Class 7

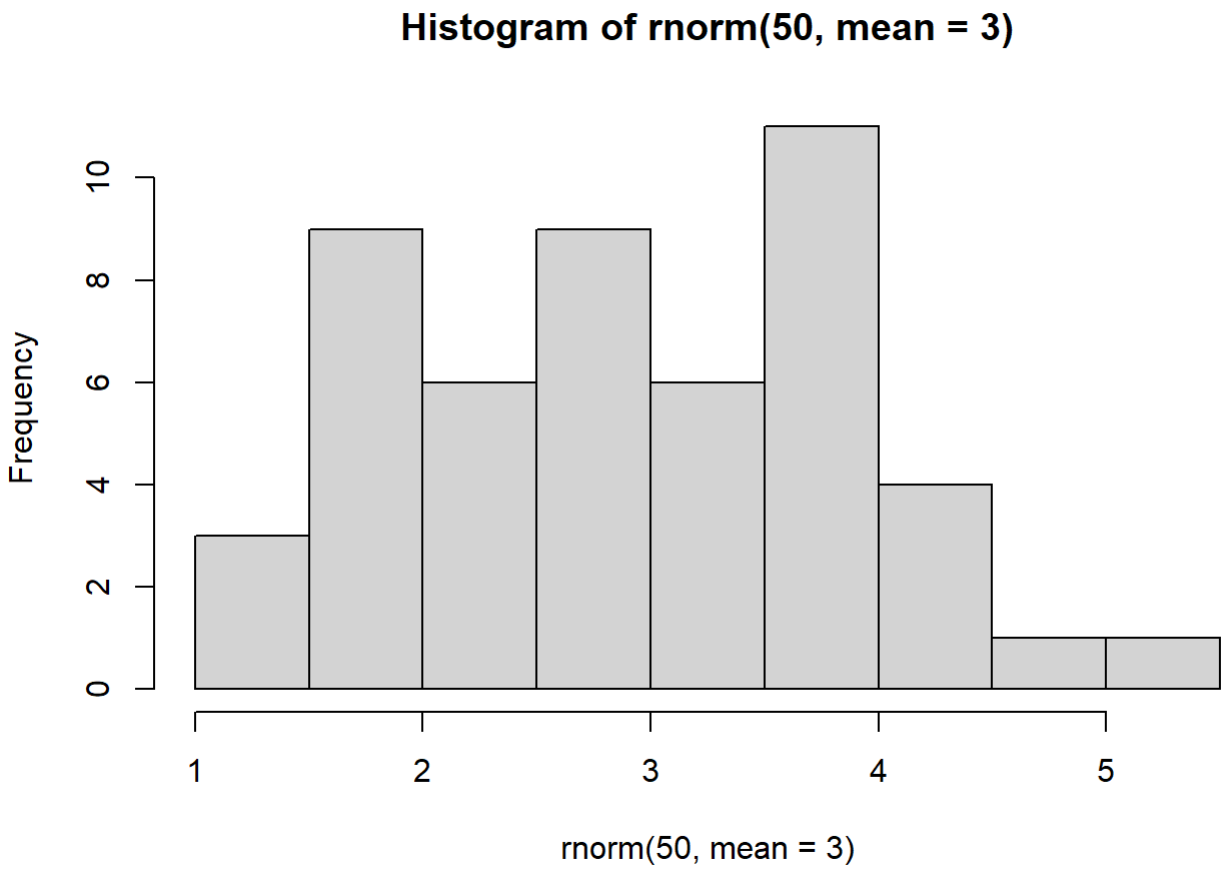
AUTHOR
Charlie Rezanka (A15837296)

#Clustering

First, lets make some data to cluster so we an 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(50, mean=3))
```



Lets get 30 points with a mean of 3.

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

```
 [1]  3.9570370  3.0952452  2.9970242  2.8996540  2.1386816  3.2298037
 [7]  2.0933567  3.1570108  2.6626312  3.5610050  3.3682637  4.8453481
[13]  2.4223204  3.8455682  2.2873779  3.0880848  0.8539646  4.0301493
[19]  3.8274040  4.4434410  3.4020797  4.1437456  2.8362601  2.7247062
```

