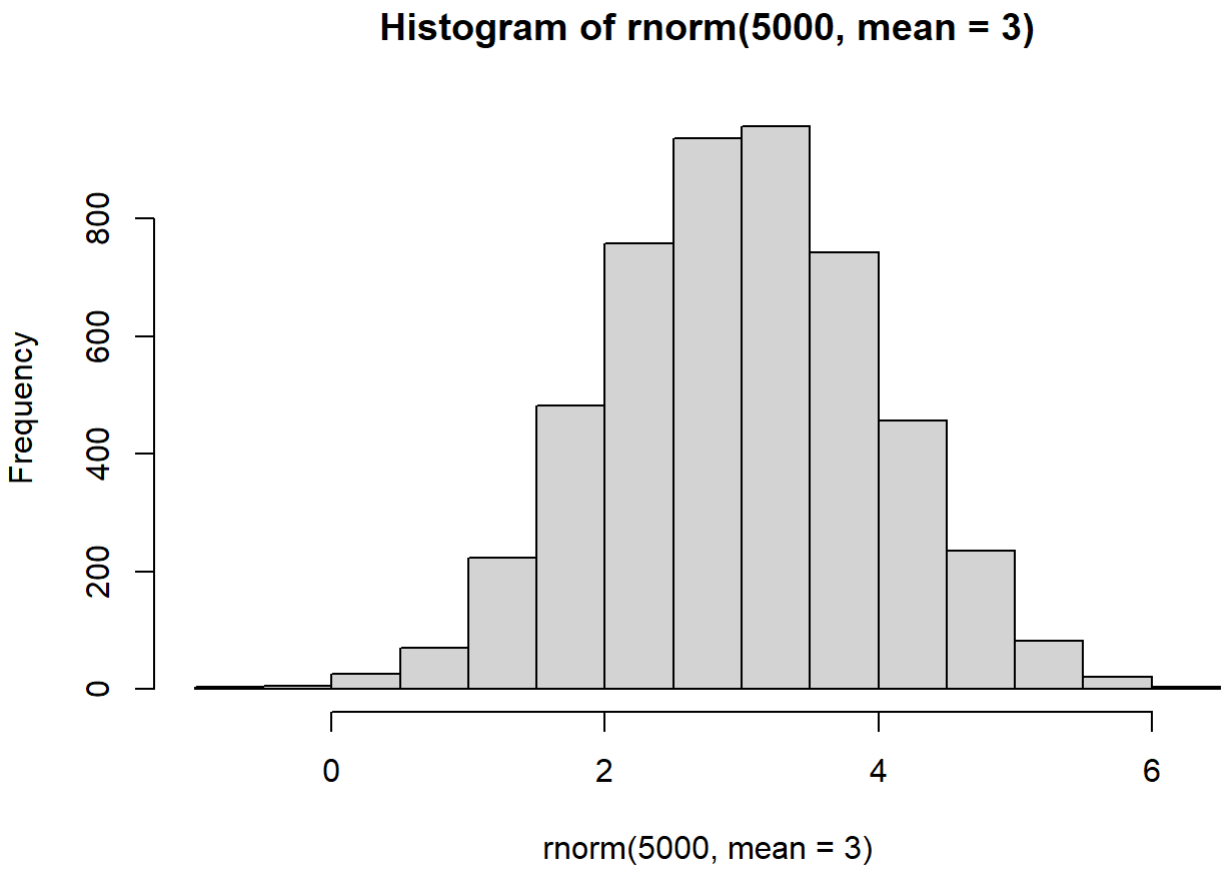
Class 7 : Clustering and PCA

AUTHOR
Ryan Chung PID A15848050

#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 generate random numbers from a normal distribution centered around `mean`

```
hist(rnorm(5000, mean = 3)) #histogram/data centered around mean value
```



