Class 7: Clustering and PCA

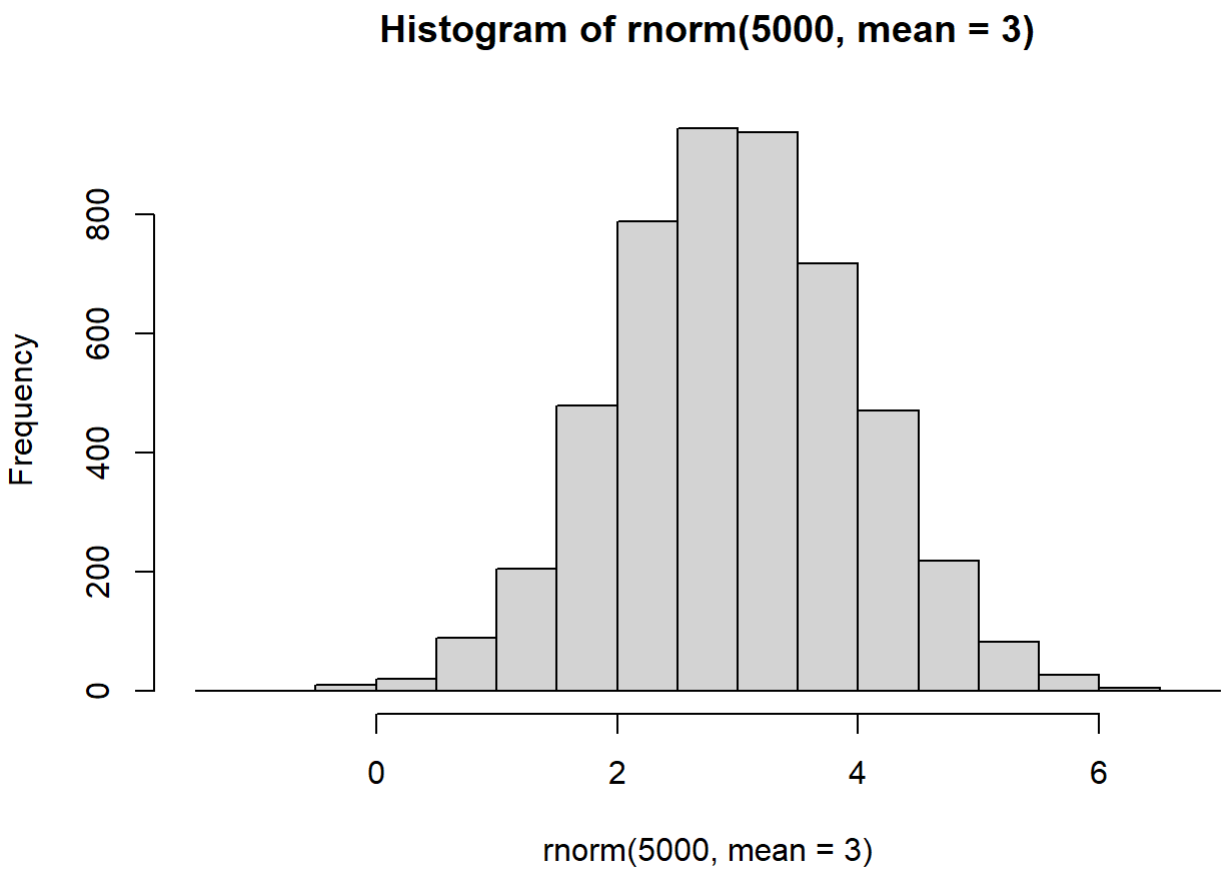
AUTHOR
Hyeseung (Frankie) Son PID: A16025601

Clustering

First let's make up some data to cluster to get a feel for these methods and how they work.