```
[25]  2.7038369  4.0261389  5.2198345  2.6621672  4.2448935  2.4176223
[31] -3.2896442 -3.4754862 -2.5205180 -2.8005186 -4.0421330 -2.2807507
[37] -3.0638581 -3.3883475 -2.1133498 -2.7621615 -3.4294349 -3.2648739
[43] -2.5975358 -2.8229022 -3.1605515 -2.6430003 -3.2073887 -3.9837174
[49] -2.5177699 -3.2855633 -2.8254680 -4.0132678 -2.5885968 -2.3400685
[55] -2.6285954 -3.6737060 -3.0679519 -2.9590448 -3.4484579 -1.6685597
```

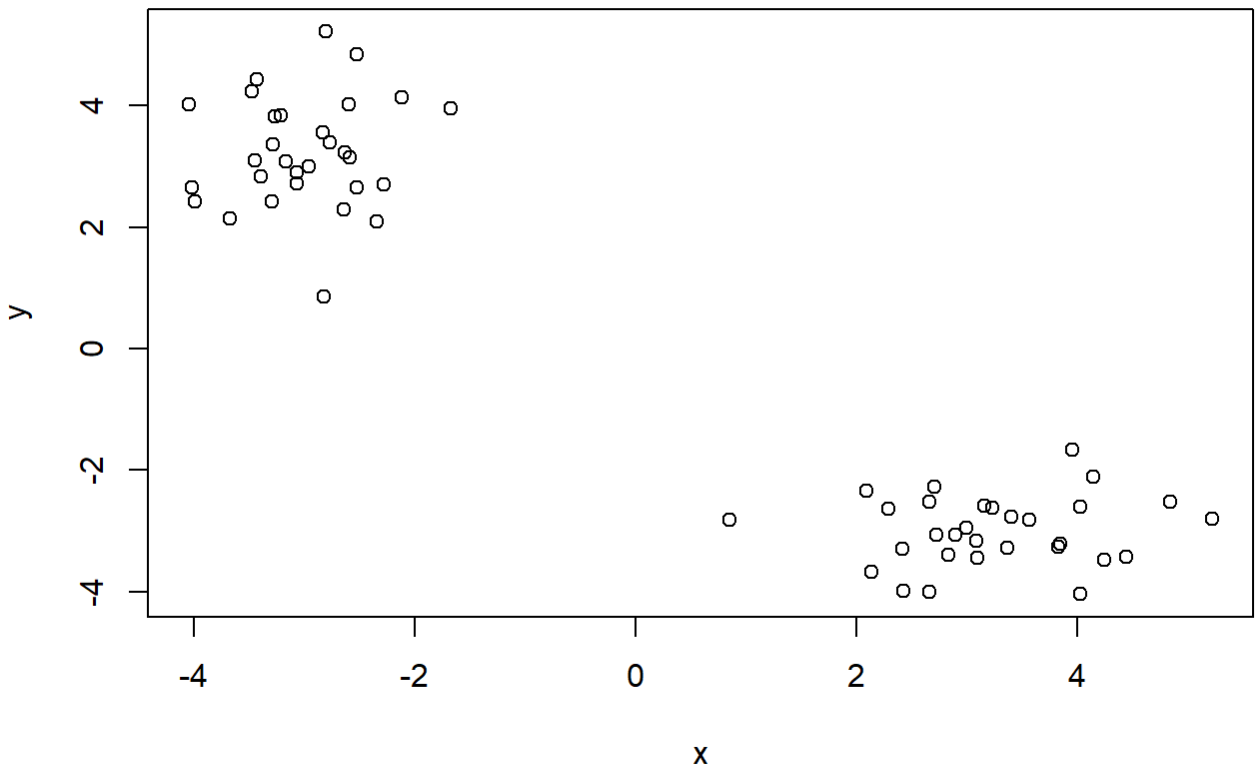
Lets put these two together:

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

```
              x          y
 [1,]  3.9570370 -1.6685597
 [2,]  3.0952452 -3.4484579
 [3,]  2.9970242 -2.9590448
 [4,]  2.8996540 -3.0679519
 [5,]  2.1386816 -3.6737060
 [6,]  3.2298037 -2.6285954
 [7,]  2.0933567 -2.3400685
 [8,]  3.1570108 -2.5885968
 [9,]  2.6626312 -4.0132678
[10,]  3.5610050 -2.8254680
[11,]  3.3682637 -3.2855633
[12,]  4.8453481 -2.5177699
[13,]  2.4223204 -3.9837174
[14,]  3.8455682 -3.2073887
[15,]  2.2873779 -2.6430003
[16,]  3.0880848 -3.1605515
[17,]  0.8539646 -2.8229022
[18,]  4.0301493 -2.5975358
[19,]  3.8274040 -3.2648739
[20,]  4.4434410 -3.4294349
[21,]  3.4020797 -2.7621615
[22,]  4.1437456 -2.1133498
[23,]  2.8362601 -3.3883475
[24,]  2.7247062 -3.0638581
[25,]  2.7038369 -2.2807507
[26,]  4.0261389 -4.0421330
[27,]  5.2198345 -2.8005186
[28,]  2.6621672 -2.5205180
[29,]  4.2448935 -3.4754862
[30,]  2.4176223 -3.2896442
[31,] -3.2896442  2.4176223
[32,] -3.4754862  4.2448935
[33,] -2.5205180  2.6621672
[34,] -2.8005186  5.2198345
[35,] -4.0421330  4.0261389
[36,] -2.2807507  2.7038369
[37,] -3.0638581  2.7247062
[38,] -3.3883475  2.8362601
```