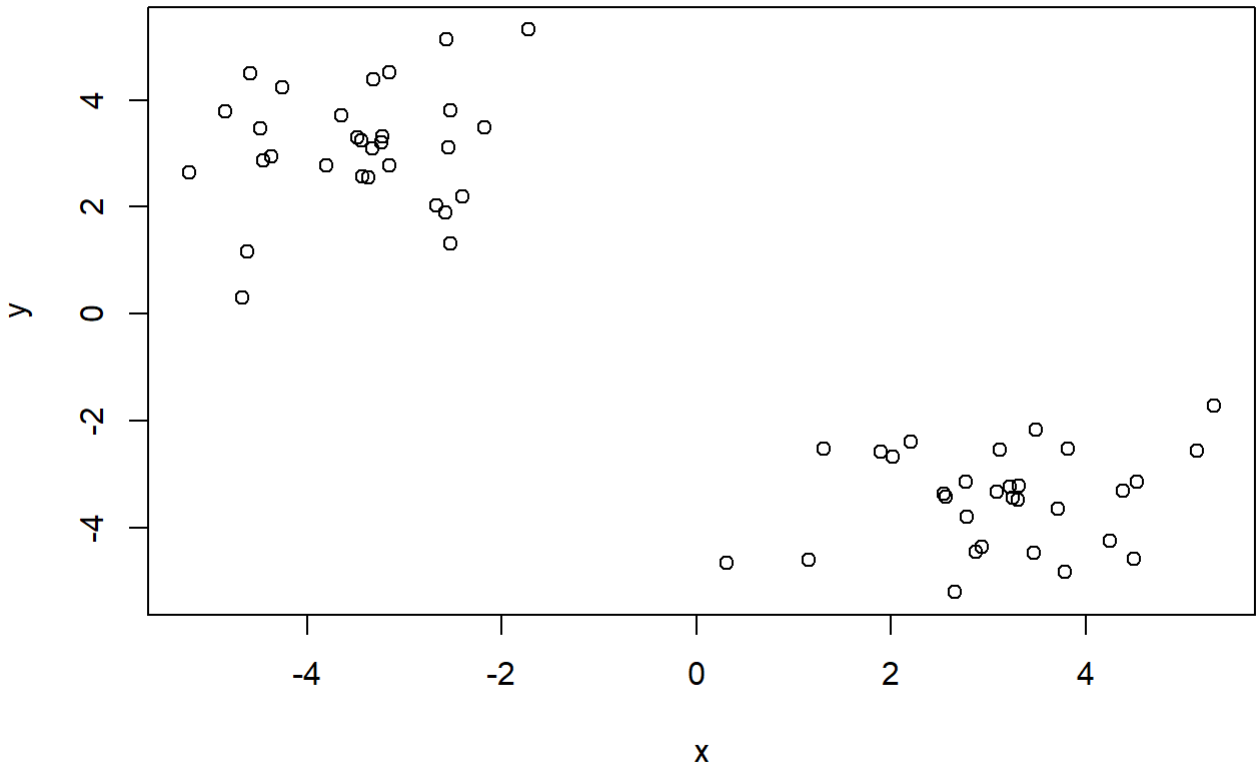
Lets get 30 points with a mean of 3

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

#cbind,rev functions
```

```
x <- cbind(x = test, y = rev(test))

rev( (c(1:5)))
```

```
[1] 5 4 3 2 1
```

```
plot(x)
```



##K-means clustering

Very popular clustering method that we can use with the `kmeans()` function in base R.

```
#anything with an equal sign is not necessary ?
#kmeans only needs an x, and centers arguement, centers - # of clusters or 'K'

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

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

Cluster means:
          x         y
1  3.120968 -3.459188
