Class 7: Machine Learning I

Troy Lee A17078296

today we are going to learn how to apply different machine learning methods, begining with clustering:

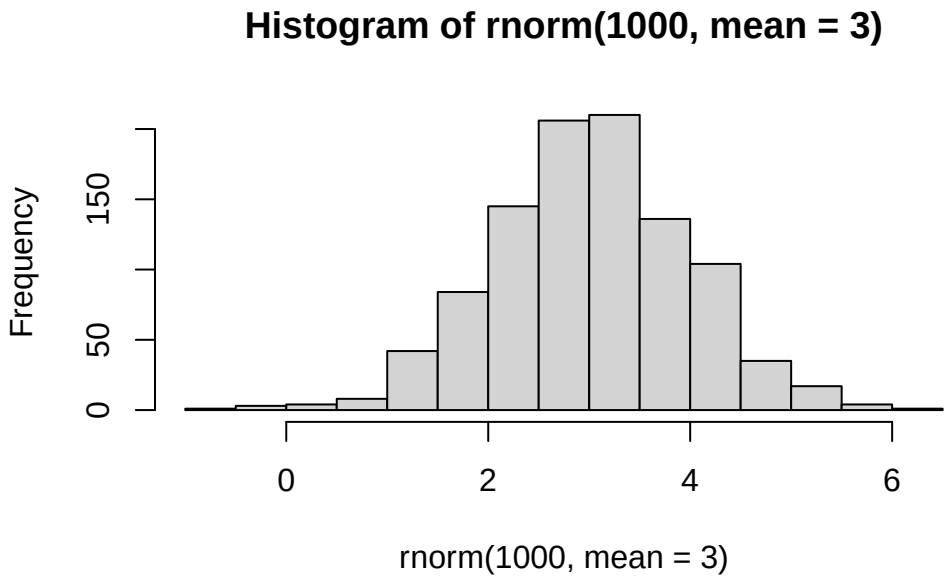
The goal here is to find groups/clusters in your input data.

Firt I will make up some data with clear groups. For this I will use the `rnorm()` function.

```
rnorm(10)
```

```
 [1] -0.86946347 -0.03835680  0.03849283 -0.44457968  0.42913850 -1.10065577
 [7] -0.44299064 -1.02870482 -2.70551148 -0.58247700
```

```
hist(rnorm(1000, mean=3))
```

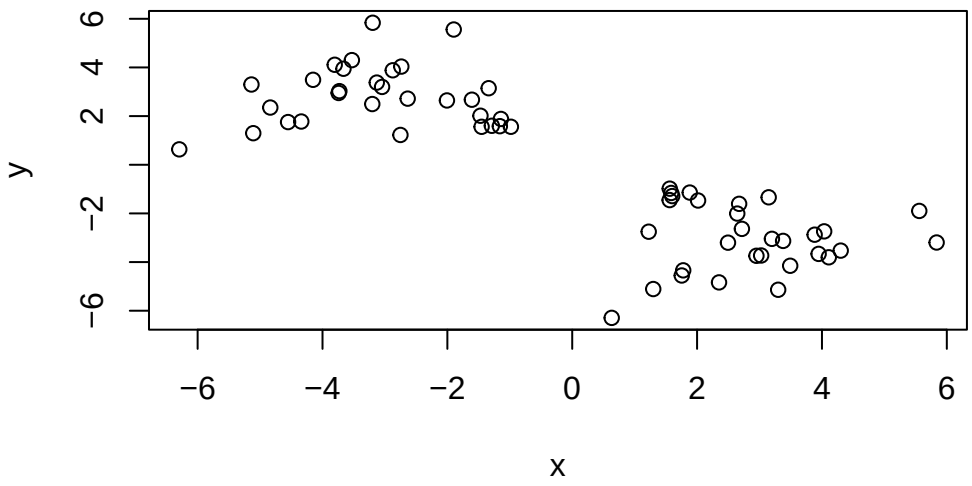


```r
n <- 30
x <- c(rnorm(n, -3), rnorm(n,+3))
y <- rev(x)

z <- cbind(x,y)
head(z)
```

```
             x         y
[1,] -2.737766 4.0355635
[2,] -2.874916 3.8833586
[3,] -3.529370 4.3005543
[4,] -6.294028 0.6325367
[5,] -1.142330 1.8829219
[6,] -3.741316 2.9507372
```

```r
plot(z)
```