```
[39,] -2.1133498  4.1437456
[40,] -2.7621615  3.4020797
[41,] -3.4294349  4.4434410
[42,] -3.2648739  3.8274040
[43,] -2.5975358  4.0301493
[44,] -2.8229022  0.8539646
[45,] -3.1605515  3.0880848
[46,] -2.6430003  2.2873779
[47,] -3.2073887  3.8455682
[48,] -3.9837174  2.4223204
[49,] -2.5177699  4.8453481
[50,] -3.2855633  3.3682637
[51,] -2.8254680  3.5610050
[52,] -4.0132678  2.6626312
[53,] -2.5885968  3.1570108
[54,] -2.3400685  2.0933567
[55,] -2.6285954  3.2298037
[56,] -3.6737060  2.1386816
[57,] -3.0679519  2.8996540
[58,] -2.9590448  2.9970242
[59,] -3.4484579  3.0952452
[60,] -1.6685597  3.9570370
```

```
plot(x)
```



##K-means Clustering

This is a very popular clustering method, especially for large datasets, that can be used with `kmeans()`

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

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

Cluster means:
          x         y
1  3.239489 -2.995441
2 -2.995441  3.239489

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 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] 33.80799 33.80799
 (between_SS / total_SS =  94.5 %)

Available components:

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

> ""Q"" How many points are in each cluster??

```
30 points in each cluster
```

> ""Q"" What component of your result corresponds to cluster size (A), cluster assignment (B), and cluster center (C)

```
A  km$size
B  km$cluster
C  km$centers
```

```
km$size
```

```
[1] 30 30
```

```
km$cluster
```

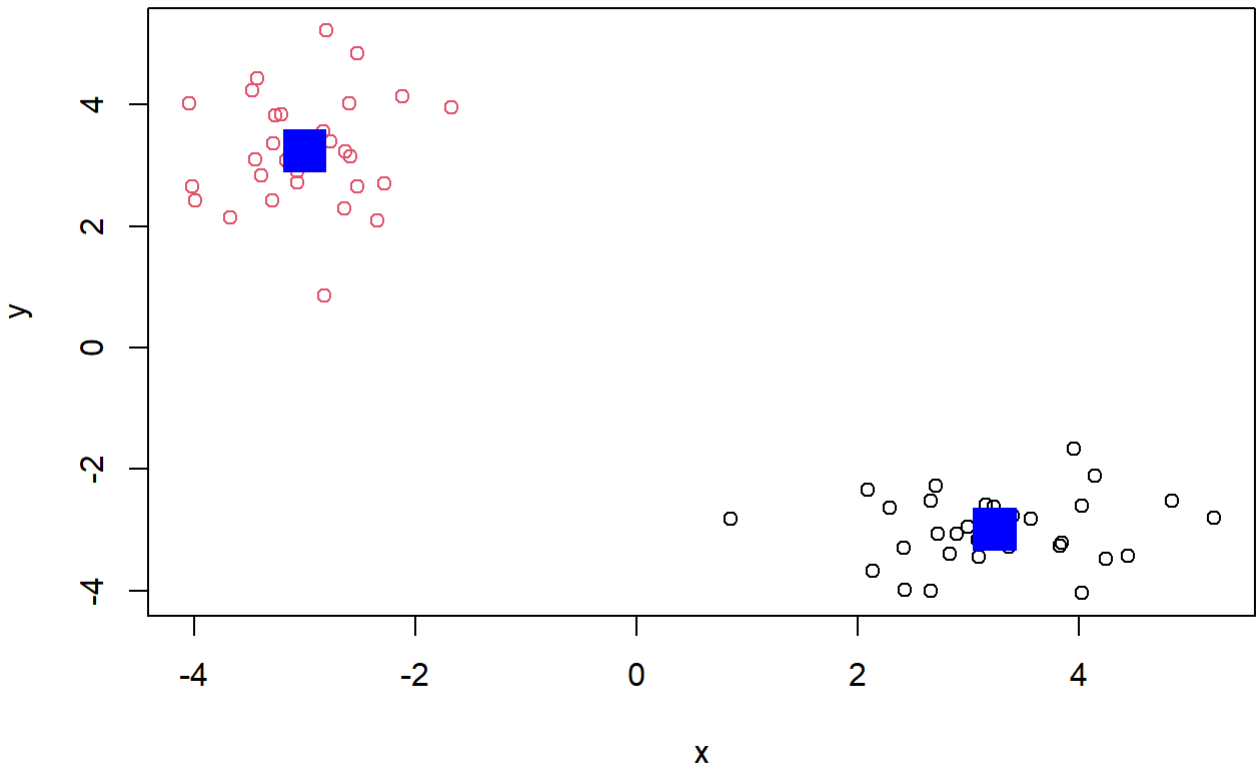
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
km$centers
```

```
          x         y
1  3.239489 -2.995441
2 -2.995441  3.239489
```

