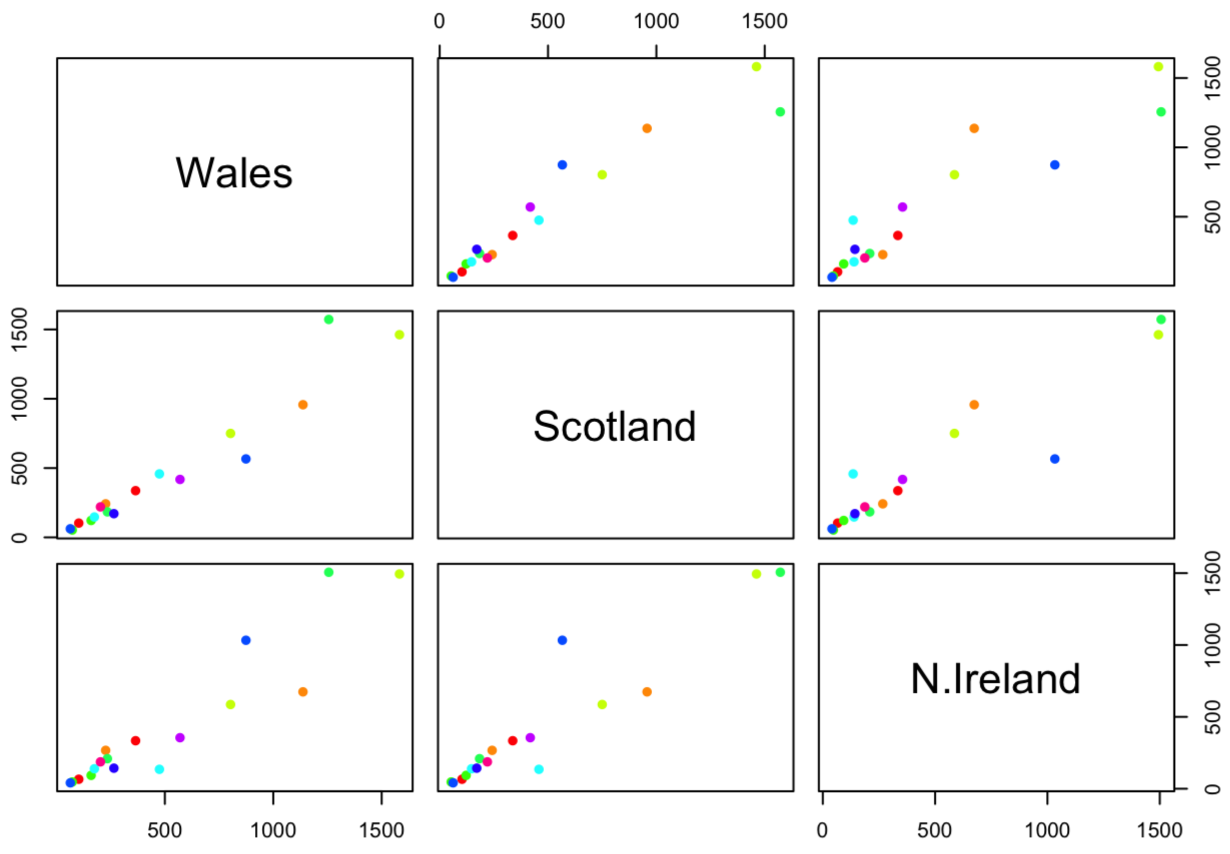
Q3: Changing what optional argument in the above barplot() function results in the following plot?

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?

