Class07: Clustering and PCA

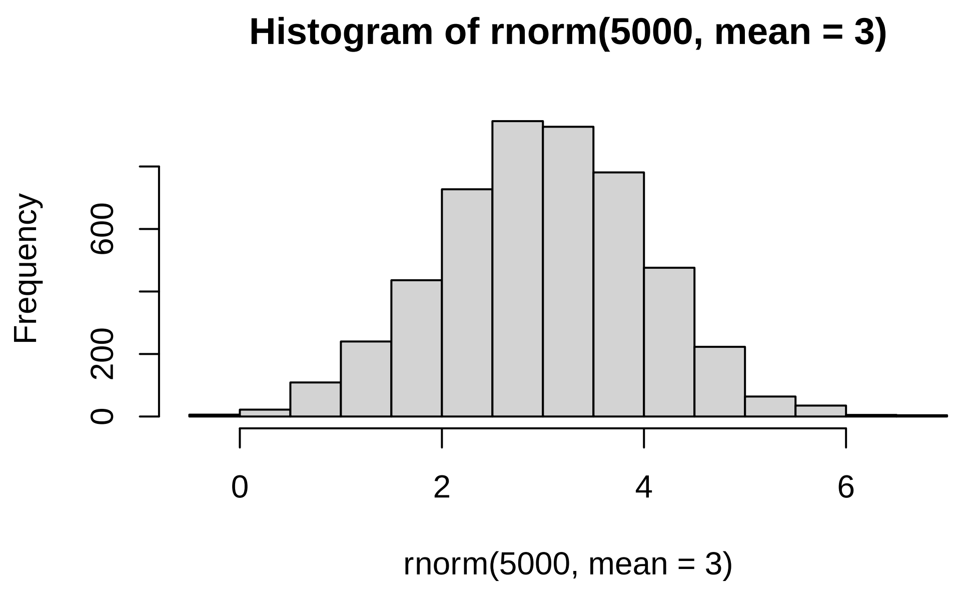
Eric Garcia Ordunez (A16316409)

#Clustering

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

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

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



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

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

```
 [1]  3.9432850  4.8722022  2.9749674  4.5421338  3.7049116  3.4127286
 [7]  3.2186652  3.4632978  2.7982153  2.9033447  0.6614982  2.7343330
[13]  3.6429153  5.0387579  3.0197487  3.7484452  2.2531125  2.7423207
[19]  1.6786133  2.8700320  3.5433802  3.0470769  3.9144089  6.3532483
[25]  3.8134372  0.5727041  3.1380112  2.5821383  2.5918790  1.8088294
[31] -2.3552644 -3.5743954 -4.0743972 -2.3015844 -2.1938405 -2.0172069
[37] -2.4222090 -2.4466368 -2.5584100 -1.7113266 -3.0537078 -2.5957530
[43] -1.1300887 -1.7125178 -2.6453741 -4.5353387 -3.0582564 -2.7490909
[49] -2.7472791 -2.9400898 -1.6865771 -2.6601737 -5.0071821 -2.5816637
[55] -2.7327963 -0.7433062 -3.4543034 -1.7691234 -1.8400599 -5.5580137
```

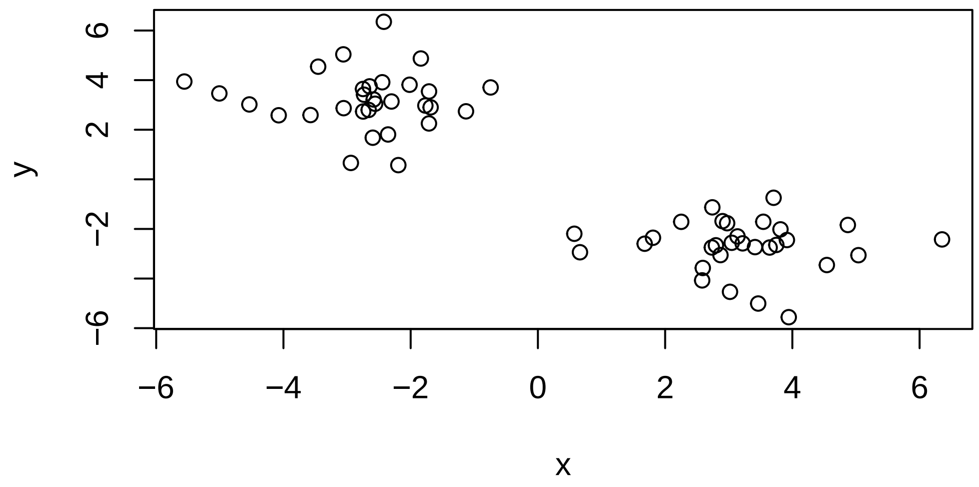
Put two these together:

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

```
               x          y
 [1,]  3.9432850 -5.5580137
 [2,]  4.8722022 -1.8400599
 [3,]  2.9749674 -1.7691234
 [4,]  4.5421338 -3.4543034
 [5,]  3.7049116 -0.7433062
 [6,]  3.4127286 -2.7327963
 [7,]  3.2186652 -2.5816637
 [8,]  3.4632978 -5.0071821
 [9,]  2.7982153 -2.6601737
[10,]  2.9033447 -1.6865771
[11,]  0.6614982 -2.9400898
[12,]  2.7343330 -2.7472791
[13,]  3.6429153 -2.7490909
[14,]  5.0387579 -3.0582564
[15,]  3.0197487 -4.5353387
[16,]  3.7484452 -2.6453741
[17,]  2.2531125 -1.7125178
[18,]  2.7423207 -1.1300887
[19,]  1.6786133 -2.5957530
[20,]  2.8700320 -3.0537078
[21,]  3.5433802 -1.7113266
```

```
[22,]  3.0470769 -2.5584100
[23,]  3.9144089 -2.4466368
[24,]  6.3532483 -2.4222090
[25,]  3.8134372 -2.0172069
[26,]  0.5727041 -2.1938405
[27,]  3.1380112 -2.3015844
[28,]  2.5821383 -4.0743972
[29,]  2.5918790 -3.5743954
[30,]  1.8088294 -2.3552644
[31,] -2.3552644  1.8088294
[32,] -3.5743954  2.5918790
[33,] -4.0743972  2.5821383
[34,] -2.3015844  3.1380112
[35,] -2.1938405  0.5727041
[36,] -2.0172069  3.8134372
[37,] -2.4222090  6.3532483
[38,] -2.4466368  3.9144089
[39,] -2.5584100  3.0470769
[40,] -1.7113266  3.5433802
[41,] -3.0537078  2.8700320
[42,] -2.5957530  1.6786133
[43,] -1.1300887  2.7423207
[44,] -1.7125178  2.2531125
[45,] -2.6453741  3.7484452
[46,] -4.5353387  3.0197487
[47,] -3.0582564  5.0387579
[48,] -2.7490909  3.6429153
[49,] -2.7472791  2.7343330
[50,] -2.9400898  0.6614982
[51,] -1.6865771  2.9033447
[52,] -2.6601737  2.7982153
[53,] -5.0071821  3.4632978
[54,] -2.5816637  3.2186652
[55,] -2.7327963  3.4127286
[56,] -0.7433062  3.7049116
[57,] -3.4543034  4.5421338
[58,] -1.7691234  2.9749674
[59,] -1.8400599  4.8722022
[60,] -5.5580137  3.9432850
```

```
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 -2.695199  3.186288
2  3.186288 -2.695199

Clustering vector:
 [1] 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 1 1 1 1 1 1 1 1
[39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Within cluster sum of squares by cluster:
[1] 72.94783 72.94783
 (between_SS / total_SS =  87.7 %)
```

```
Available components:

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

Q How many points are in each cluster?

```r
km$size
```

```
[1] 30 30
```

Q what component of your result object details - cluster size?

- cluster assignment/membership?