2 -3.459188  3.120968

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
```

```
Within cluster sum of squares by cluster:
[1] 60.55293 60.55293
 (between_SS / total_SS =  91.5 %)

Available components:

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

###kmean checkpoint questions >Q: Use the components to find out- Cluster size, cluster membership, and center

```
km$size #Returns you cluster sizes - 30 points per cluster
```

```
[1] 30 30
```

```
km$cluster #Returns cluster assignments
```

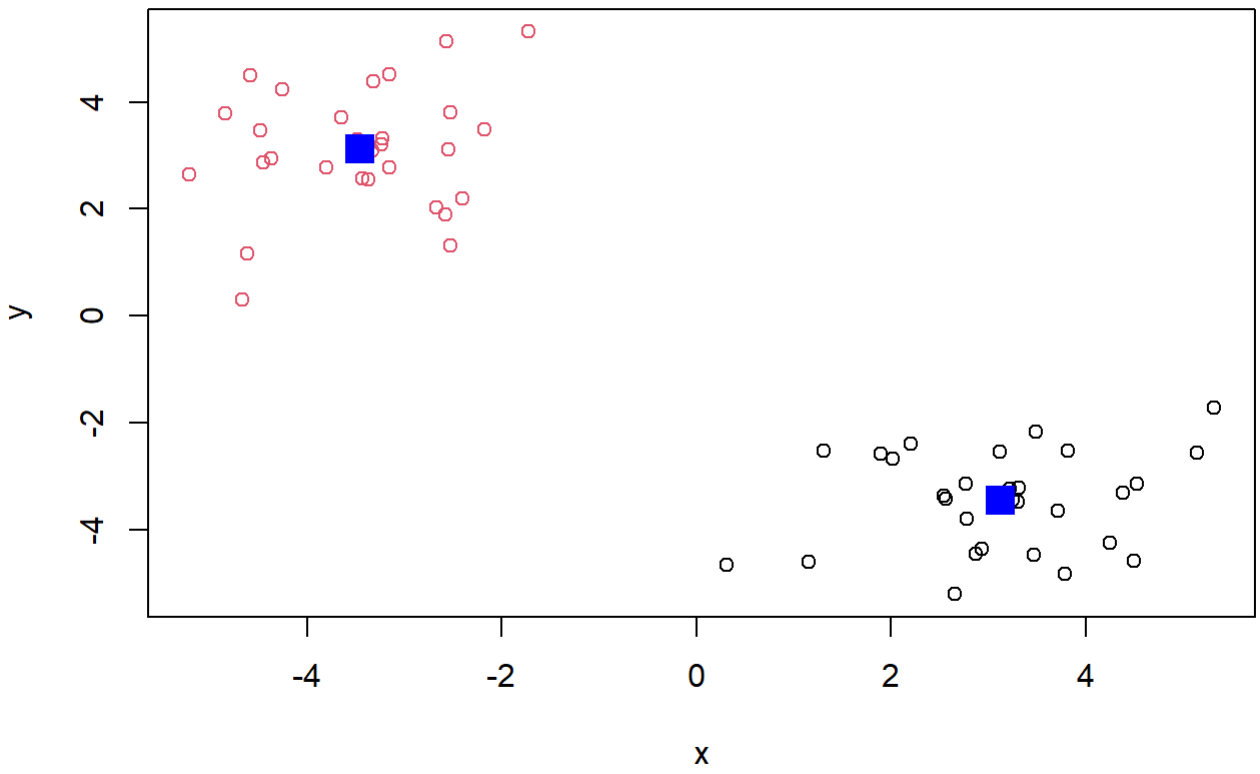
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
km$centers #Returns the coordinates of your centers
```