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

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



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

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

```
 [1]  3.6819657  3.2556342  3.9545247  2.4429037  2.0310621  4.0611833
 [7]  5.5056489  2.6176844  1.1461044  2.3127116  4.6898223  1.1311837
[13]  3.2574985  3.5365421  3.1044467  2.5324660  1.8612895  2.8552785
```

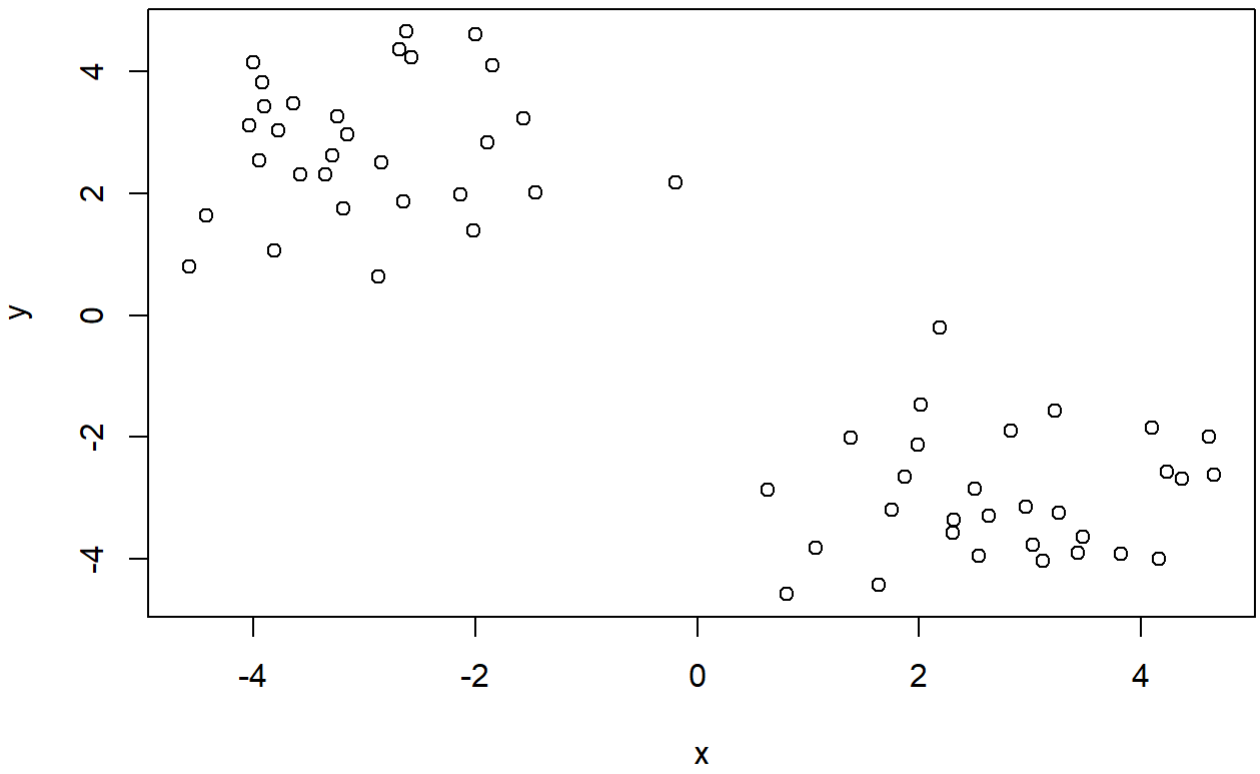
```
[19]  3.0052499  2.8772242  3.5992224  4.7291699  2.4081172  3.1617321
[25]  2.6723925  2.2023044  3.5269407  2.3351980  2.2559003  4.1674618
[31] -2.2658761 -2.3226852 -3.6002231 -3.4311966 -1.3059425 -3.0867173
[37] -3.4996130 -2.1392299 -3.5597296 -5.1763659 -3.4731030 -0.7003227
[43] -3.4006407 -2.3091682 -3.2237272 -2.3549657 -2.0743640 -4.3541240
[49] -3.5278705 -2.4507471 -2.4346241 -3.9327771 -2.4999203 -2.4032178
[55] -3.6902833 -2.3396218 -4.0423318 -2.9636023 -2.7548057 -1.8199553
```