```r
km$cluster
```

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

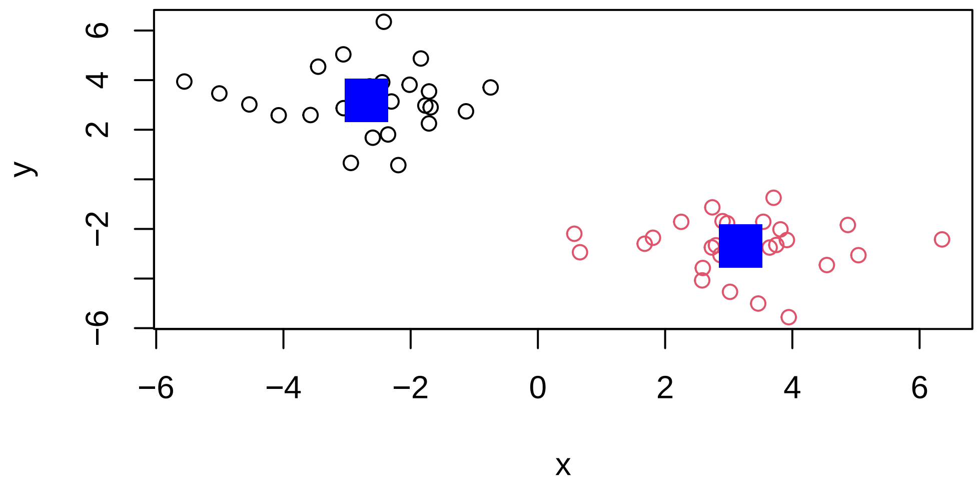
- cluster center?

```r
km$centers
```

```
          x         y
1 -2.695199  3.186288
2  3.186288 -2.695199
```

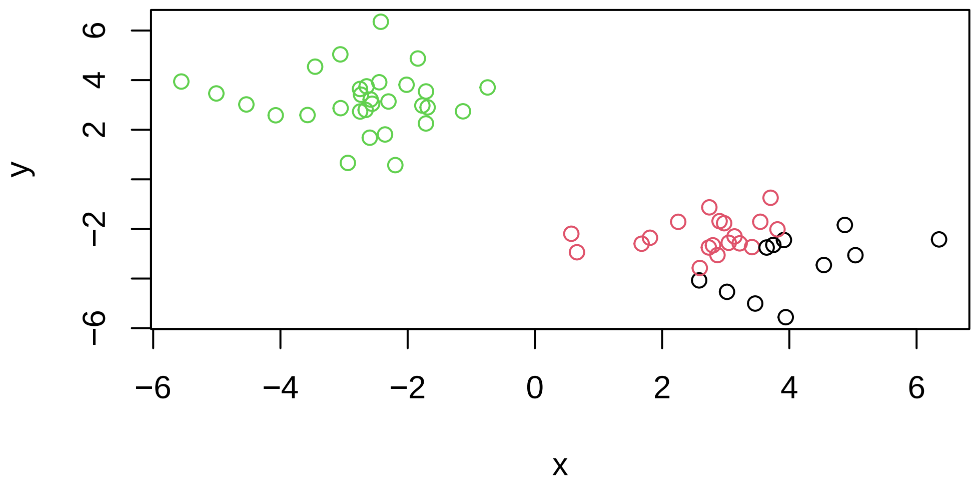
Q. Plot x colored by the kmeans cluster assignment and add cluster centers blue

```r
mycols <- c(1,2)
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=3)
plot(x, col= km$cluster)
```



Hierarchial Clustering

We can use the `hclust()` funciton for Hiierarchial 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.

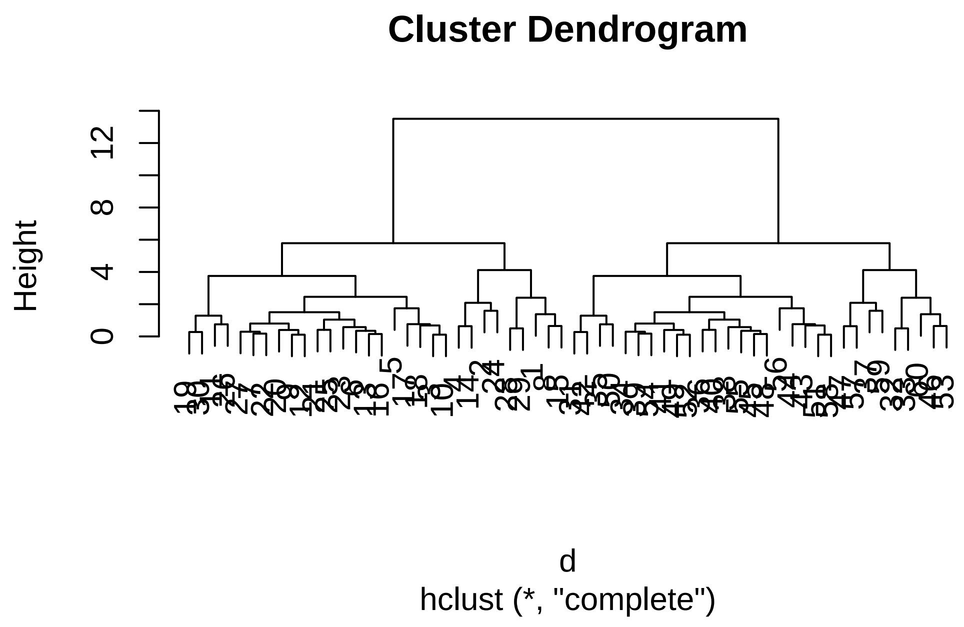
```
d <- dist(x)
hc <- hclust(d)
hc
```