Use the `kmeans()` function setting k to 2 and nstart=20

Inspect/print the results

Q. How many points are in each cluster?

Q. What 'component' of your result objct details - cluster size? - cluster assignment/membership? - cluster center?

Q. Plot z colored by the kmeans cluster assignment and add cluster centers as blue points

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

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

Cluster means:
          x         y
1 -3.026897  2.797971
2  2.797971 -3.026897

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 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

Within cluster sum of squares by cluster:
[1] 102.6464 102.6464
 (between_SS / total_SS =  83.2 %)

Available components:

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

Results in kmeans object `km`

```
attributes(km$centers)
```

```
$dim
[1] 2 2

$dimnames
$dimnames[[1]]
[1] "1" "2"

$dimnames[[2]]
[1] "x" "y"
```

cluster size?

```
km$size
```

```
[1] 30 30
```

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 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
```

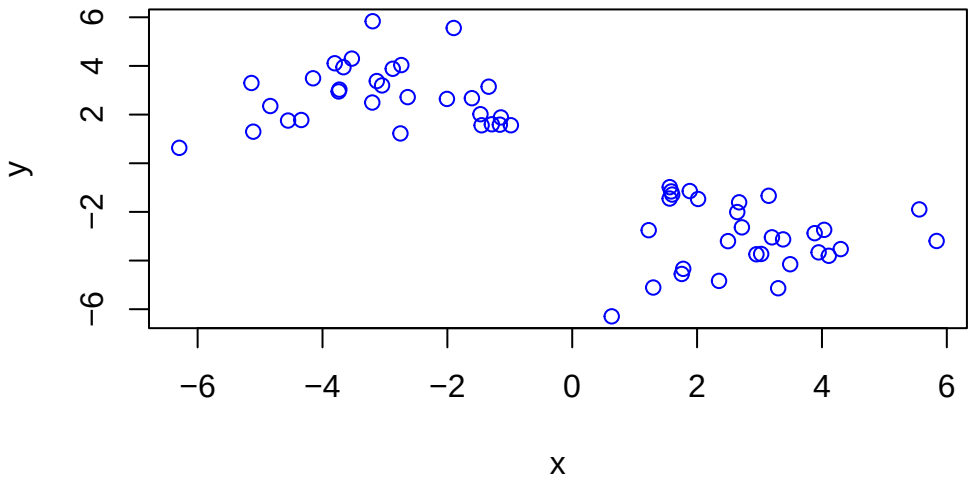
cluster center?

```
km$center
```

```
          x         y
1 -3.026897  2.797971
2  2.797971 -3.026897
```

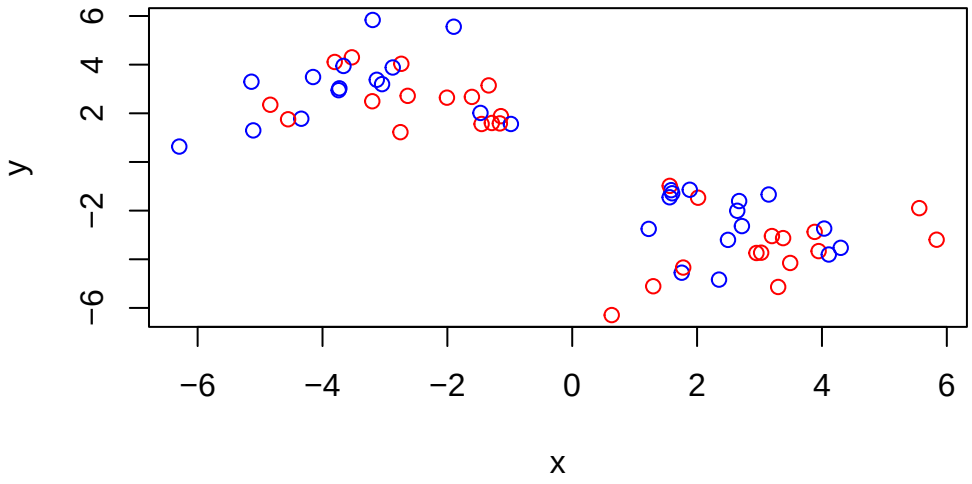
> Q. Plot z colored by the kmeans cluster assignment and add cluster centers as blue points