```
          x         y
1  3.120968 -3.459188
2 -3.459188  3.120968
```

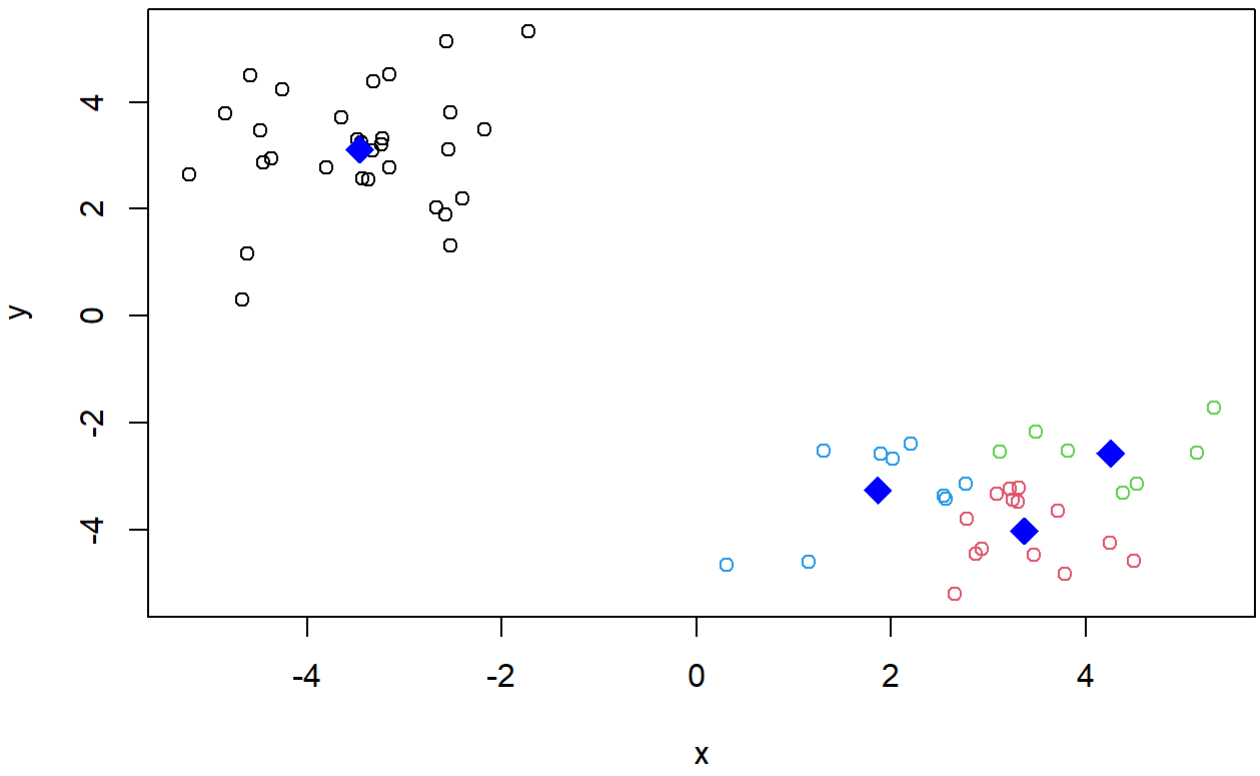
> Q: Plot x, colored by means cluster assignment, adding cluster centers as blue points

```
mycols <- c(km$cluster)
plot(x, col = mycols )
points(km$centers, col = "blue", pch = 15 , cex = 2)
```



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

```
km2 <- kmeans(x, centers = 4)
plot(x , col = km2$cluster)
points(km2$centers, col = "blue", pch = 18, cex = 2)
```




