Class07 Machine Learning 1

Ethan Lai

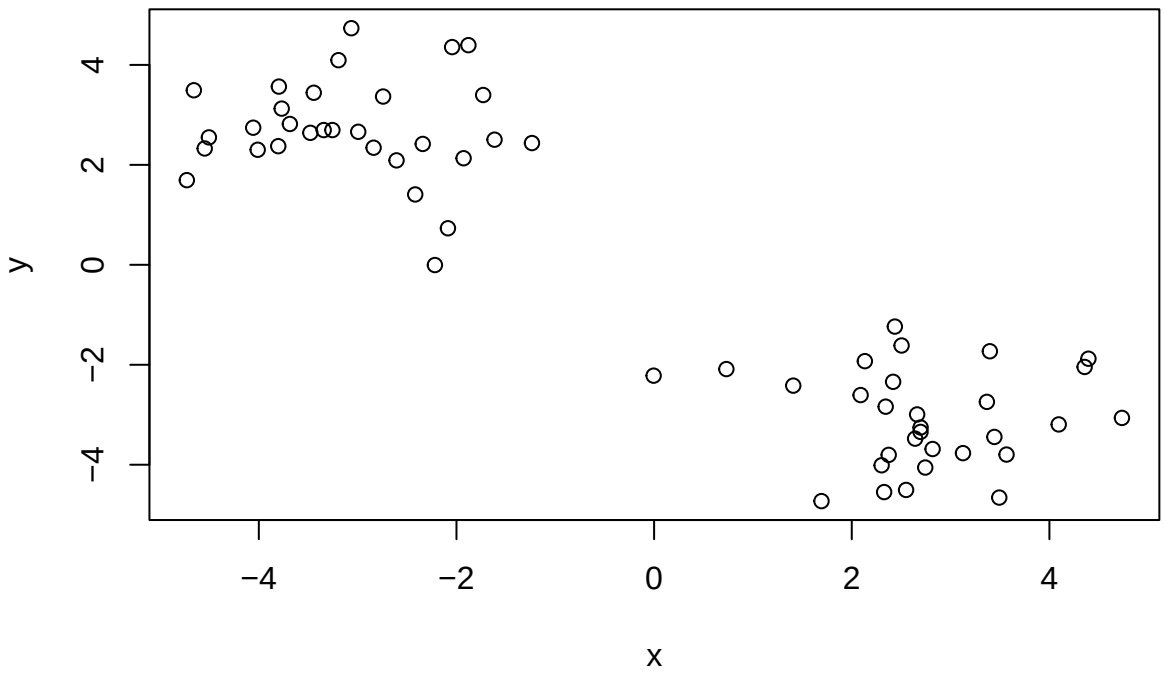
2/9/0222

#Clustering Methods

K-means

First generate some data to test. Concatenate two rnorm samples, and column bind this vector and its reverse.

```
tmp<- c(rnorm(30,-3), rnorm(30,3))
x<-cbind(x=tmp, y=rev(tmp))
plot(x)
```



```r
k <- kmeans(x, centers=2, nstart=10)
k
```

```
## K-means clustering with 2 clusters of sizes 30, 30
##
## Cluster means:
##           x         y
## 1  2.717717 -3.067167
## 2 -3.067167  2.717717
##
## 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 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] 57.88157 57.88157
##  (between_SS / total_SS =  89.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

How many points per cluster?

```r
k$size
```

```
## [1] 30 30
```

What are the centroids of the clusters?