```
plot(z, col="blue")
```

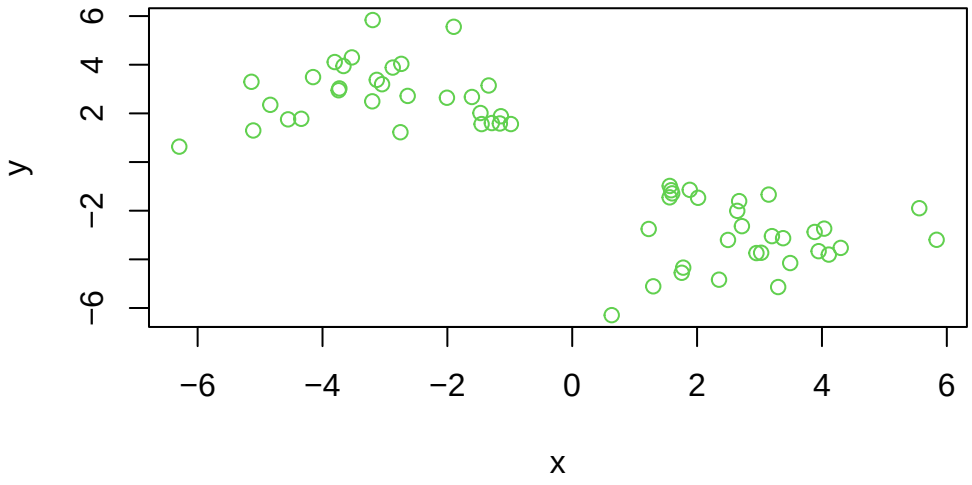


R will re-cycle the shorter color vector to be the same length as the longer (number of data points) in z

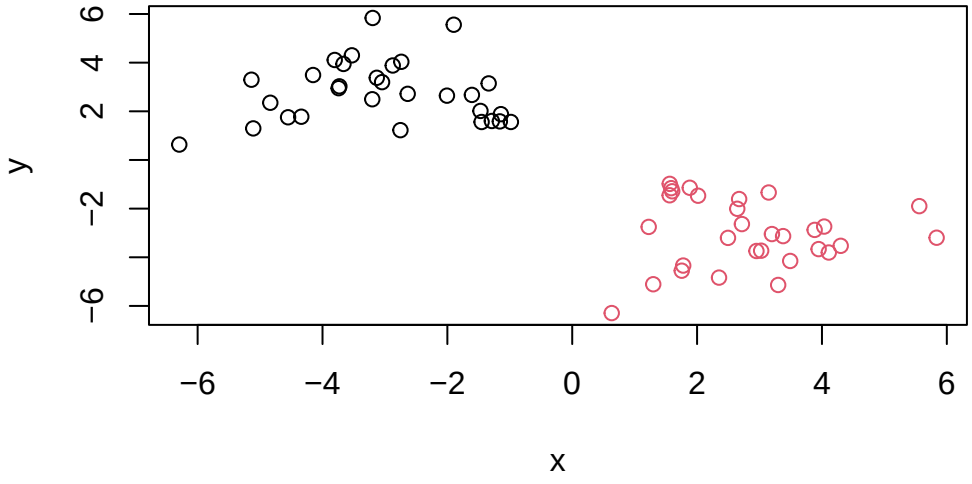
```
plot(z, col=c("red","blue"))
```