If a point lies on the diagonal line I think that means it has the same value for both axis, so that value is the same or very similar for both countries represented in the plot. (the fold change is equal to 0)

```
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?

Whatever category the dark blue dot represents seems to stand out for Northern Ireland in comparison to the other countries.

##PCA to the rescue!

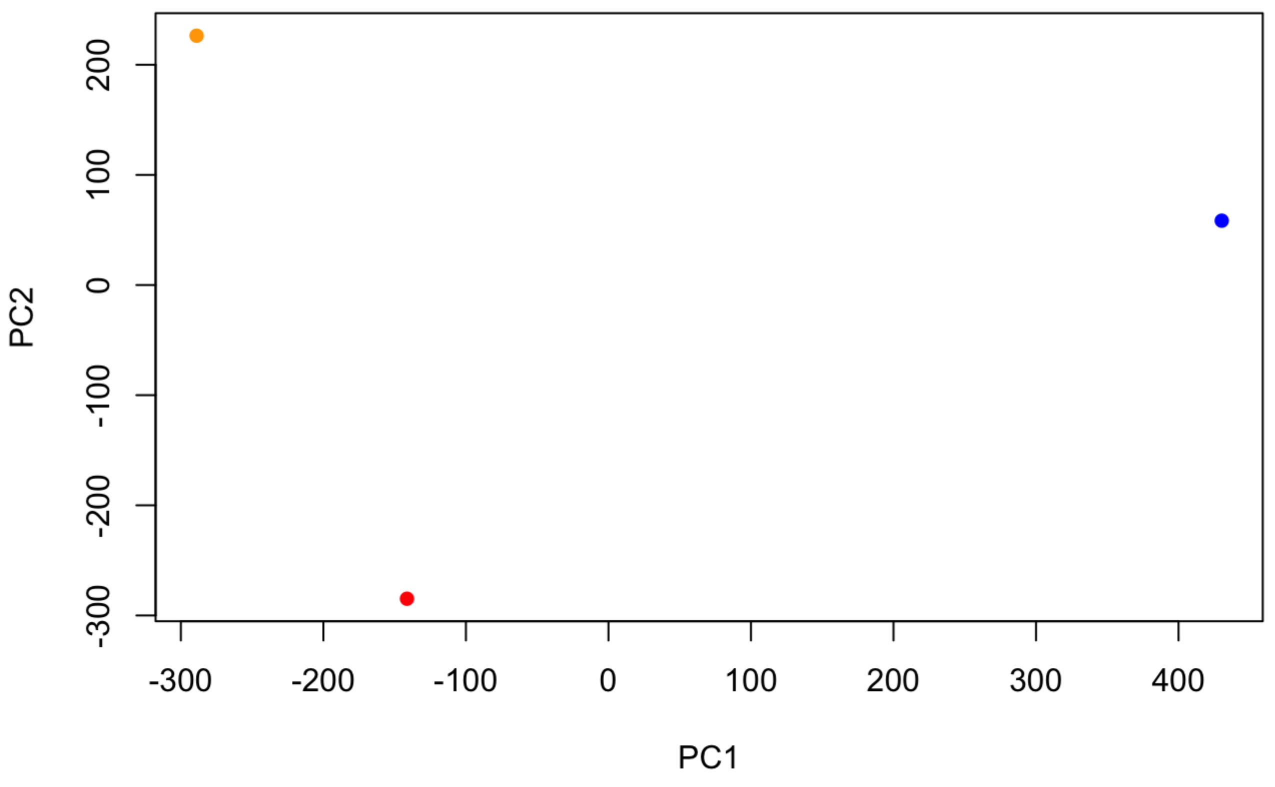
The main function in base R to do PCA is called `prcomp()`. One issue with the `prcomp()` functions is that it expects the transpose of our data as input.

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