> ""Q"" Plot x colored by the kmeans cluster assignment and add cluster centers as blue points

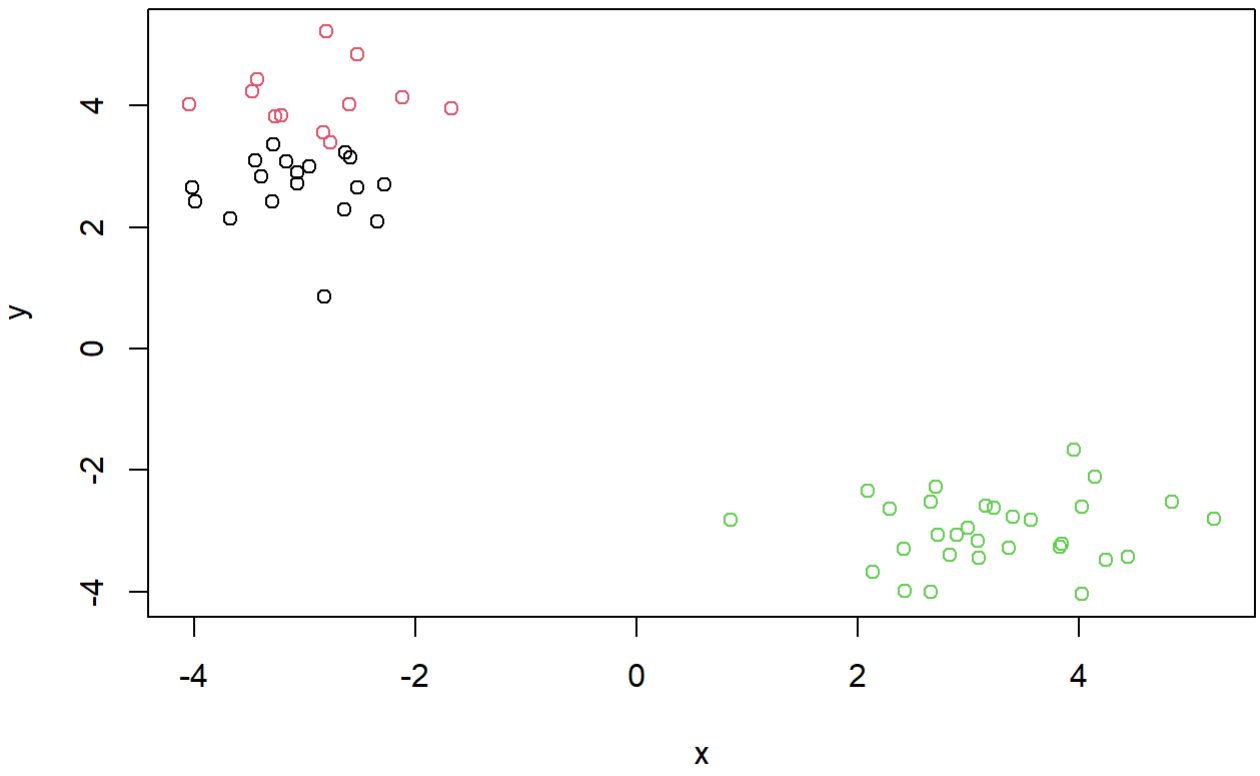
```
plot(x, col=km$cluster)
points(km$centers, col="blue", pch=15, cex=3)
```




```
##pch assignes the centers a shape similar to colors
#cex determines the size of the shape
```

> ""Q"" Lets cluster into 3 groups or some `x` data to 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 can just pass in our data as input, we need to give `hclust()` a "distance matrix".