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

```
 [1]  1.7497844  1.0681169  4.3702179  3.2577406  1.3862120  4.1608643
 [7]  3.8178465  2.0116235  1.9903409  3.2275519  3.4336526  0.6307515
[13]  2.5376733  2.6321518  2.1878534  4.2335024  4.6167060  3.4749092
[19]  3.1194058  2.5015952  3.0279666  2.3070960  0.8052735  2.9629719
[25]  4.0995219  2.3106832  2.8305523  4.6566192  1.6358647  1.8715069
[31] -2.6486210 -4.4299540 -2.6203919 -1.8935461 -3.3520025 -1.8459361
[37] -3.1531906 -4.5798429 -3.5736424 -3.7782270 -2.8482956 -4.0383174
[43] -3.6444548 -2.0021695 -2.5783560 -0.1984059 -3.2868032 -3.9440549
[49] -2.8744337 -3.9044387 -1.5621301 -2.1319558 -1.4612182 -3.9247188
[55] -4.0007830 -2.0182850 -3.2423190 -2.6809968 -3.8151565 -3.1942562
```

Put two of these together

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



K-means clustering

This is a very popular clustering method, especially for big data sets. The idea here is that we use `kmeans()`

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

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

Cluster means:
          x         y
1  2.763885 -2.974230
2 -2.974230  2.763885

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] 66.20421 66.20421
 (between_SS / total_SS =  88.2 %)
```

```
Available components:

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

```
# Which component details cluster size?
km$size
```

```
[1] 30 30
```

```
# Which component details cluster assignment/membership?
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
```

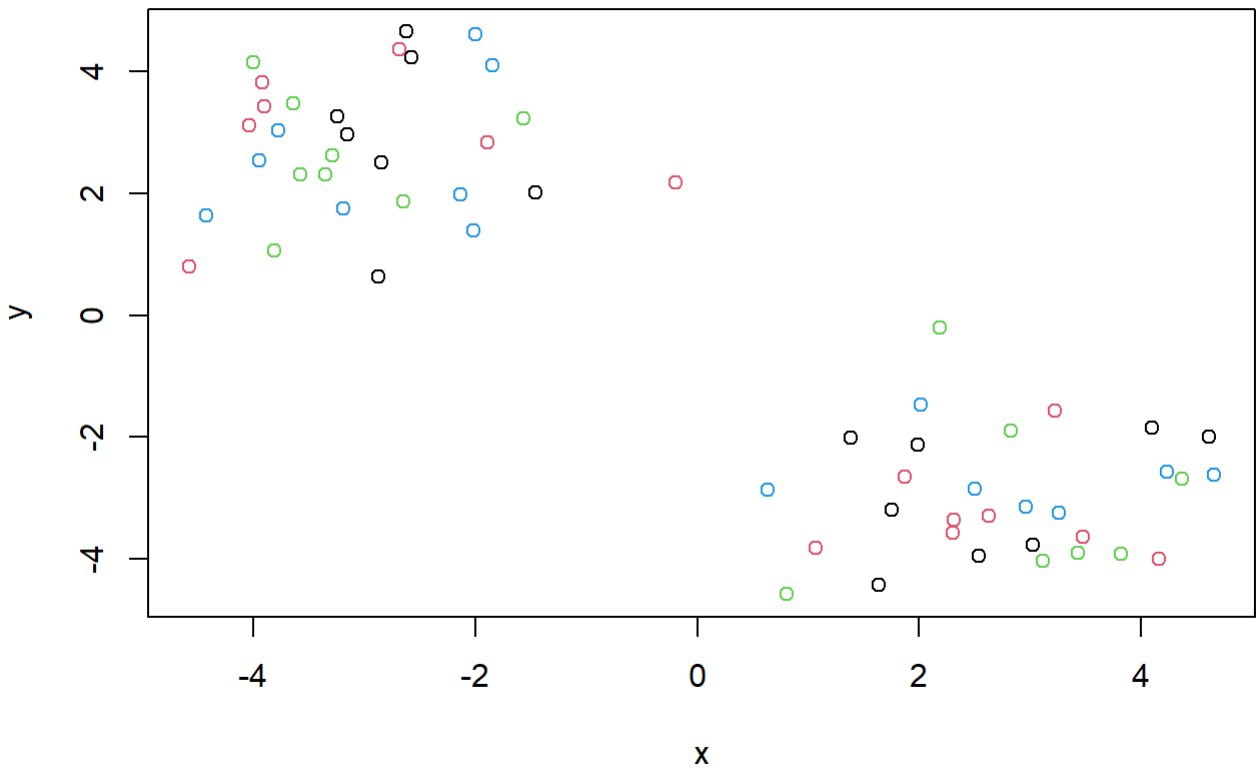
```
# Which component details each cluster centers?
km$centers
```

```
          x         y
1  2.763885 -2.974230
2 -2.974230  2.763885
```