```
Importance of components:
                            PC1      PC2       PC3
Standard deviation     379.8991 260.5533 1.515e-13
Proportion of Variance   0.6801   0.3199 0.000e+00
Cumulative Proportion    0.6801   1.0000 1.000e+00
```

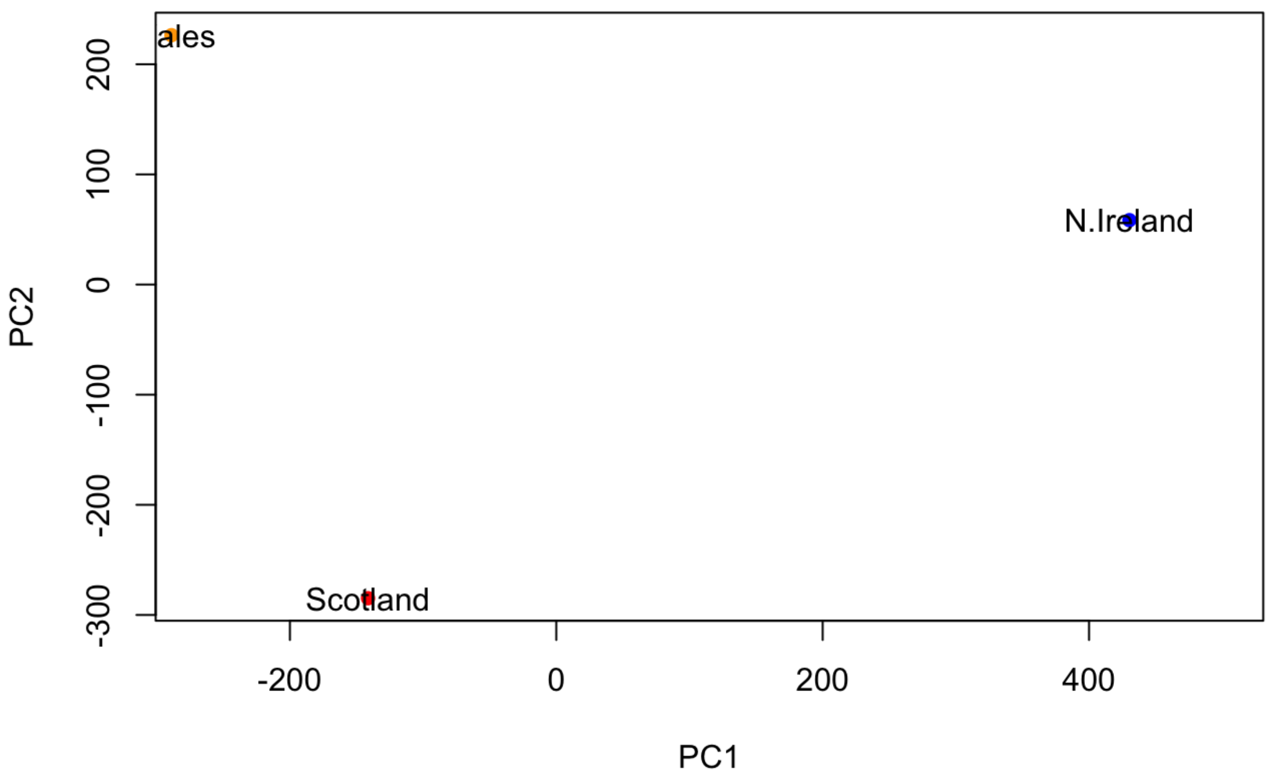
The object returned by `prcomp` has our results that include a $x component. This is our “scores” along the PCs (i.e. the plot of our data along the new PC axis)

```
plot(pca$x[,1], pca$x[,2],
  xlab="PC1", ylab="PC2",
  col=c("orange", "red", "blue", "darkgreen"),
  pch=16)
```



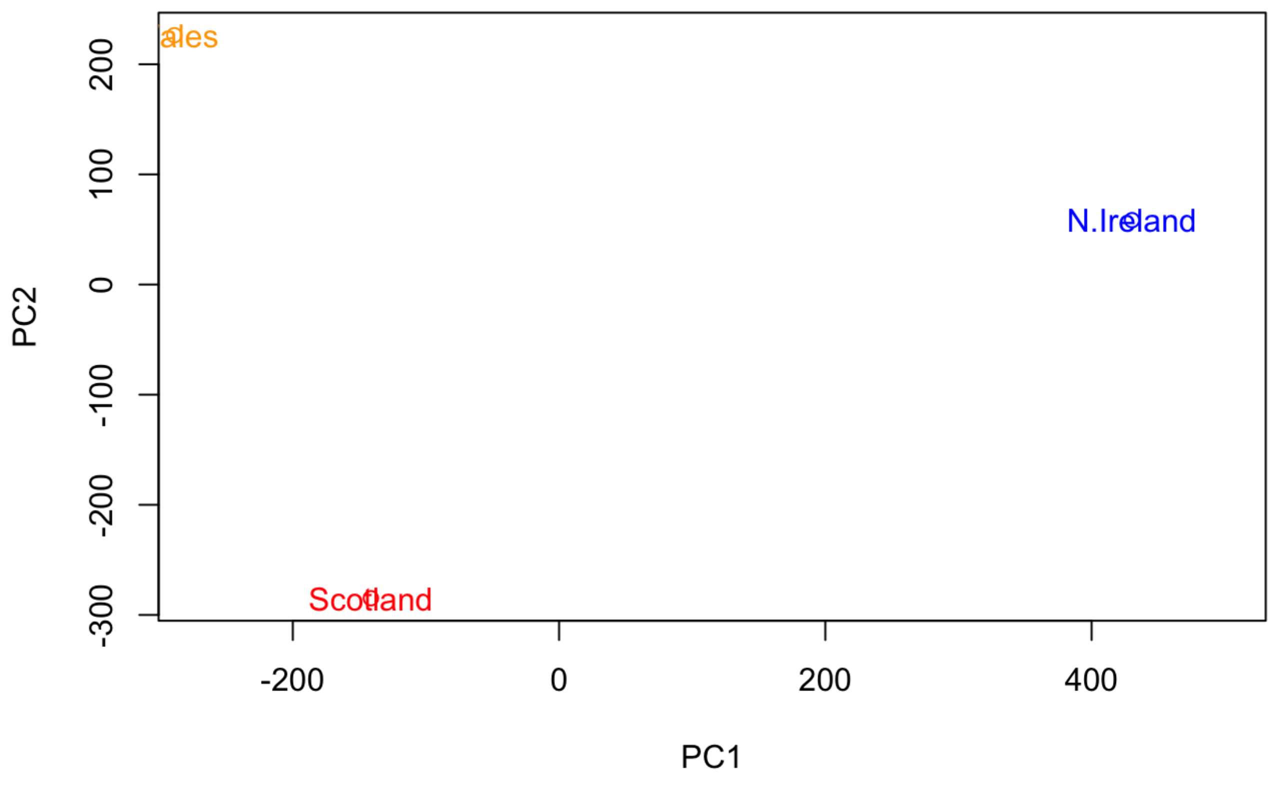
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
# Plot PC1 vs PC2
plot(pca$x[,1], pca$x[,2],
     xlab="PC1", ylab="PC2",
     xlim=c(-270,500), col=c("orange", "red", "blue", "darkgreen"), pch=16)
text(pca$x[,1], pca$x[,2], colnames(x))
```