```r
k$centers
```

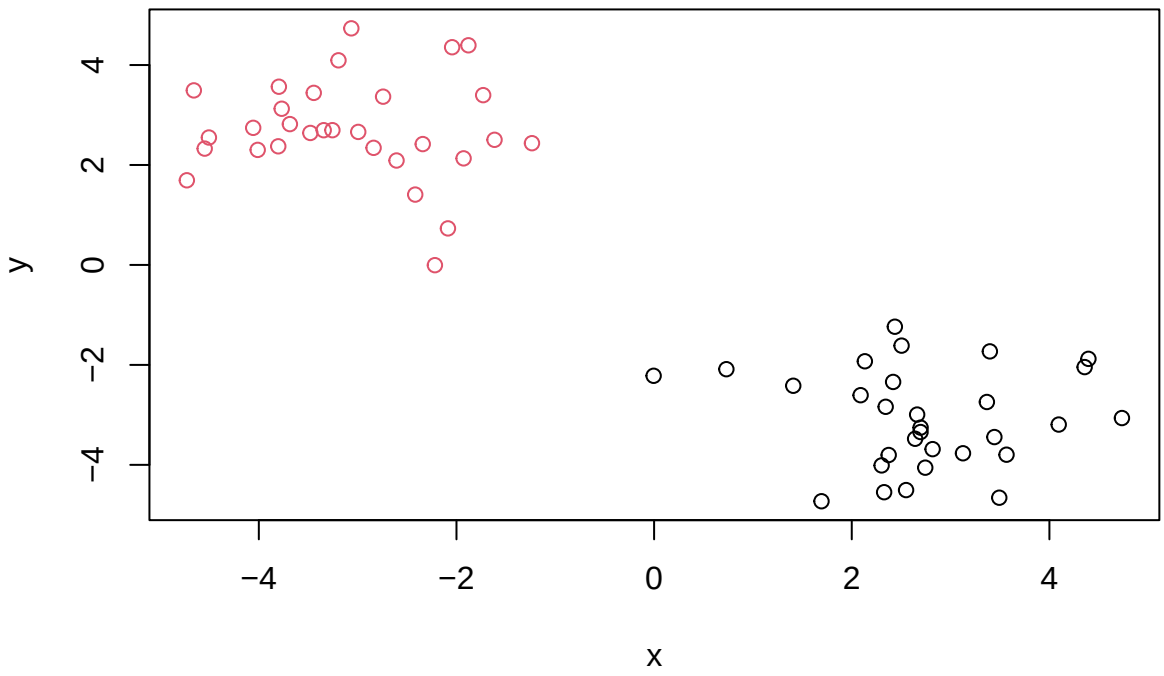
```
##           x         y
## 1  2.717717 -3.067167
## 2 -3.067167  2.717717
```

What is the cluster vector?

```r
kclusters<- k$cluster
kclusters
```

```
##  [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 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
```

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



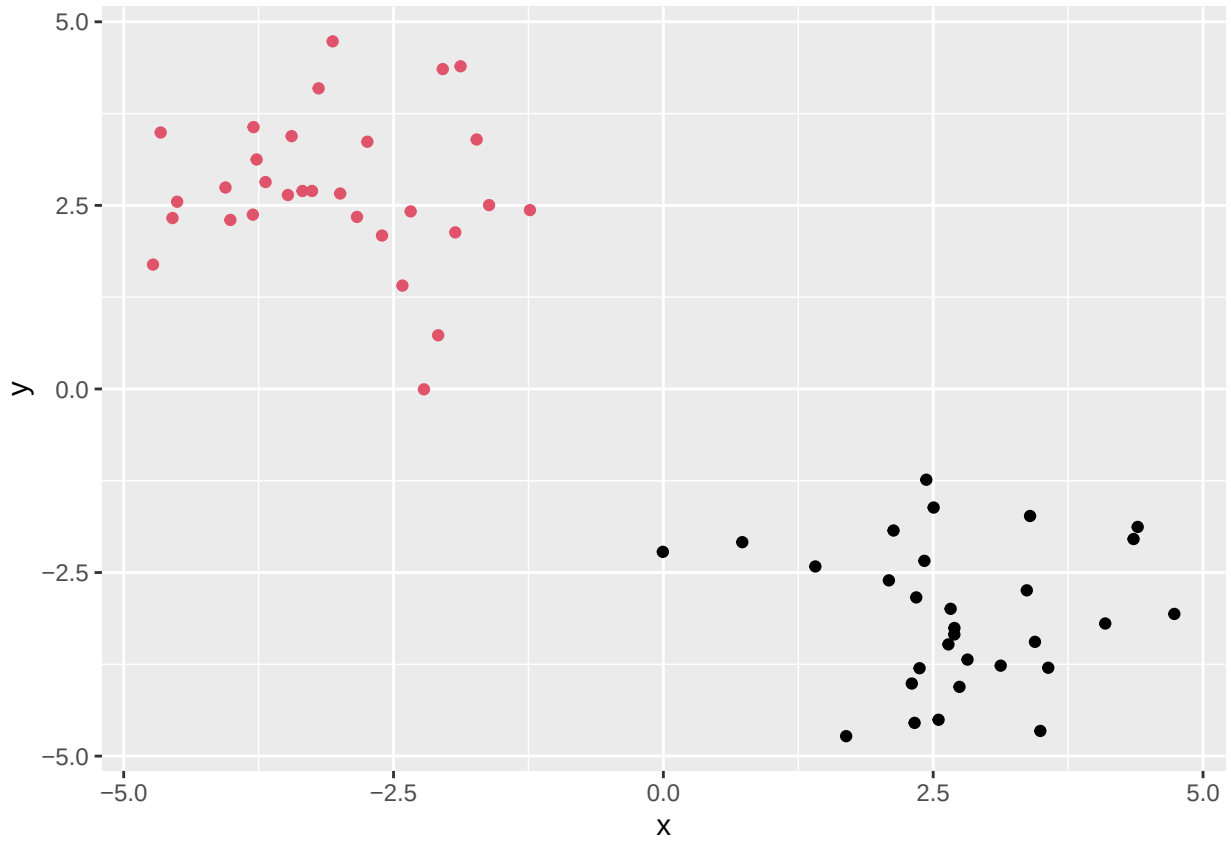
Recall that R has recycling, so we must be careful