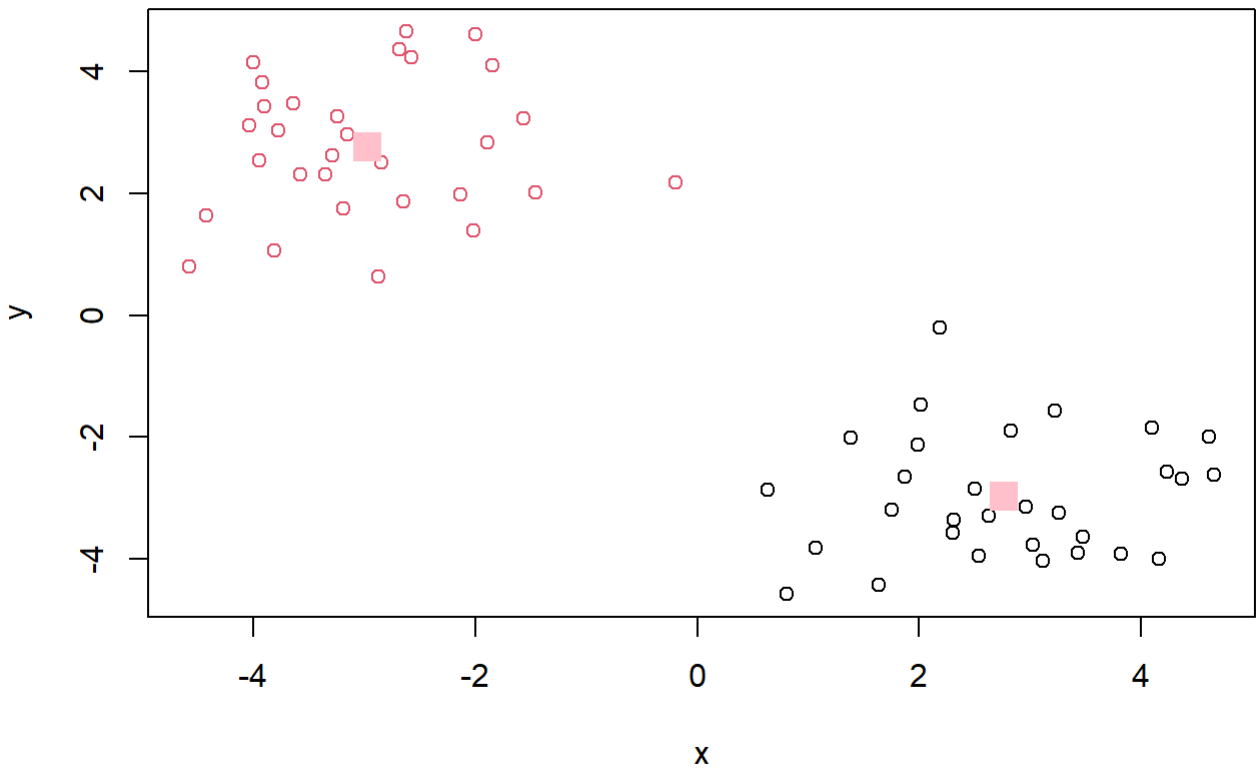
```
# Which component details how spread the points are?
km$tot.withinss
```

```
[1] 132.4084
```

```
mycols <- c(1,2,3,4)
plot(x, col=mycols)
```