```r
plot(z, col=3)
```

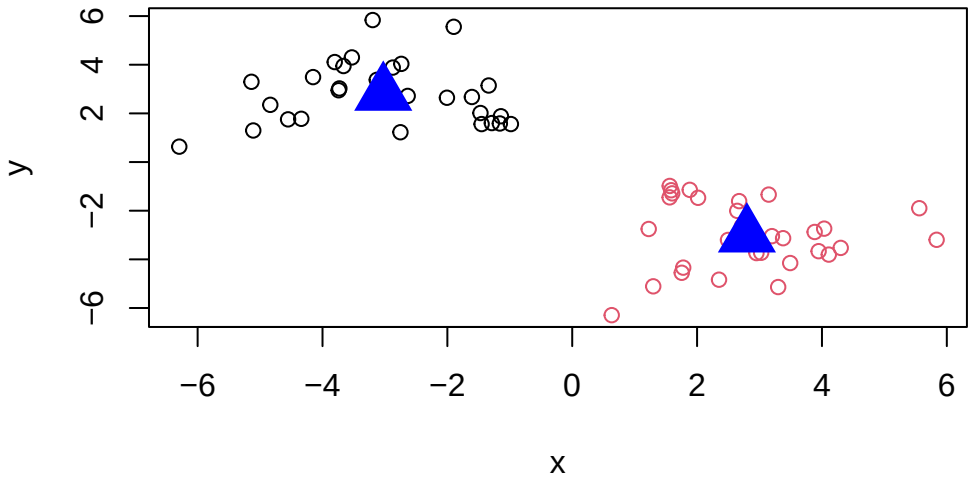



```r
plot(z, col=km$cluster)
```



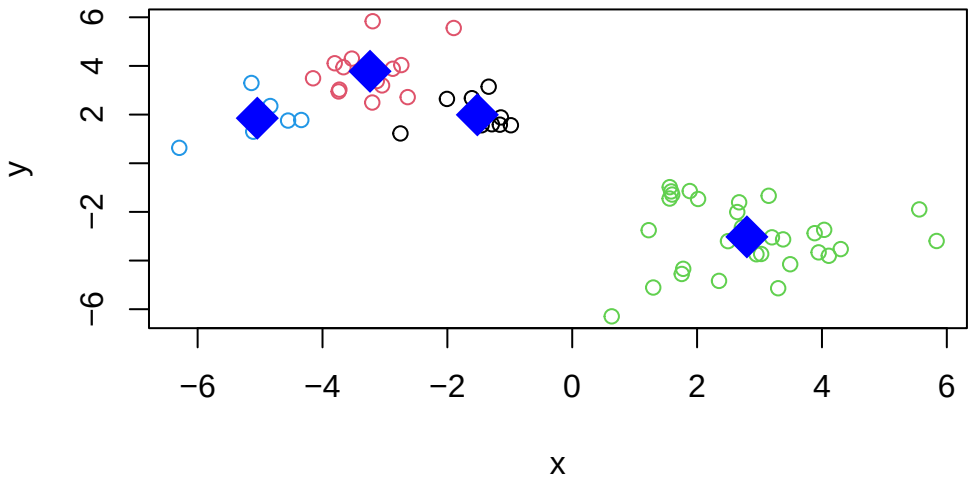
We can use the `points()` function to add new points to an existing plot... like the cluster centers.

```
plot(z, col=km$cluster)
points(km$centers, col="blue", pch=17, cex=3)
```



Q. Can you run kmeans and ask for 4 clusters please and plot the results like we have done above?

```
km4 <- kmeans(z, centers =4)
plot(z, col=km4$cluster)
points(km4$centers, col="blue", pch=18, cex=3)
```



Hierarchical Clustering

Let's take our same made-up data **z** and see how hclust works.

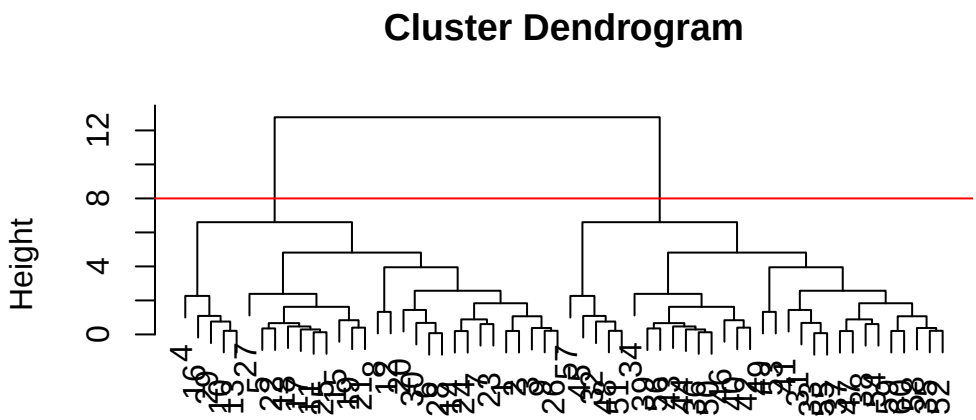
First we need a distance matrix of our data to be clustered

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