```
y<-1:5
cbind(y,"red")
```

```
##      y
## [1,] "1" "red"
## [2,] "2" "red"
## [3,] "3" "red"
## [4,] "4" "red"
## [5,] "5" "red"
```

#GG plot version

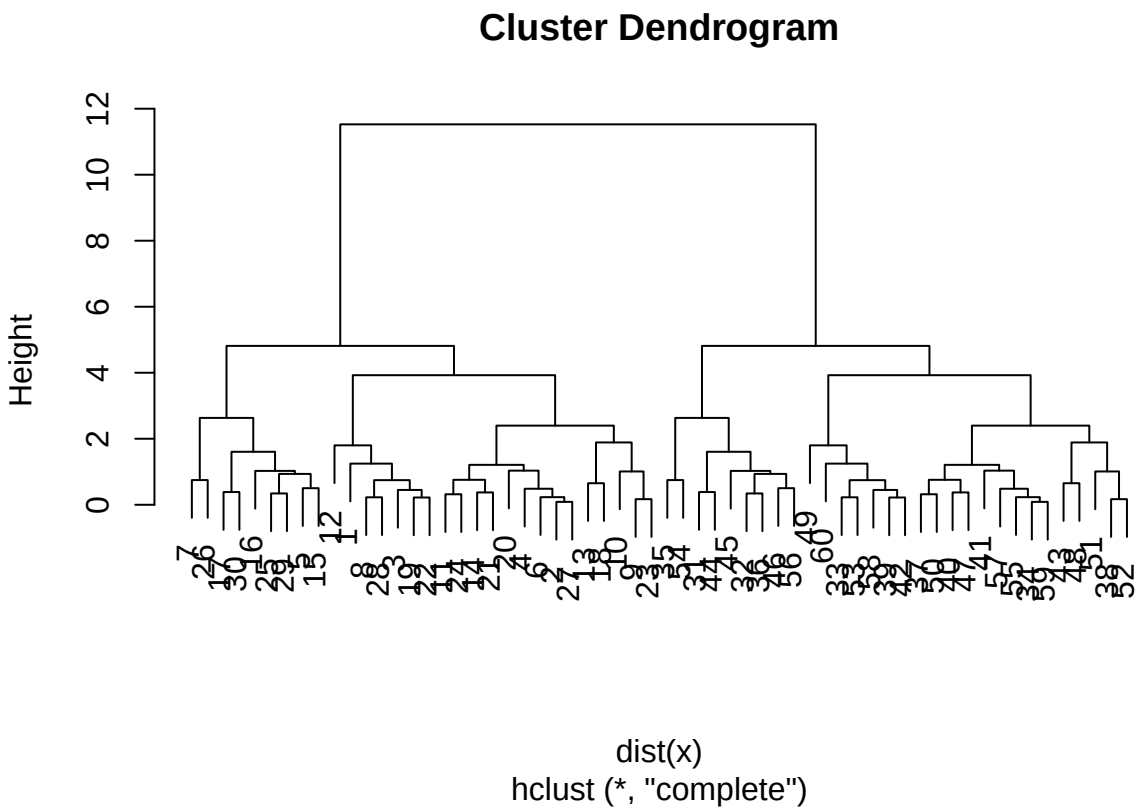
```
library(ggplot2)
ggplot(data.frame(x),aes(x,y))+geom_point(color=kclusters)
```



#Hierchical Clustering

Must give hclust a distance matrix as in put, not the raw data. hclust objects have a custom plot function, producing a cluster dendrogram.

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

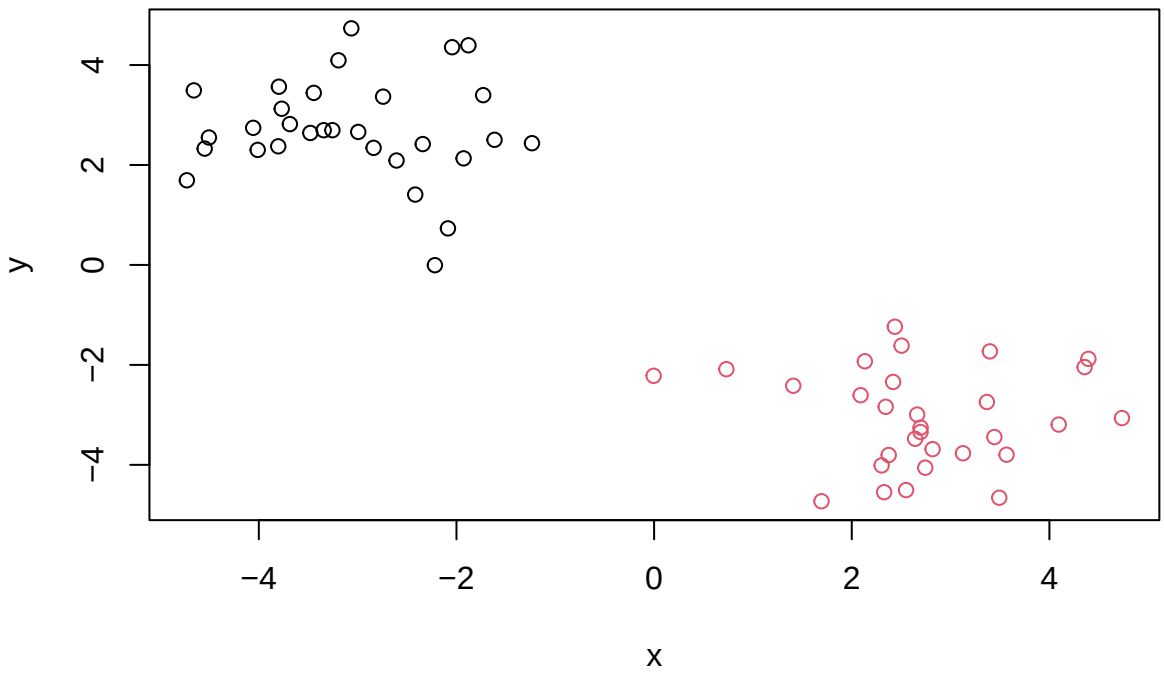



```
#abline(h=10)
```

Now we “cut” the tree to get our clusters. The function to do so is cutree()

```
ct<- cutree(hc, h=10)

plot(x, col=ct)
```



PCA

PCA is a useful method for analyzing large multidimensional datasets

Import data set of UK foods

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

Let's look at x:

```
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
```

```
nrow(x)
```

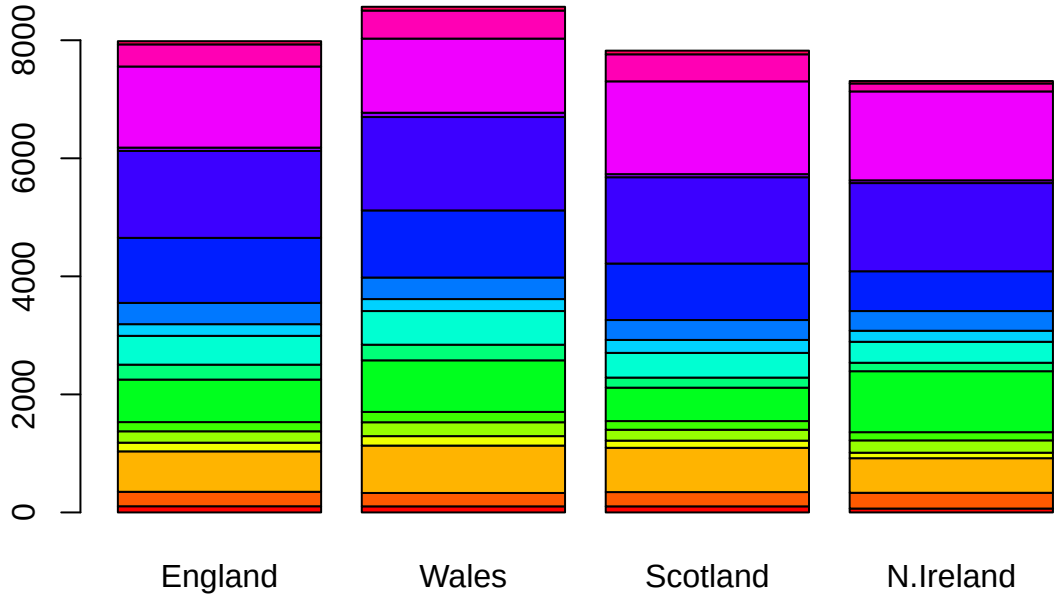
```
## [1] 17
```

```
ncol(x)
```

```
## [1] 4
```

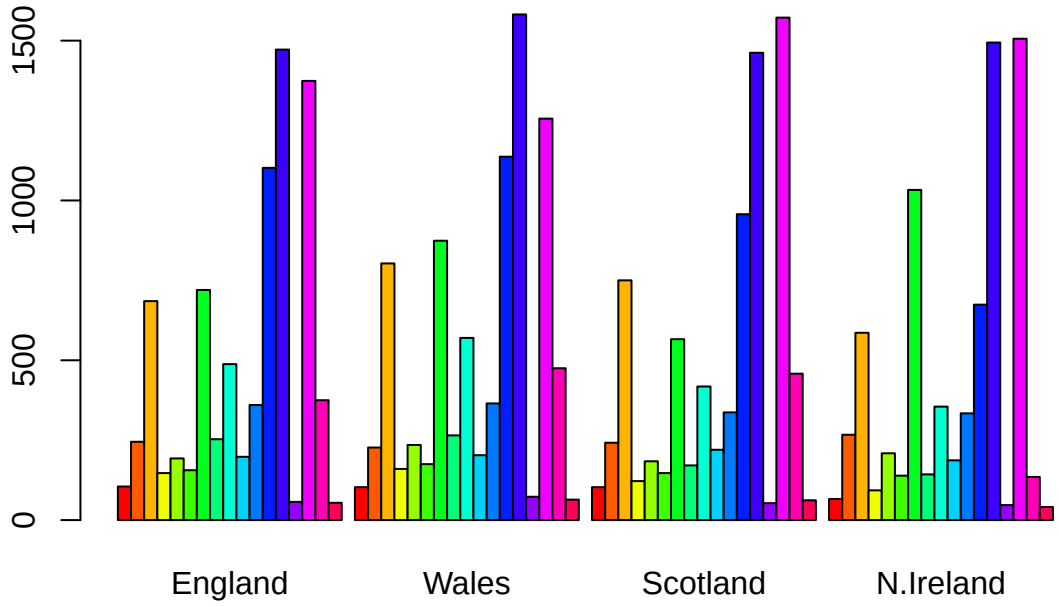
We could plot this with a barplot

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



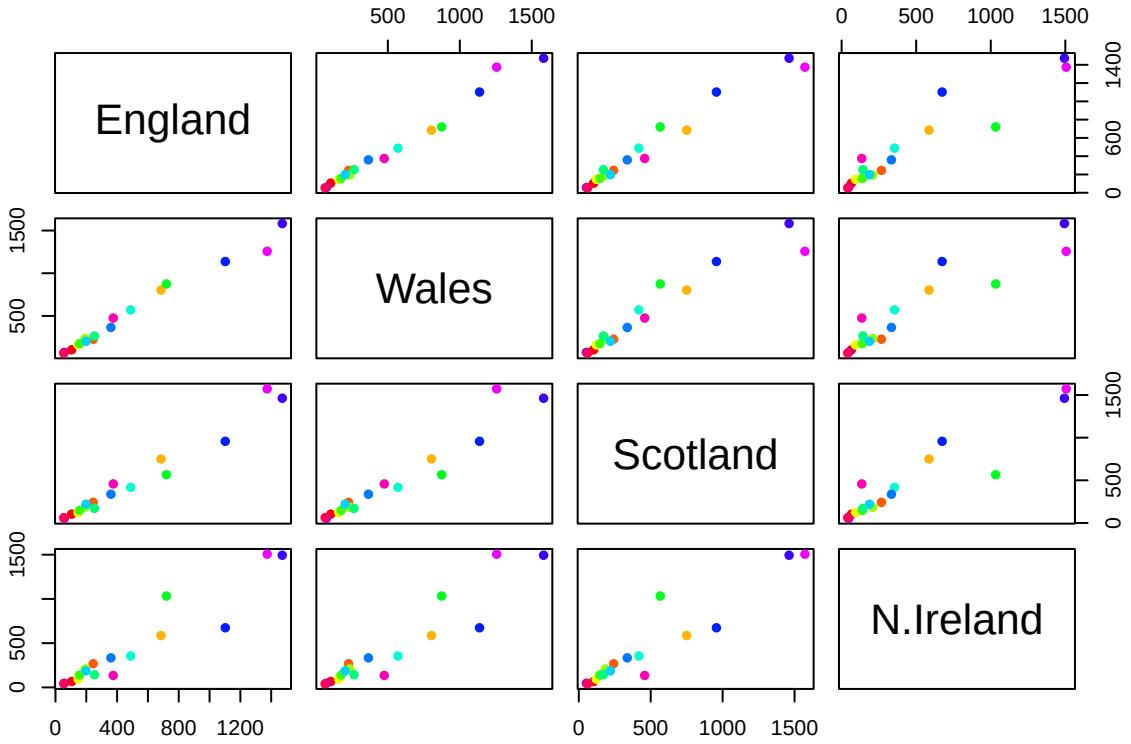
We can unstack with 'besides=TRUE'

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



One plot that is useful is a pairs plot. This produces a matrix of all pairwise correlations.

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



PCA to the rescue

What does PCA tell us about this data?

The main PCA function in base R is called 'prcomp()'