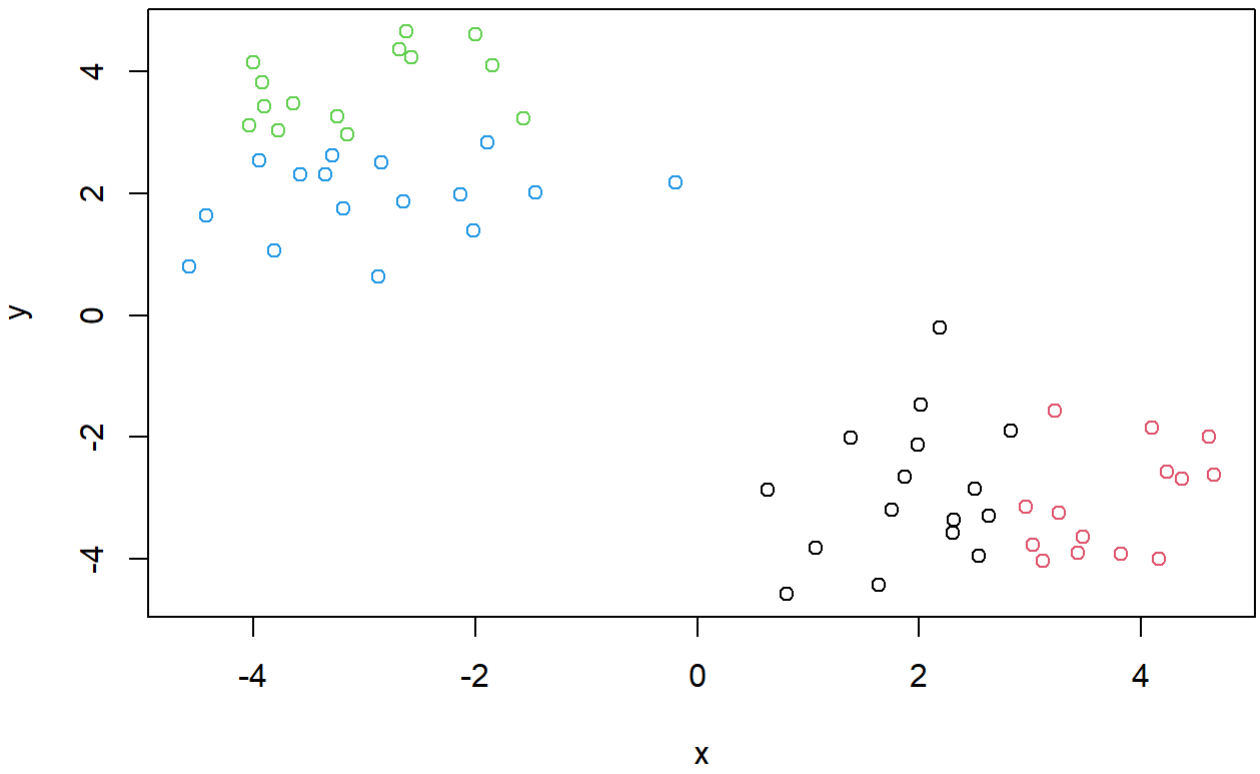



```
plot(x, col=km$cluster)
points(km$centers,col="pink", pch = 15, cex = 2 )
```