```
Call:
hclust(d = d)

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

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



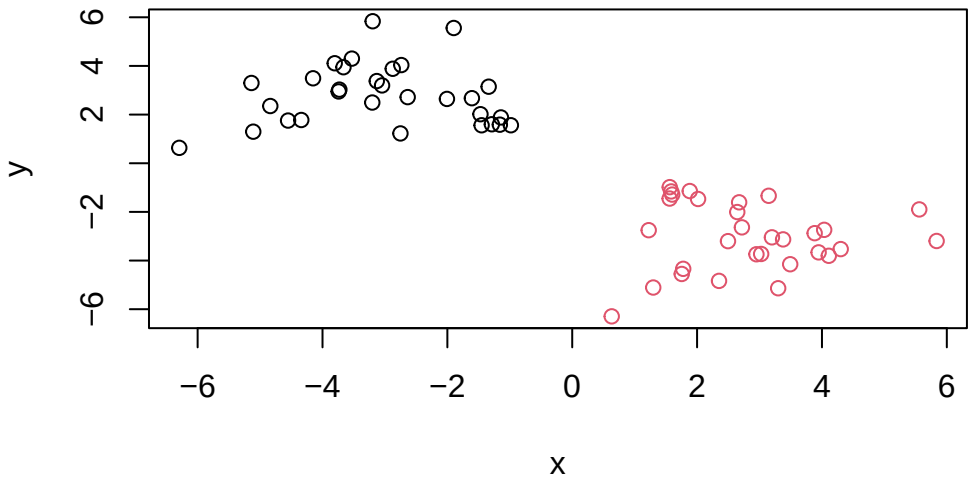
I can get my cluster membership vector by "cutting the tree" with the `cutree()` function like so:

```
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 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
```

Can you plot `z` colored by our hclust results:

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



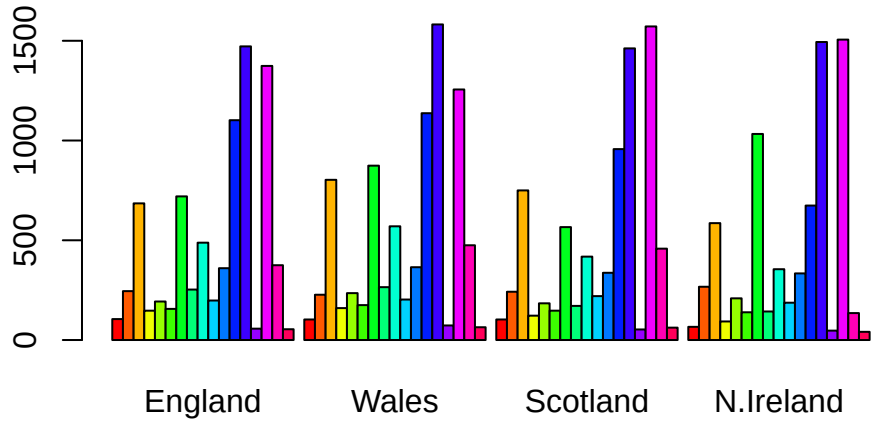
PCA of UK food data

Read data from the UK on food consumption in different parts of the UK

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names=1)
head(x,)
```

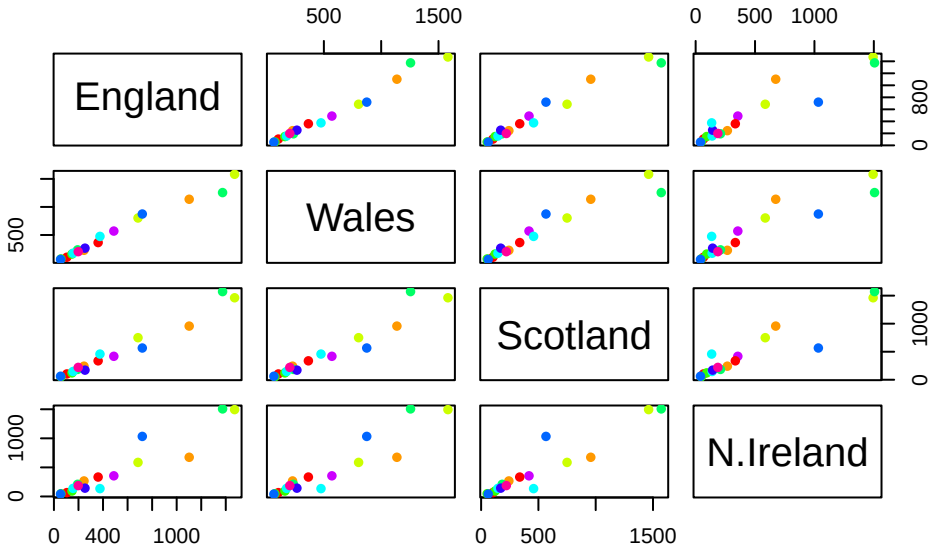
```
              England Wales Scotland N.Ireland
Cheese            105   103      103        66
Carcass_meat      245   227      242       267
Other_meat        685   803      750       586
Fish              147   160      122        93
Fats_and_oils     193   235      184       209
Sugars        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

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



A so-called "Pairs" plot cna be useful for small datasets like this one

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



It is hard to see structure and trends in even this small data-set. How will we ever do this when we have a big datasets with 1,000s or 10s of thousands of things we are measuring...

PCA to the rescue

Let's see how PCA deals with this dataset. So main function in base R to do PCA is called `prcomp()`

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

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 3.176e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

Let's see what is inside this `pca` object that we created from running `prcomp()`

```
attributes(pca)
```

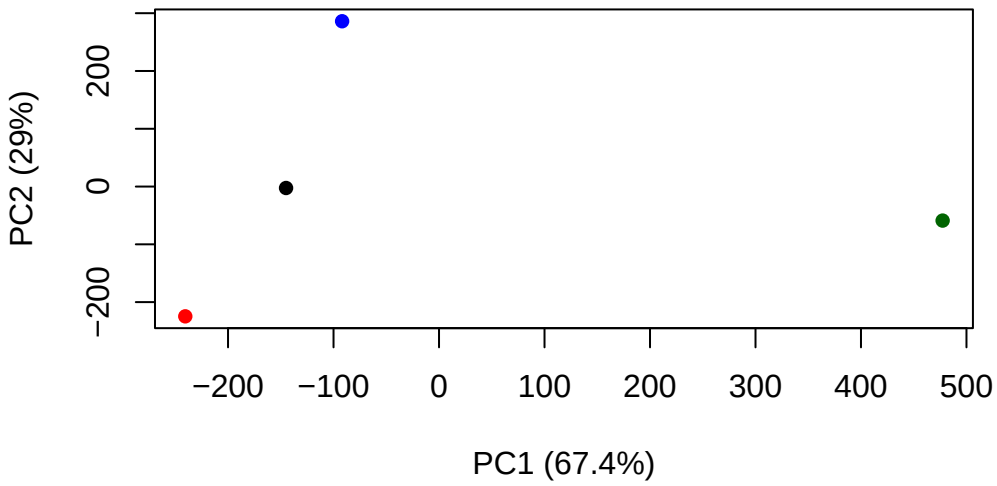
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

```
pca$x
```

```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -4.894696e-14
Wales     -240.52915 -224.646925 -56.475555  5.700024e-13
Scotland   -91.86934  286.081786 -44.415495 -7.460785e-13
N.Ireland  477.39164  -58.901862  -4.877895  2.321303e-13
```

```
plot(pca$x[,1], pca$x[,2], col=c("black", "red", "blue", "darkgreen"), pch=16,
     xlab="PC1 (67.4%)", ylab="PC2 (29%)")
```




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