```

Call:
hclust(d = d)

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

```r
plot(hc)
```



I can now "cut" my tree with the `cutree()` to yield a cluster membership vector.

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

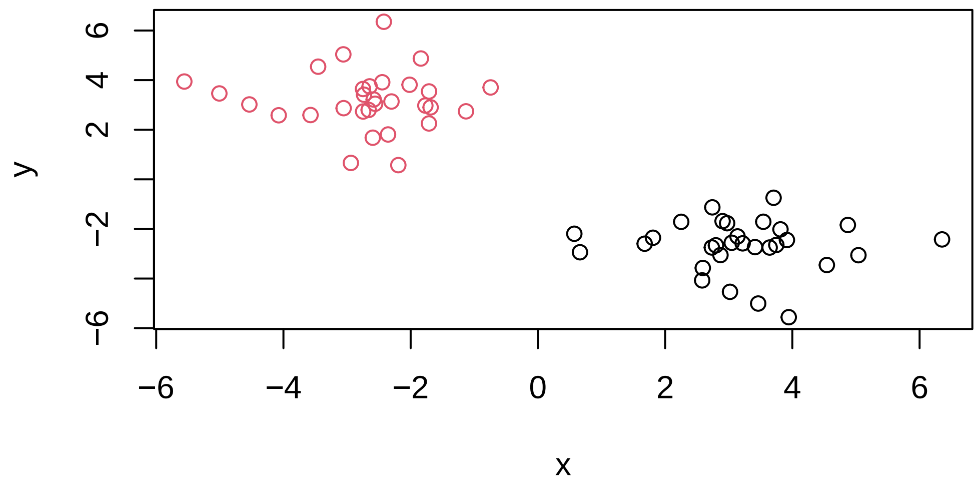
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

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

```r
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 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
```

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



Principle Component Analysis (PCA)

```r
# Adding the `row.names()` helps the first column to have its own.
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
```

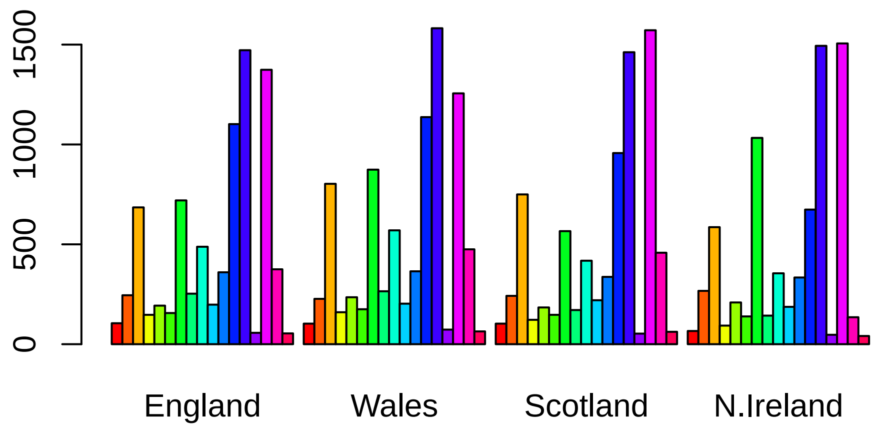
Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

```
dim(x)
```

```
[1] 17  4
```