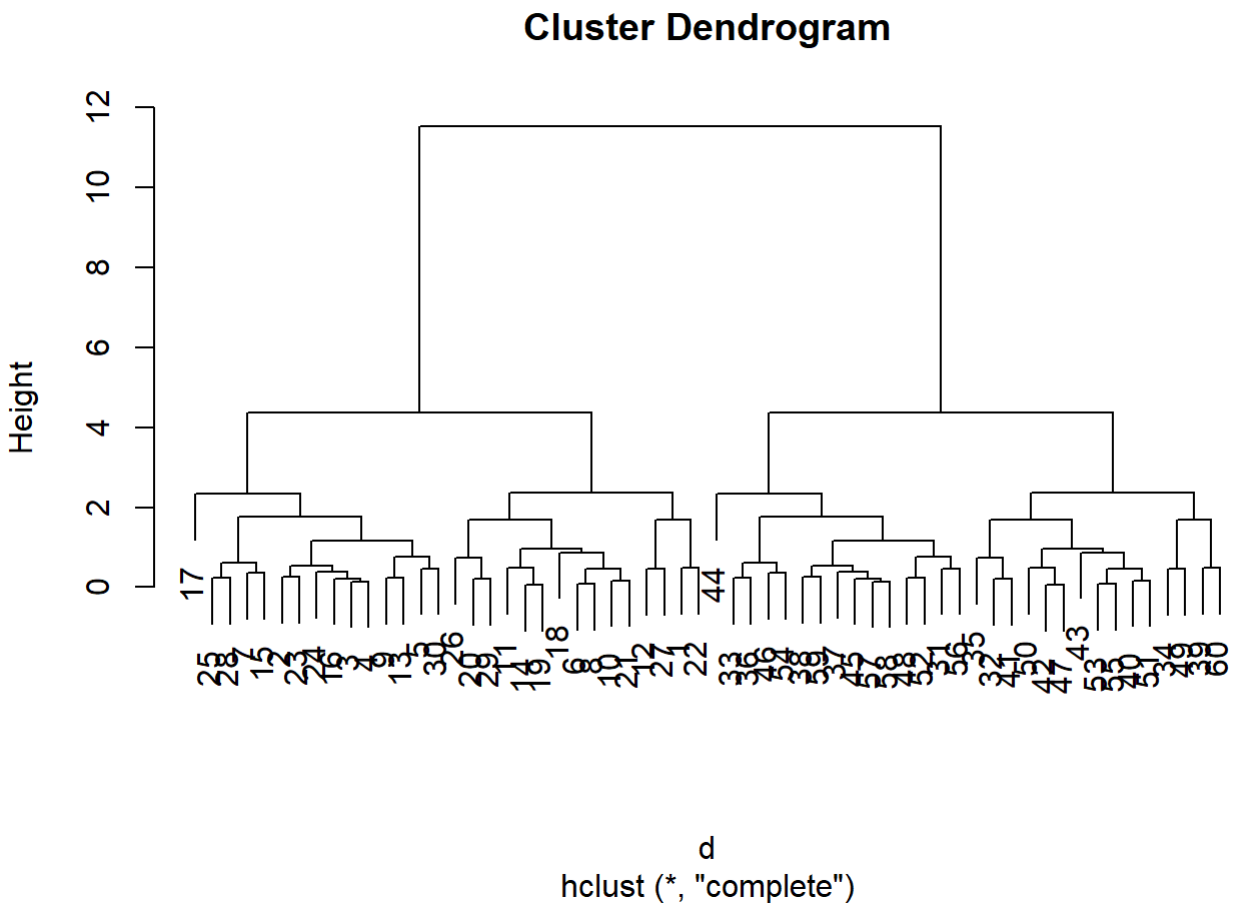
We will use the `dist()` function to start with.

```
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 `cutree()` to