> 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)
```



Hierarchical CLustering

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

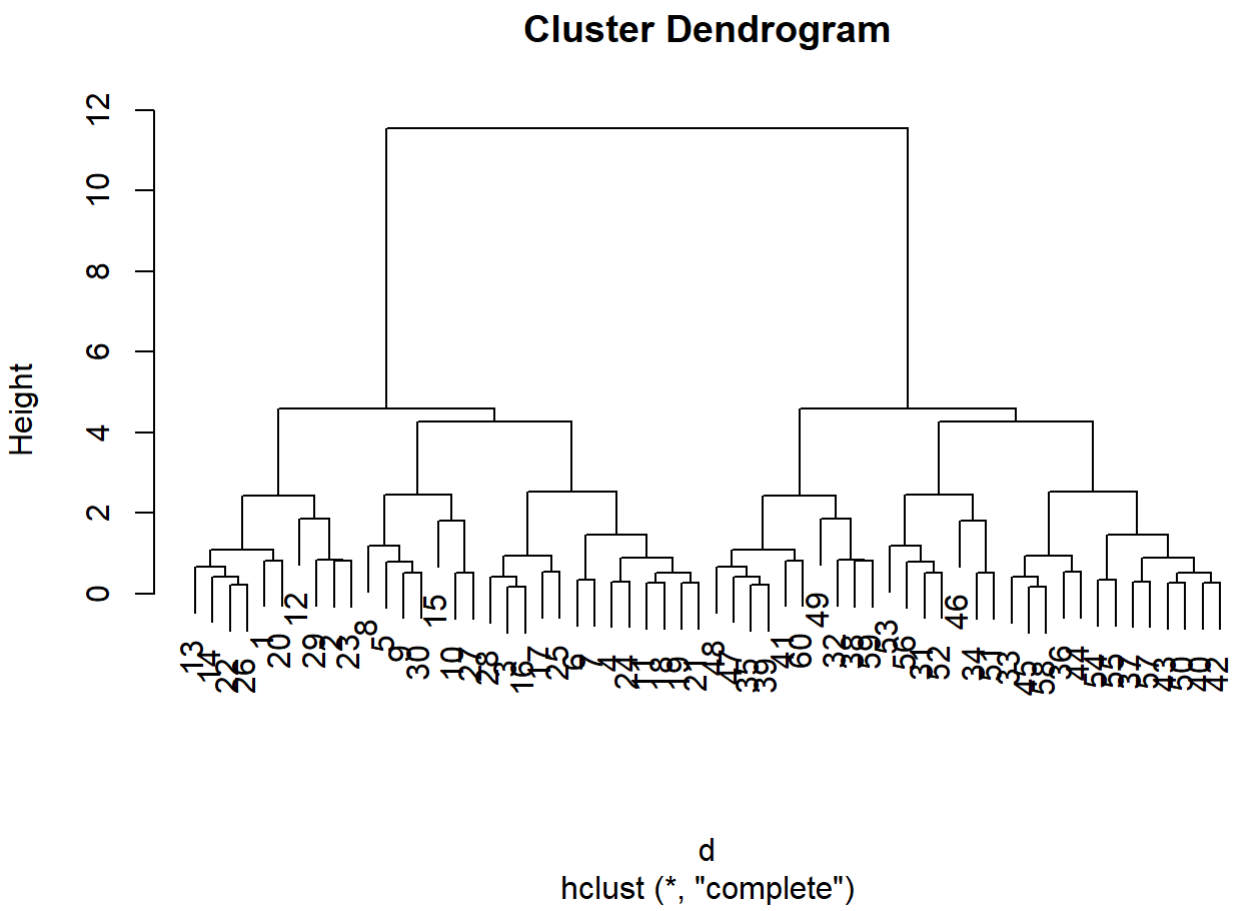
We will start with the `dist()` function.

```
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.

```
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 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
```

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

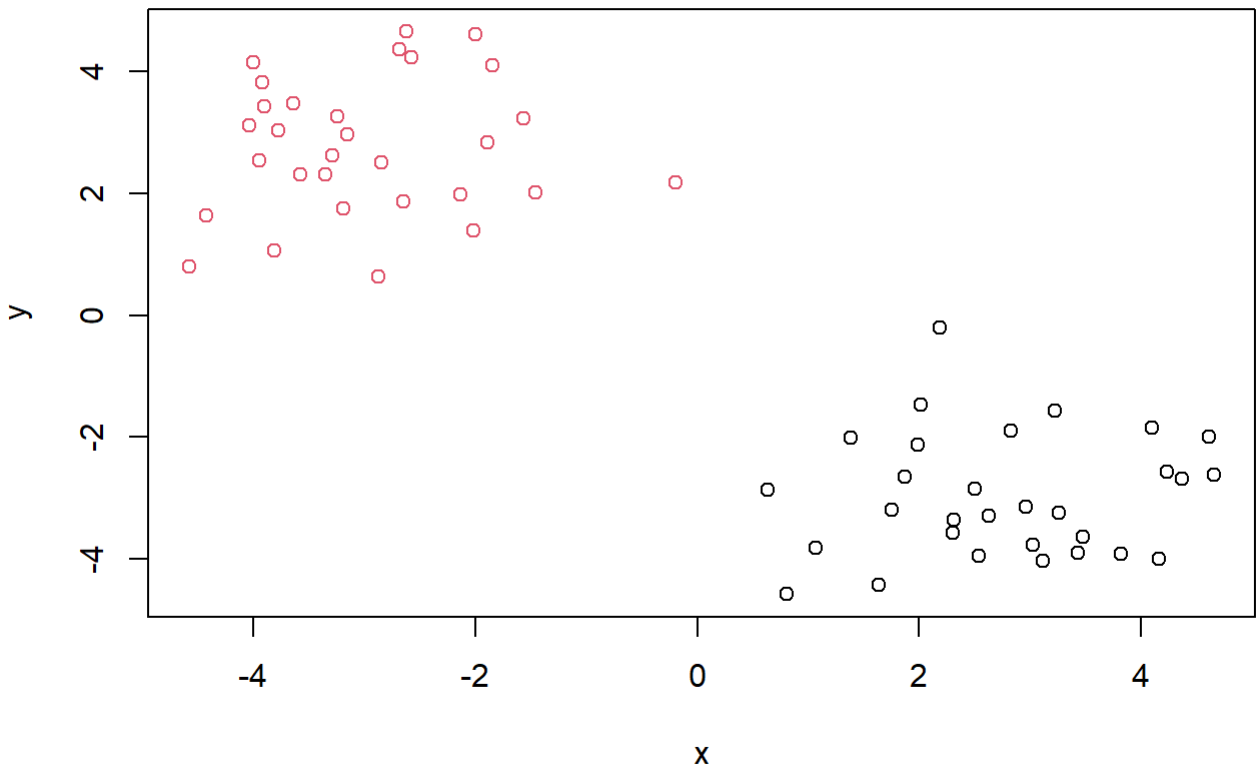
```
cutree(hc, k=2)
```

```
 [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 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
```

```
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 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 of UK Food Data