```
#totss is the measure of spread --> make elbow plots, where more centers usually drops totss
#scree plot = elbow plot, cause lower totss = better answer cause less spread (?)
```

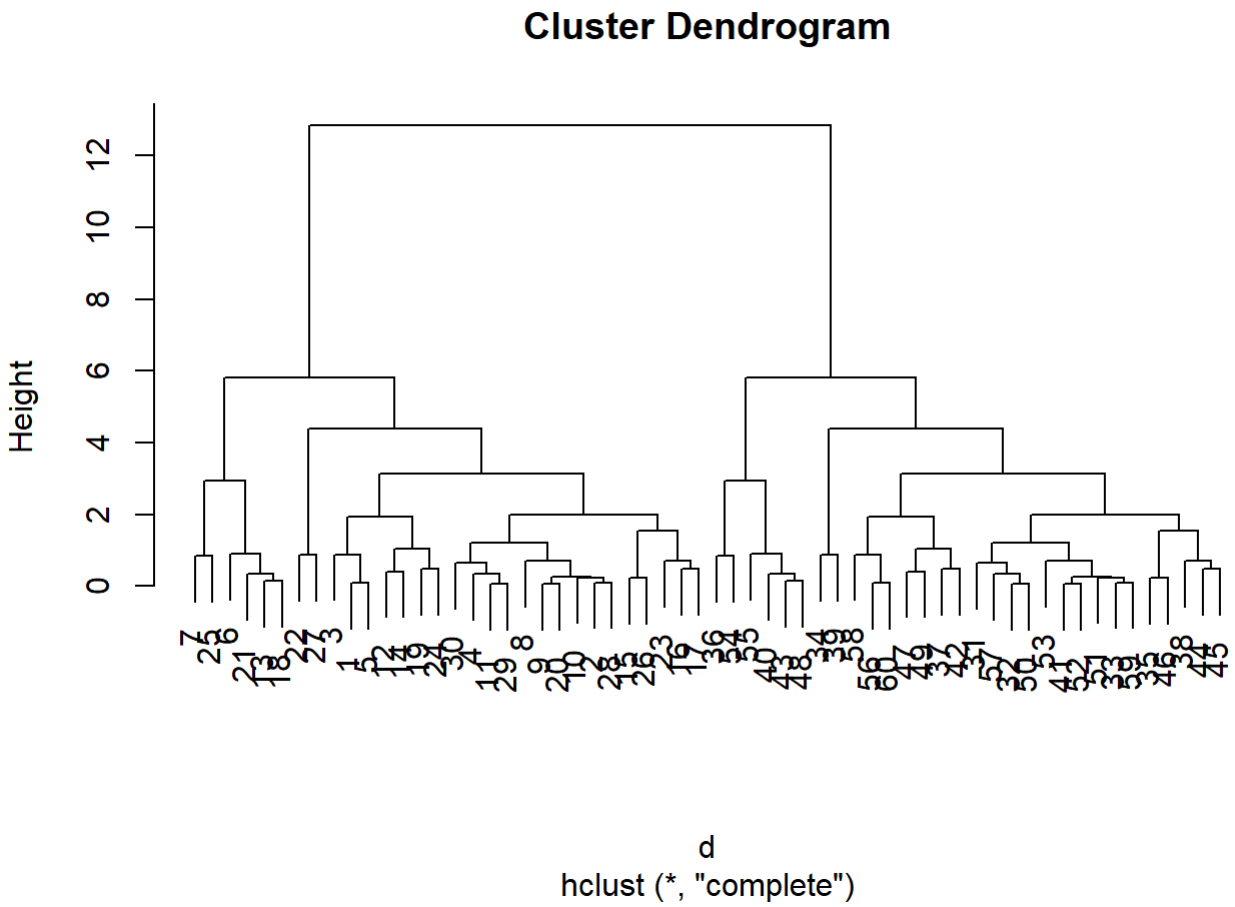
##Hierarchical Clustering

We can use the `hclust()` function for hierarchical clustering. But unlike `kmeans()`, where we could pass in our data as input, we need to give `hclust()` a "distance matrix" which is produced by `dist()`. The `dist()` function gives euclidean distance (normal distance we know for x y distances). 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) #h gives height of the cut
grps
```