```
# Use the prcomp() PCA function

# note that prcomp rquires transpose of our matrix
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
```

a plot of PC1 vs PC2 is often called a PCA plot or "score plot"

```
attributes(pca)
```

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

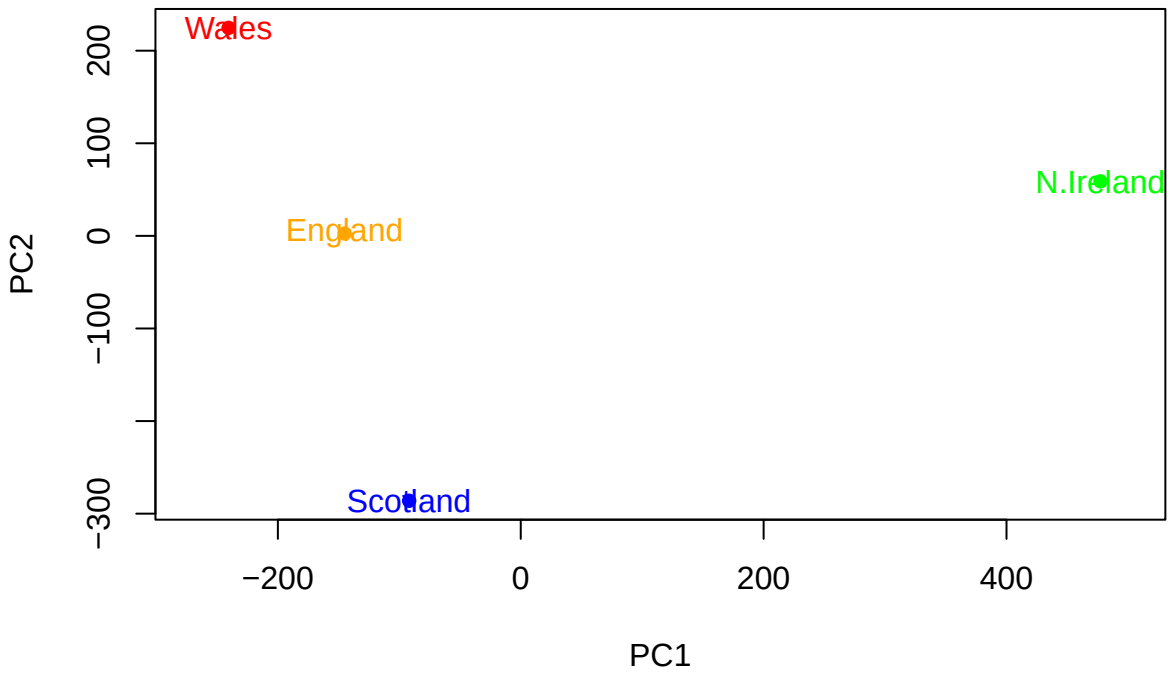
```
##
## $class
## [1] "prcomp"
```

To generate our score plot, we want pca$x component of the resulting object