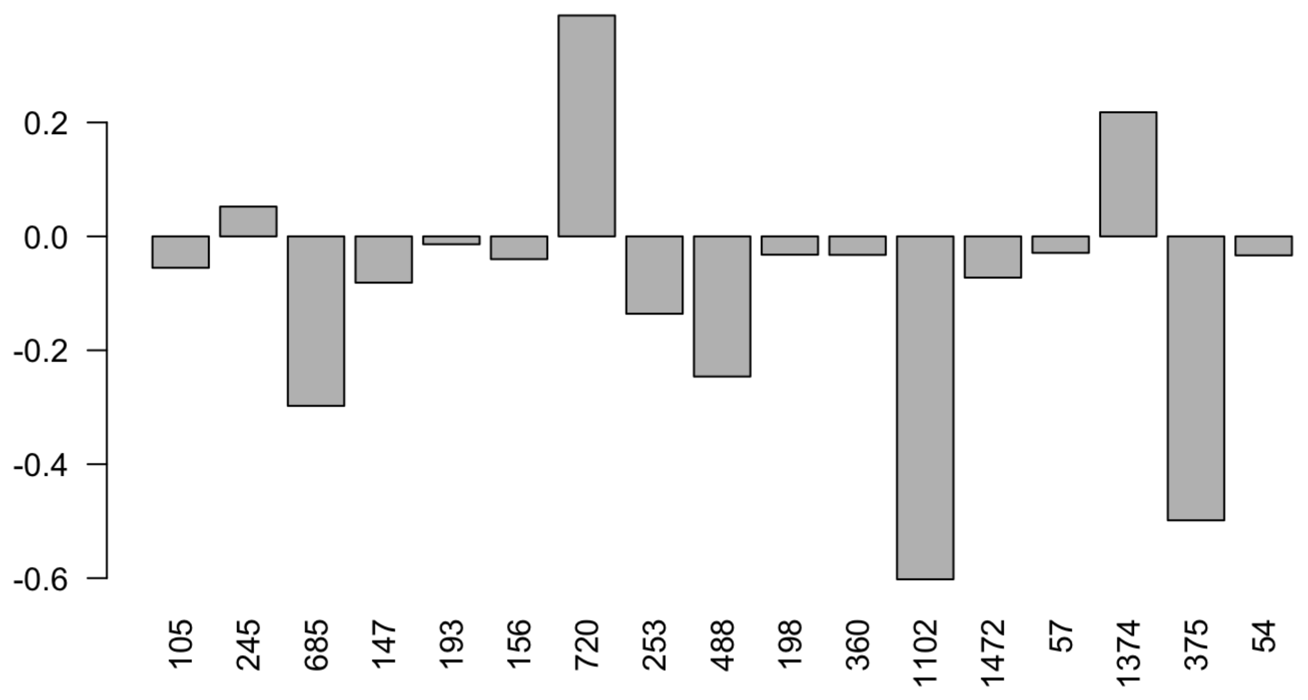
Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

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



Lets focus on PC1 as it accounts for > 90% of variance

```r
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



Northern Ireland consumes less fresh fruit, less alcoholic drinks, and more fresh potatoes than the other countries.