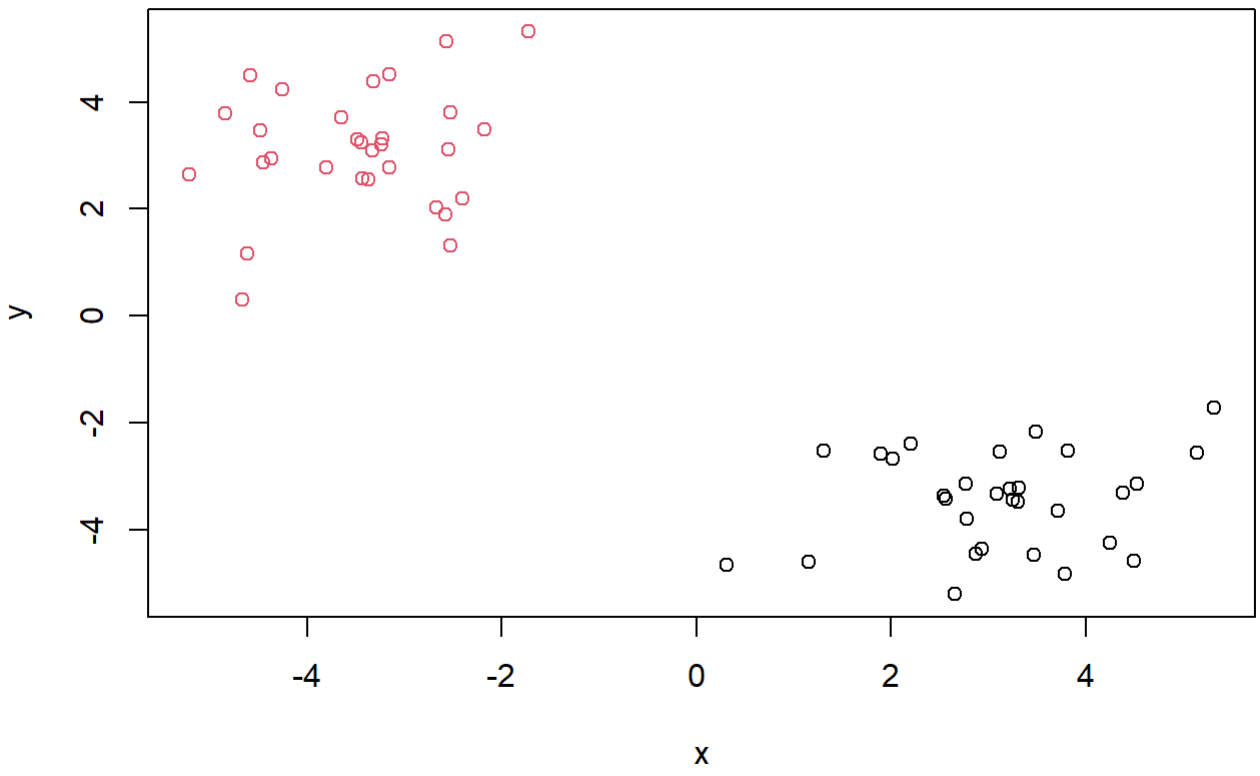
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

You can also tell `cutree()` to cut where it yields "k" number of 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 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"
y <- read.csv(url) #(url, row.names = 1)

str(y)
```

```
'data.frame':   17 obs. of  5 variables:
 $ X        : chr  "Cheese" "Carcass_meat " "Other_meat " "Fish" ...
 $ England  : int  105 245 685 147 193 156 720 253 488 198 ...
 $ Wales    : int  103 227 803 160 235 175 874 265 570 203 ...
 $ Scotland : int  103 242 750 122 184 147 566 171 418 220 ...
 $ N.Ireland: int  66 267 586 93 209 139 1033 143 355 187 ...
```

```
nrow(y)
```

```
[1] 17
```

```
ncol(y)
```

```
[1] 5
```

```
dim(y)
```

```
[1] 17  5
```

We have 17 rows and 5 columns

> Q preview the first 6 rows

```
head(y)
```

```
              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
```

###minus indexing

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

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

```
y <- y[ , -1] # this will remove the first column every single time you run this code (kinda bad)
head(y)
```

```
             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
```

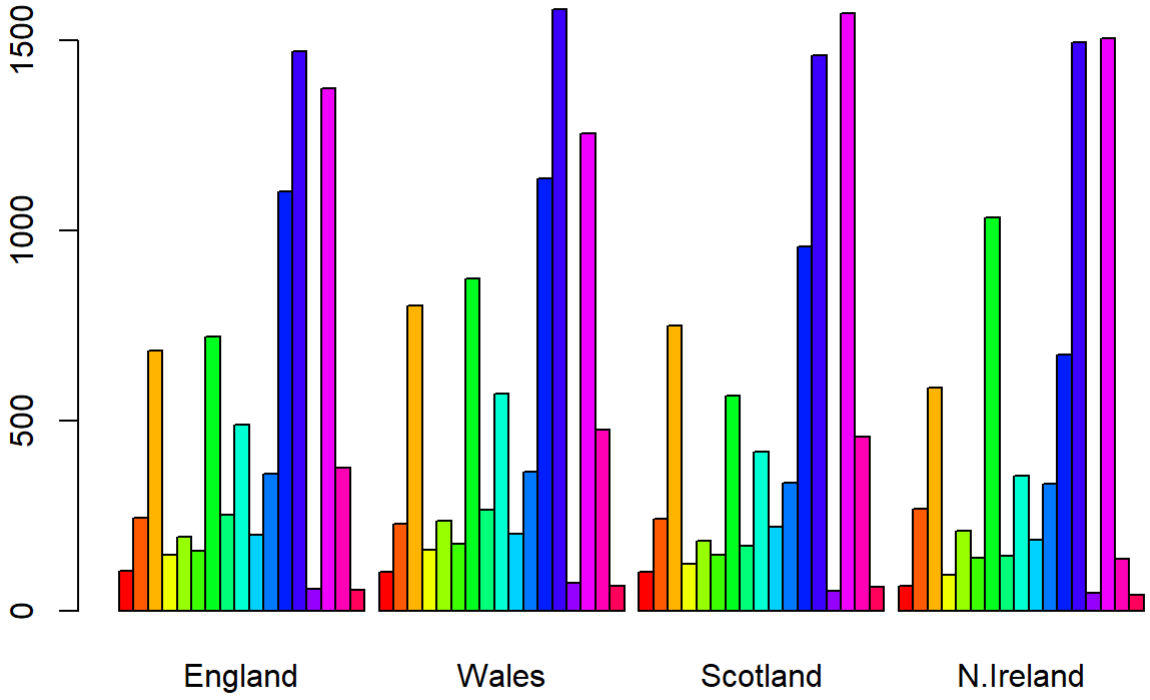
```
dim(y)
```