```
url <- "https://bioboot.github.io/bggn213_f17/class-material/UK_foods.csv"
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

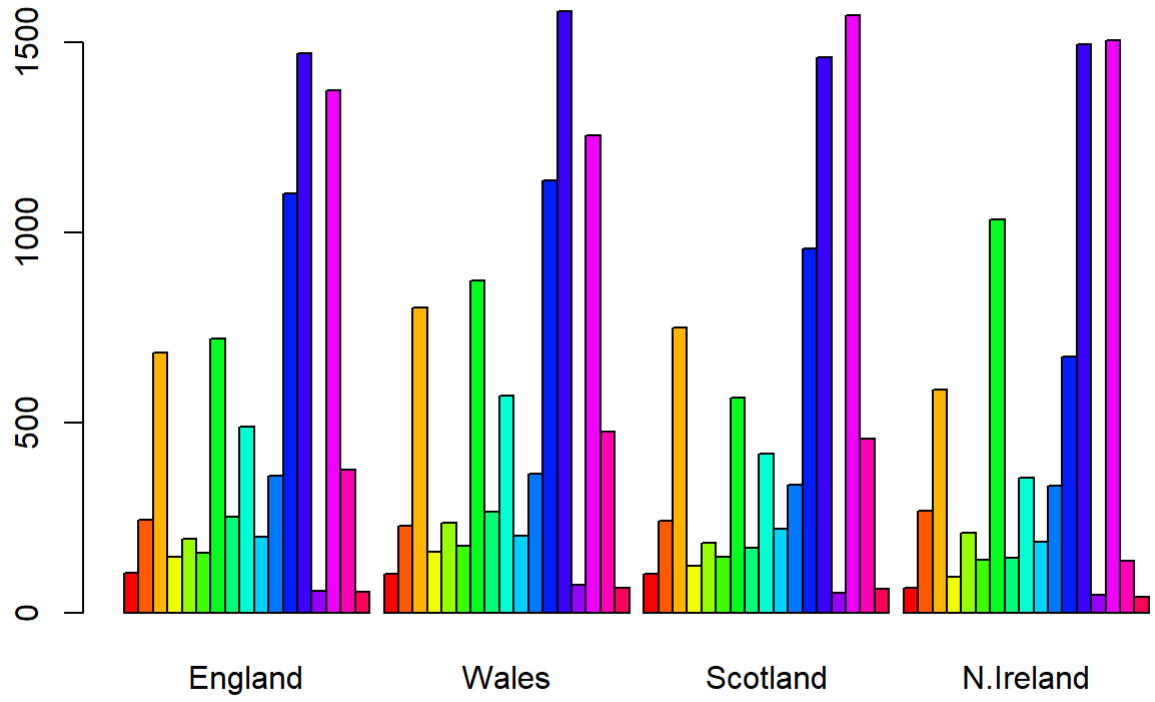
```
View(x)
```

USe `dim()` to find the number of rows and columns for a matrix of data.

```
dim(x)
```