```
pca$x
```

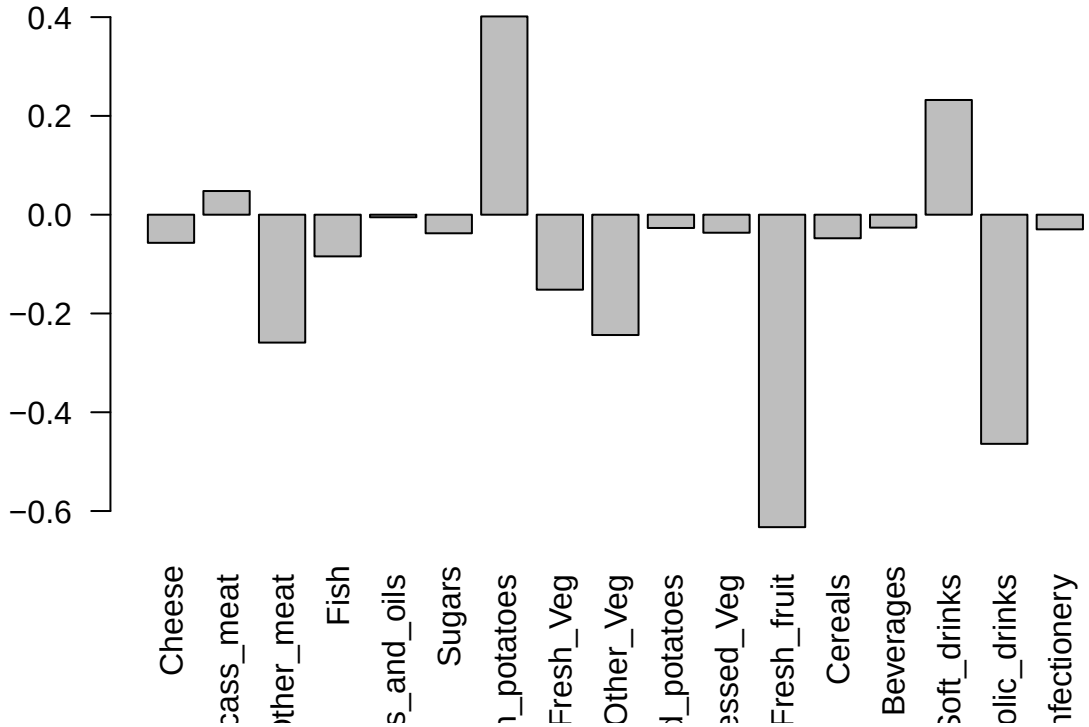
```
##                  PC1         PC2         PC3           PC4
## England   -144.99315    2.532999 -105.768945  2.842865e-14
## Wales     -240.52915  224.646925   56.475555  7.804382e-13
## Scotland   -91.86934 -286.081786   44.415495 -9.614462e-13
## N.Ireland  477.39164   58.901862    4.877895 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

```
colorv<- c("orange", "red", "blue", "green")
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500), col=colorv, pch=16)
text(pca$x[,1], pca$x[,2], colnames(x), col=colorv)
```



The loadings(aka weights) tell us how the original variables contribute to the PCs

```
barplot(pca$rotation[,1], las=2)
```



#RNA Seq Dataset