```
[1] 17  4
```

> 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?

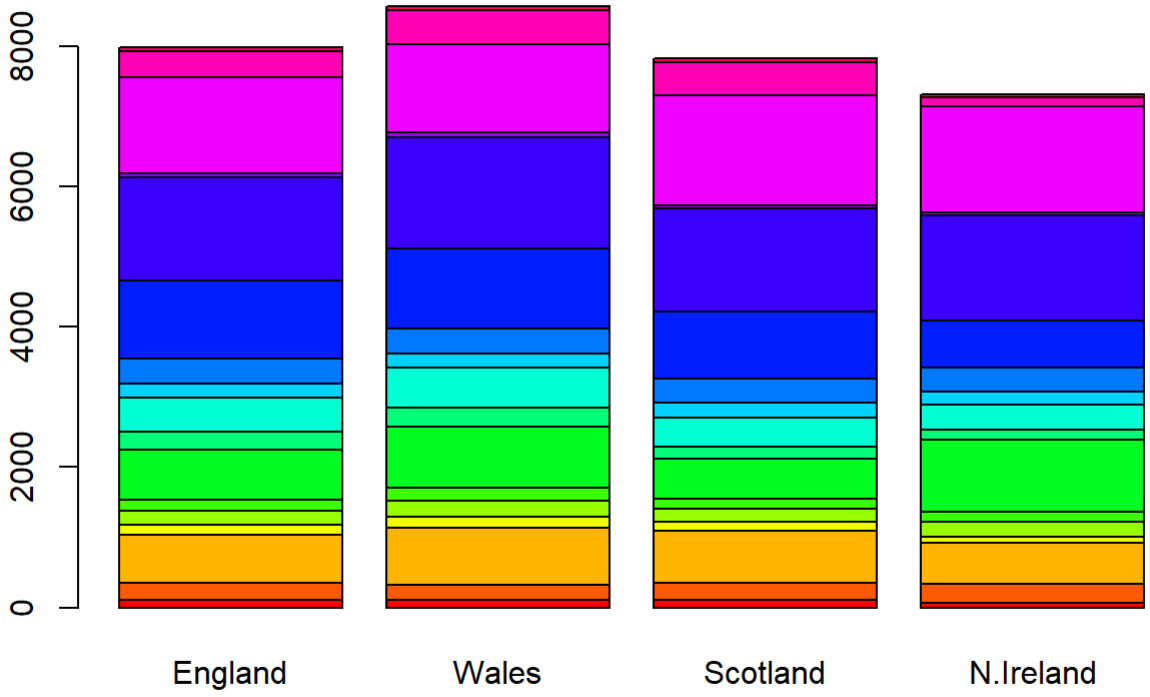
I prefer the `row.names()` approach becauses I am liable to change and rerun my code many different times and I would likely end up with an empty dataframe.

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



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

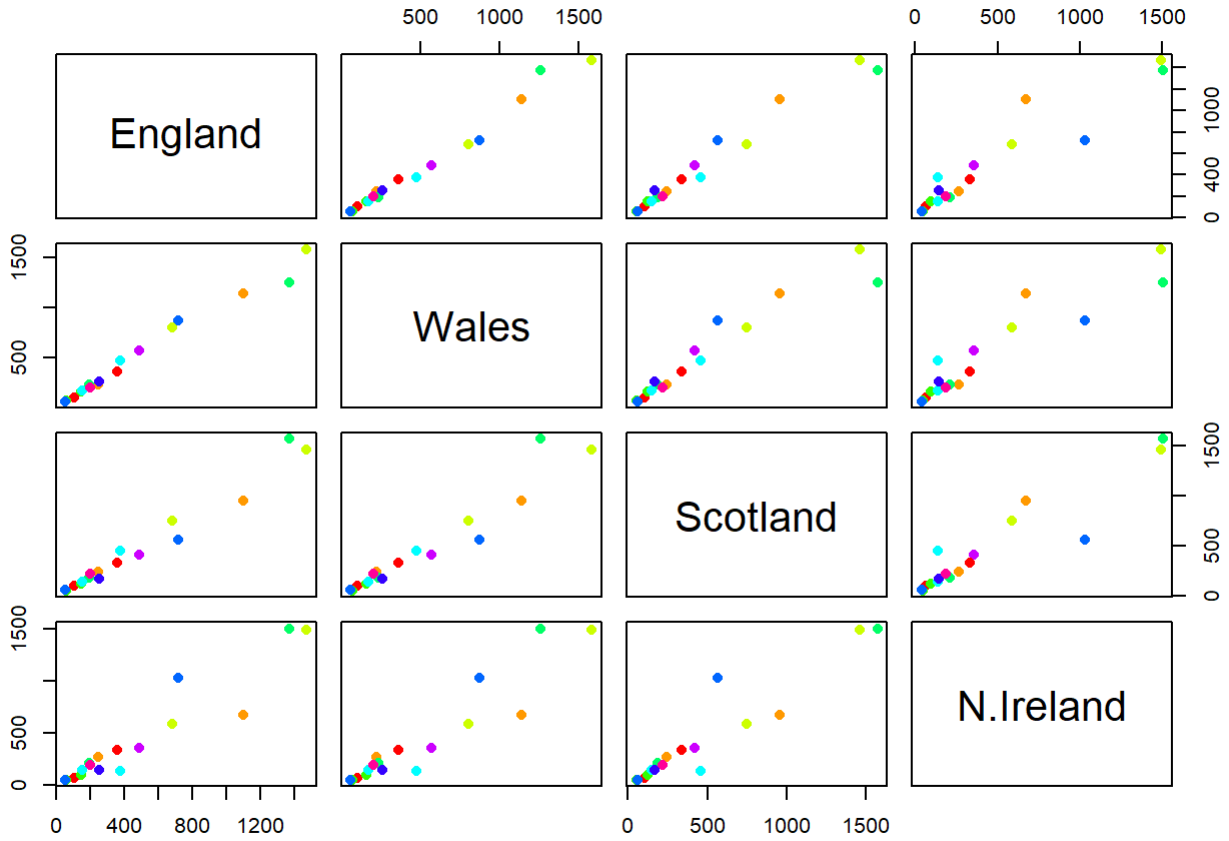
```
barplot(as.matrix(y), beside=F, col=rainbow(nrow(y)))
```



Changing the beside arguement to False gives us the vertical plot.

> 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(y, col = rainbow(10), pch = 16)
```




```
#reading: L-R = england is y axis, up-down = x axis if the two compared are the same, then they w
```