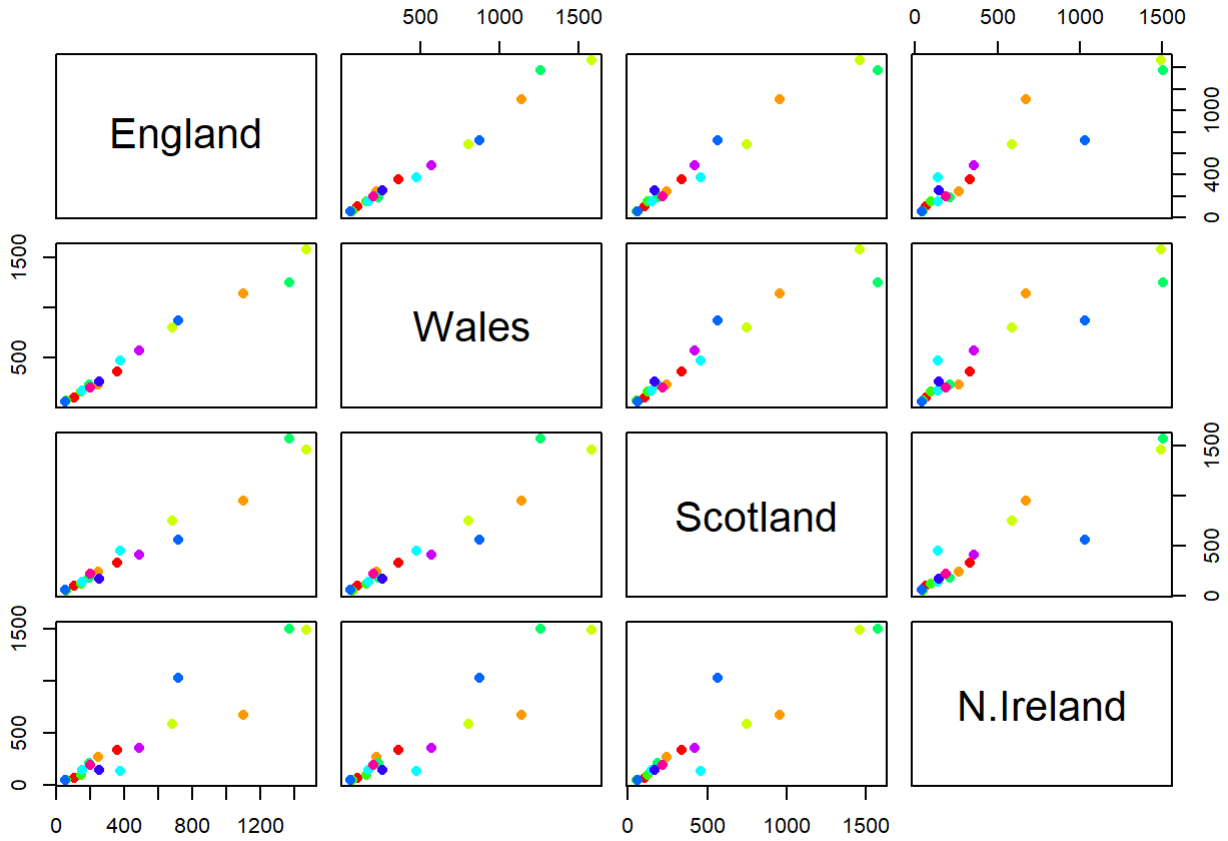
```
[1] 17  4
```

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



A pairs plot of the following data would give:

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




```
#For the orange dot on the right upper most box graphs a plot indicates that there is more consum
```

Let's rearrange the data to a more useful visualization. First we find a new correlation, and a new set of principal axes. The `prcomp()` function is the main function in base R and it expects to transpose our input data.

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

```
                PC1         PC2         PC3           PC4
England    -144.99315    2.532999 -105.768945  2.842865e-14
Wales      -240.52915  224.646925   56.475555  7.804382e-13
Scotland    -91.86934 -286.081786   44.415495 -9.614462e-13
N.Ireland   477.39164   58.901862    4.877895  1.448078e-13
```

We'll use x=PC1 v. y=PC2 as our new plot.

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