yield a cluster membership vector

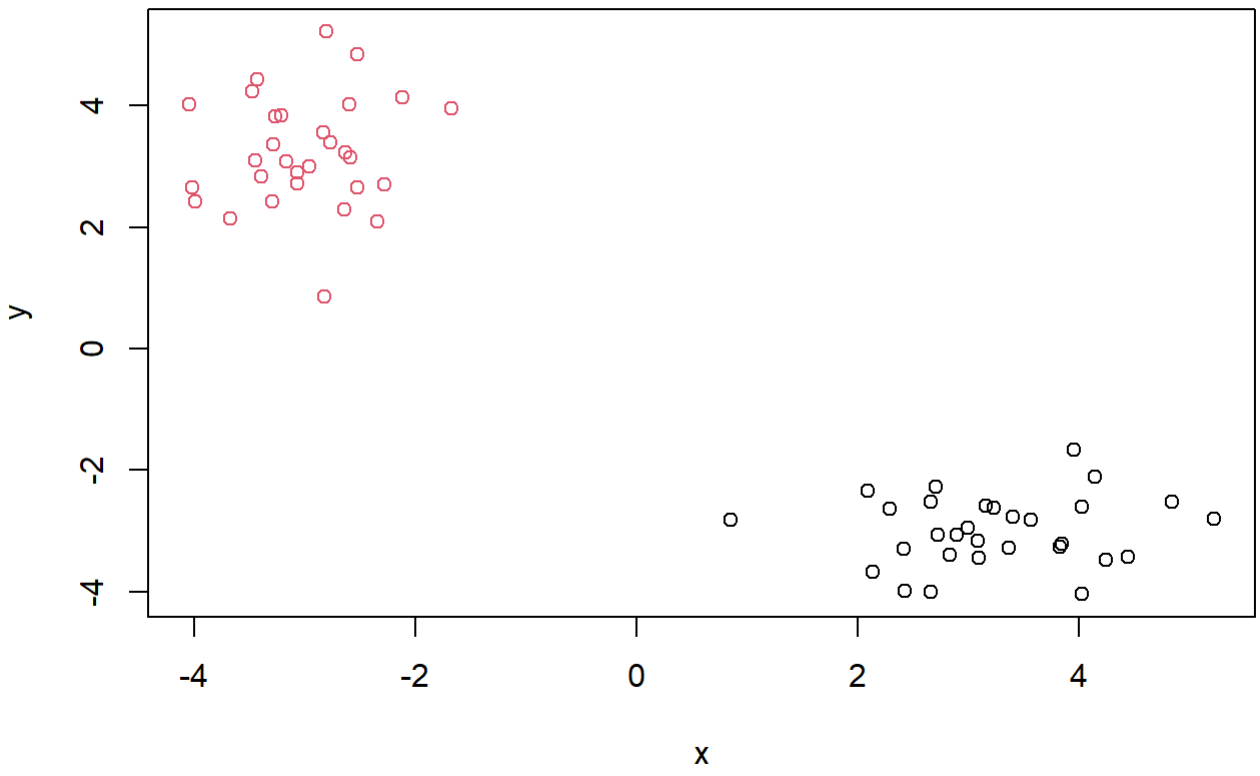
```
grps <- cutree(hc, h=8)
```

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
```