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

The blue, orange, and cyan categories are the main differences between N. Ireland and the other countries just from a visual standpoint.

###using the `prcomp()` function

```
#`t()` gives you the transposed stuff (?) idk wat do
pca <-prcomp(t(y))
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
```

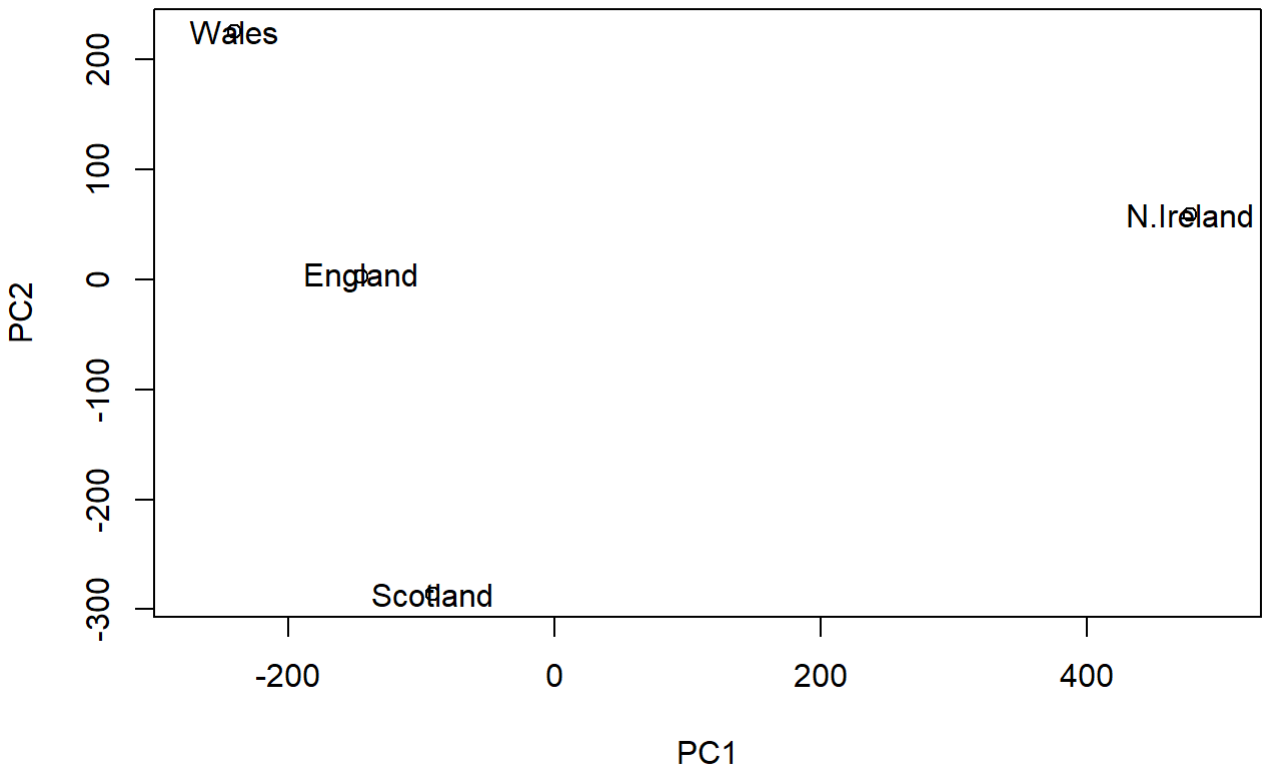
```
attributes(pca)
```

```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

pca$x

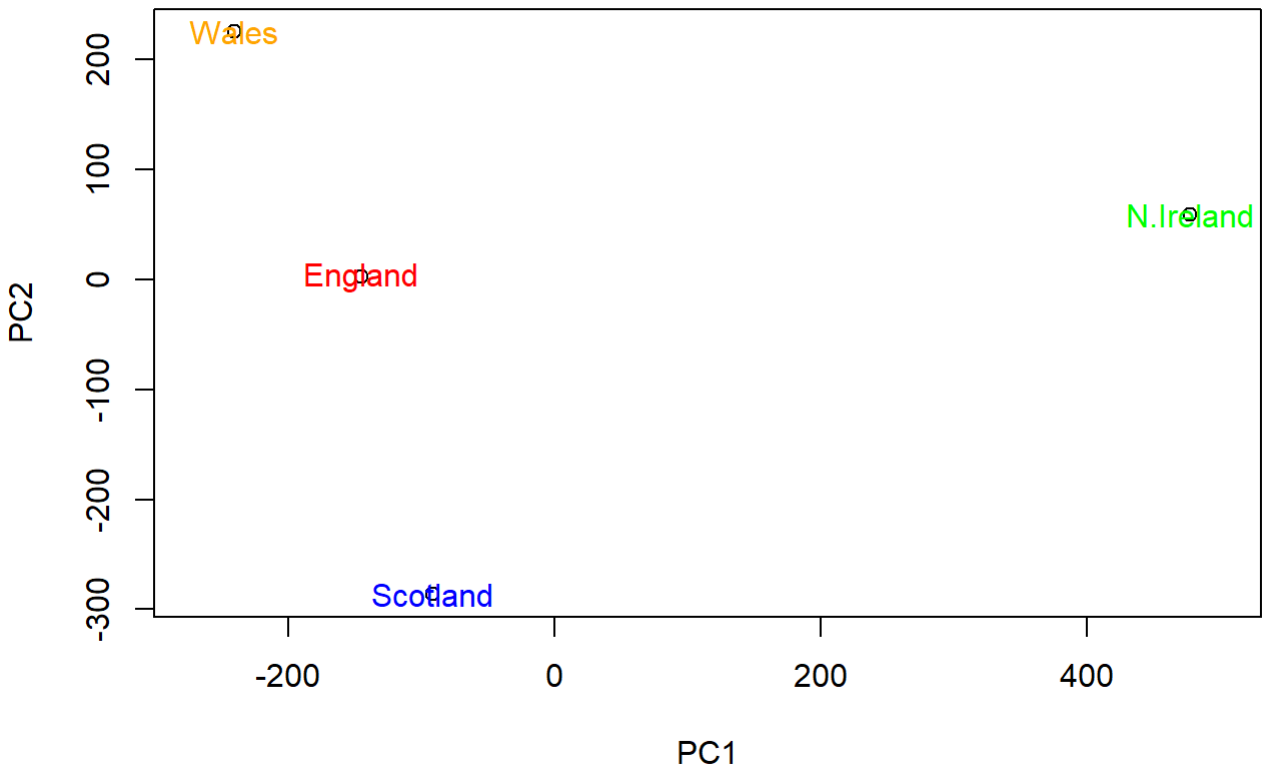
> Q7: Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

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



> 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))
text(pca$x[ ,1], pca$x[ ,2], colnames(y), col = c("red", "orange", "blue", "green"))
```



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