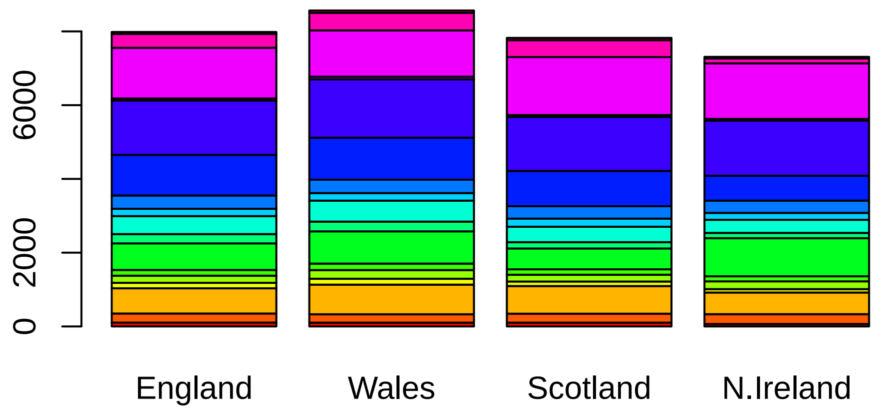
Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

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



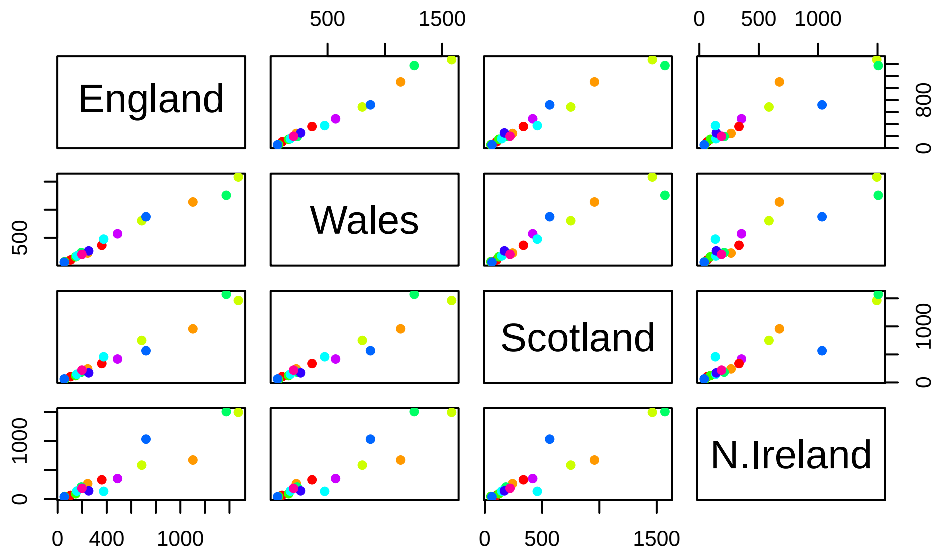
Q3: Changing what optional argument in the above barplot() function results in the following plot?

```r
barplot(as.matrix(x), beside=FALSE, 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?

```r
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?

The main PCA function is base R is called `prcomp()` it expects thr trasnpose of our data.

```r
# Use the prcomp() PCA function
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
```

```r
attributes(pca)
```

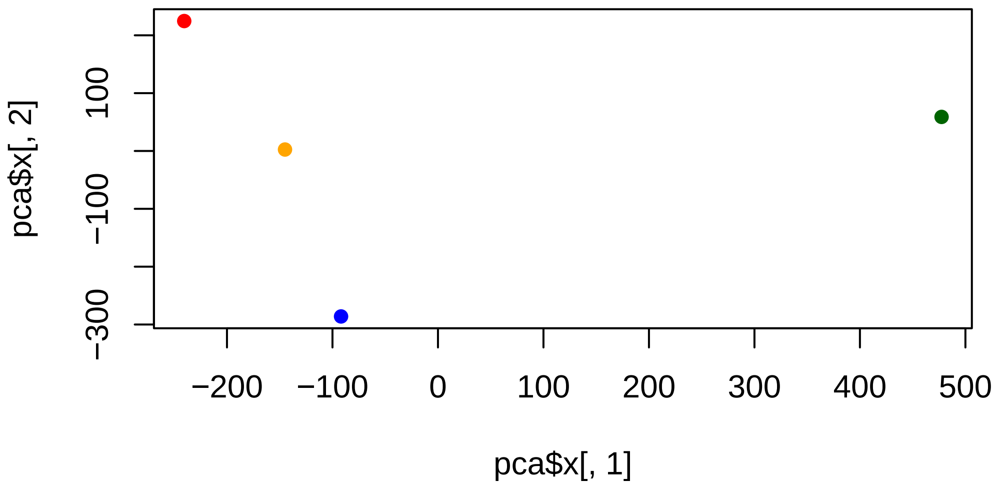
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$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
```

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


pca$x[, 1]
pca$x[, 2]