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



#Principal Component Analysis, PCA Worksheet

Data will first be plotted along a primary axis (PC1) to show variability. Subsequent criteria for variability are added as additional axis and the variance of points for that criteria is shown as distribution.

```
x<- read.csv("https://tinyurl.com/UK-foods", 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 this dataset? What functions can you use to answer this question?

17 rows, 4 columns. nrow and ncol can be used respectively. dim() also works.

```
nrow(x)
```

```
[1] 17
```

```
ncol(x)
```

```
[1] 4
```

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

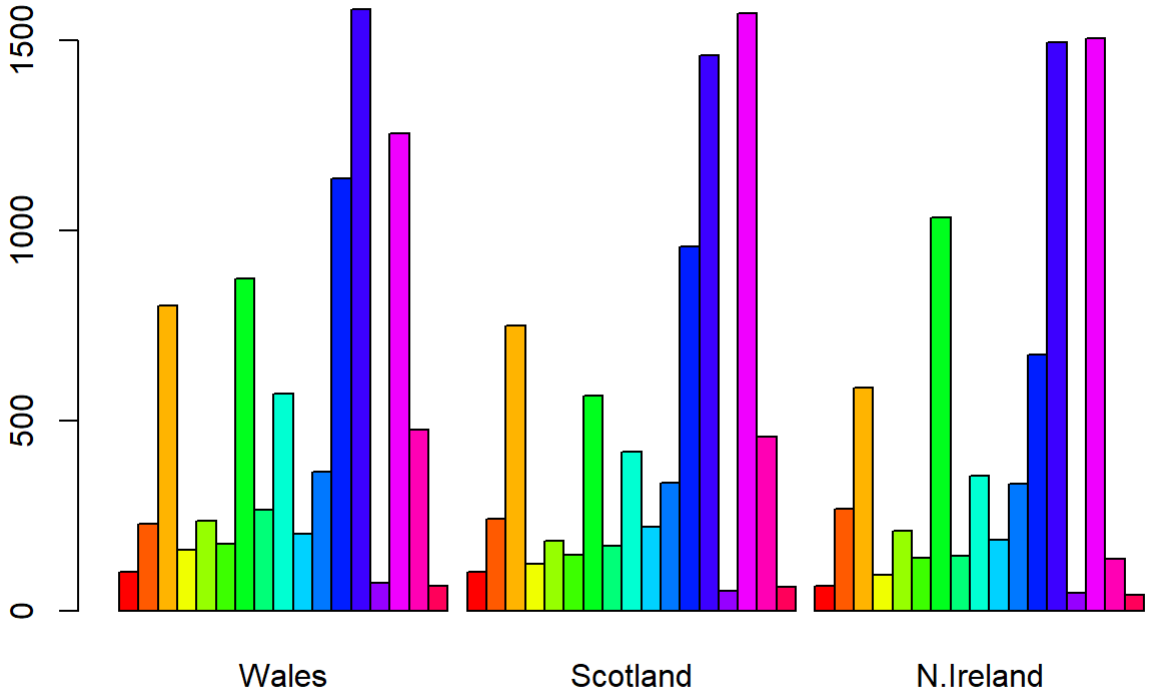
```
    Wales Scotland N.Ireland
105   103      103        66
245   227      242       267
685   803      750       586
147   160      122        93
193   235      184       209
156   175      147       139
```

""Q2"" Which sorting approach is preferred?

x <- read.csv(url, row.names=1) head(x)` is the preferred code because running the earlier code will cause the number of columns to repeatedly shift.

This can be graphed:

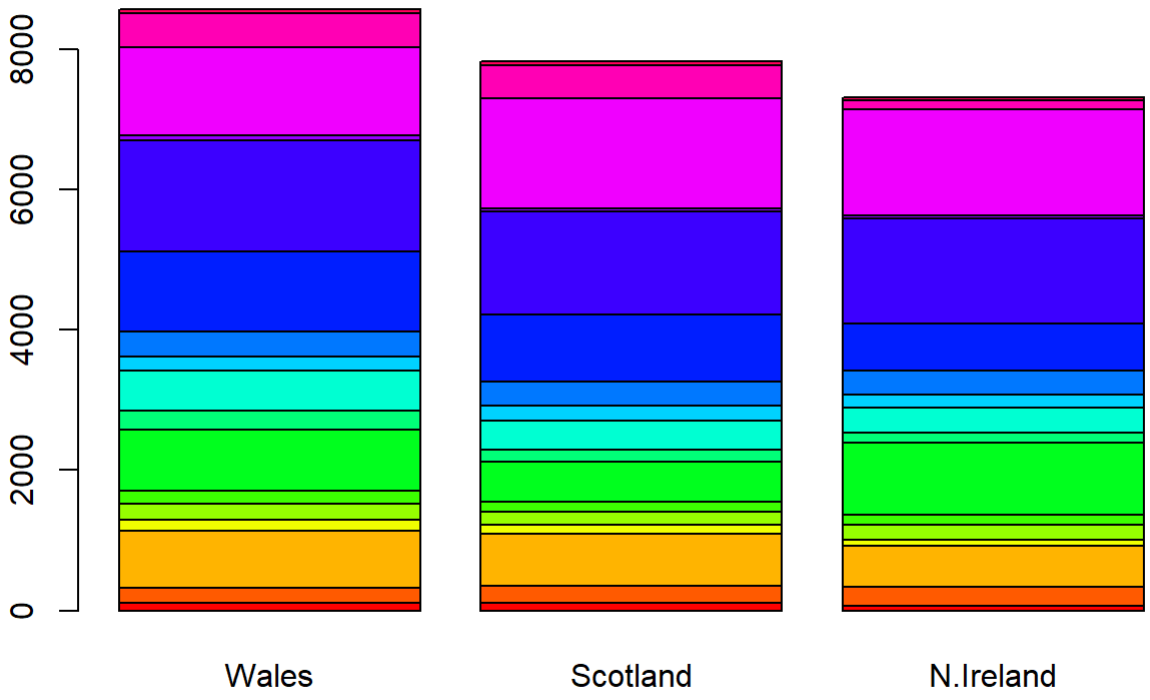
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

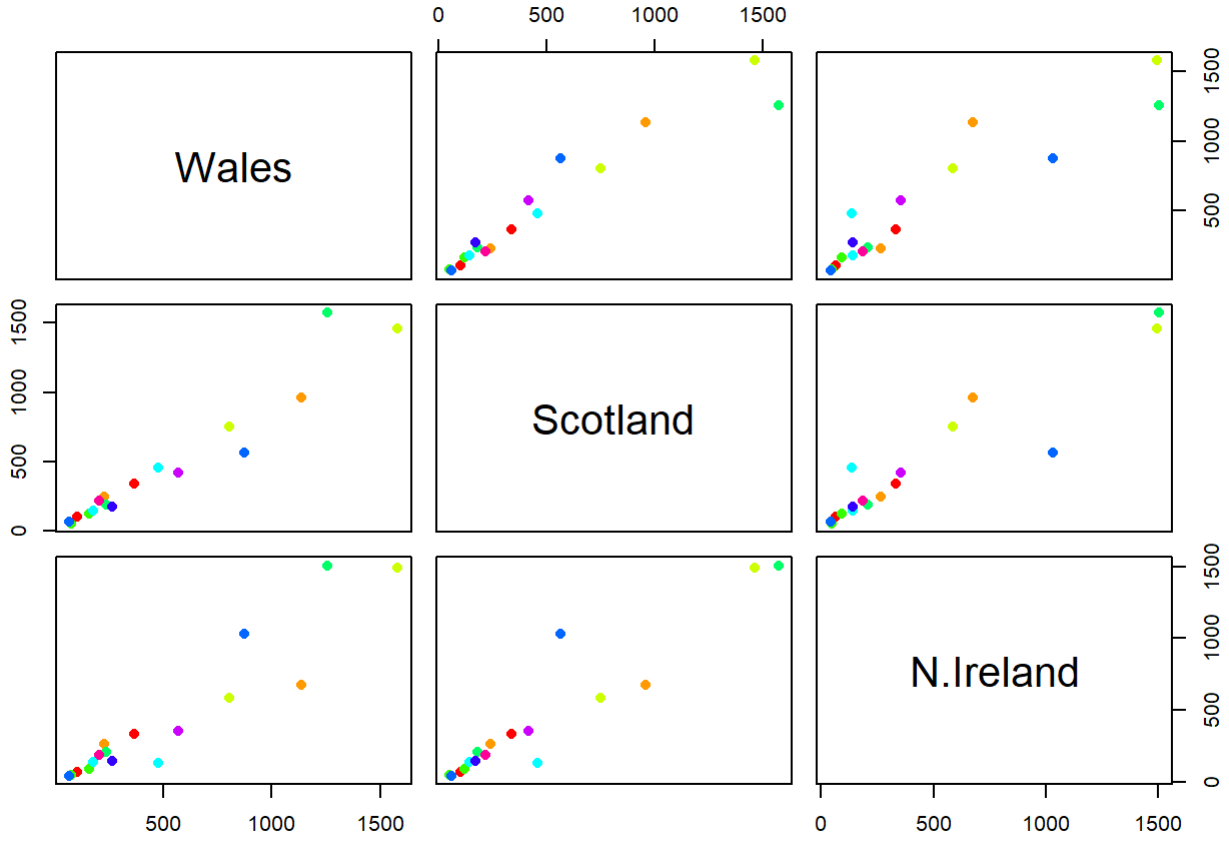
Change the beside factor from true to false

```
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)
```



Each plot is a similarity graph between the y axis country and x axis country. Each dot represents an area of comparison (ex. fresh fruit). The x and y axis are the levels of consumption for the respective country. The closer a point is to the middle of the graph, the more evenly distributed consuption of that food is between the two countries.

How can we generate a PCA function of this data? The main PCA function is called `prcomp()`, which expects the transpose of our data.

```
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
```

```
attributes(pca)
```

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

```
$class
[1] "prcomp"
```

```
pca$x
```

```
                PC1        PC2           PC3
Wales      -288.9534  226.36855  2.296774e-14
Scotland   -141.3603 -284.81172  4.517428e-13
N.Ireland   430.3137   58.44317 -1.407069e-13
```

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