Machine Learning

Pham Vo

2/13/2022

Clustering with kmeans and hclust()

We will begin by making up some data to cluster. shortcut to insert code chunk: option+command+i

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

```
##               x          y
##  [1,]  4.2347574 -4.9365848
##  [2,]  3.3596015 -3.0430727
##  [3,]  4.3189254 -1.8155747
##  [4,]  2.3577094 -3.9550889
##  [5,]  3.4140226 -3.1392119
##  [6,]  3.4825113 -3.4872698
##  [7,]  3.6417728 -2.1300113
##  [8,]  2.8425338 -3.5283979
##  [9,]  3.9547135 -2.5499144
## [10,]  2.6801855 -3.4023781
## [11,]  3.2225554 -3.7673758
## [12,]  2.6247568 -4.4102463
## [13,]  1.4888526 -3.8175338
## [14,]  3.6314556 -3.0617546
## [15,]  1.2445519 -3.8664872
## [16,]  4.0780839 -1.2648538
## [17,]  2.9736588 -1.6253573
## [18,]  2.8033944 -3.5290531
## [19,]  3.5371391 -0.7760335
## [20,]  3.3591323 -3.6637642
## [21,]  3.7945046 -4.1831013
## [22,]  2.7439267 -3.8402006
## [23,]  4.4674454 -4.2069651
## [24,]  2.4958294 -4.6194852
## [25,]  2.6650930 -2.2922960
## [26,]  2.1556406 -3.7717725
## [27,]  3.4293102 -3.0912247
## [28,]  3.9995536 -1.0218711
## [29,]  2.3447089 -0.4673843
## [30,]  1.8751953 -1.5395258
## [31,] -1.5395258  1.8751953
## [32,] -0.4673843  2.3447089
## [33,] -1.0218711  3.9995536
## [34,] -3.0912247  3.4293102
## [35,] -3.7717725  2.1556406
## [36,] -2.2922960  2.6650930
## [37,] -4.6194852  2.4958294
## [38,] -4.2069651  4.4674454
## [39,] -3.8402006  2.7439267
## [40,] -4.1831013  3.7945046
## [41,] -3.6637642  3.3591323
## [42,] -0.7760335  3.5371391
## [43,] -3.5290531  2.8033944
## [44,] -1.6253573  2.9736588
## [45,] -1.2648538  4.0780839
## [46,] -3.8664872  1.2445519
## [47,] -3.0617546  3.6314556
## [48,] -3.8175338  1.4888526
## [49,] -4.4102463  2.6247568
## [50,] -3.7673758  3.2225554
## [51,] -3.4023781  2.6801855
## [52,] -2.5499144  3.9547135
## [53,] -3.5283979  2.8425338
## [54,] -2.1300113  3.6417728
## [55,] -3.4872698  3.4825113
## [56,] -3.1392119  3.4140226
## [57,] -3.9550889  2.3577094
## [58,] -1.8155747  4.3189254
## [59,] -3.0430727  3.3596015
## [60,] -4.9365848  4.2347574
```

Run kmeans()

```
k <- kmeans(x, centers=2, nstart=20)
k
```

```
## K-means clustering with 2 clusters of sizes 30, 30
##
## Cluster means:
##           x         y
## 1 -3.026793  3.107384
## 2  3.107384 -3.026793
##
## 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 1 1
##
## Within cluster sum of squares by cluster:
## [1] 61.71633 61.71633
##  (between_SS / total_SS =  90.1 %)
##
## Available components:
##
```

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

Q. what size is each cluster?

```
k$size
```

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

Q. Cluster centers

```
k$centers
```

```
##           x         y
## 1 -3.026793  3.107384
## 2  3.107384 -3.026793
```

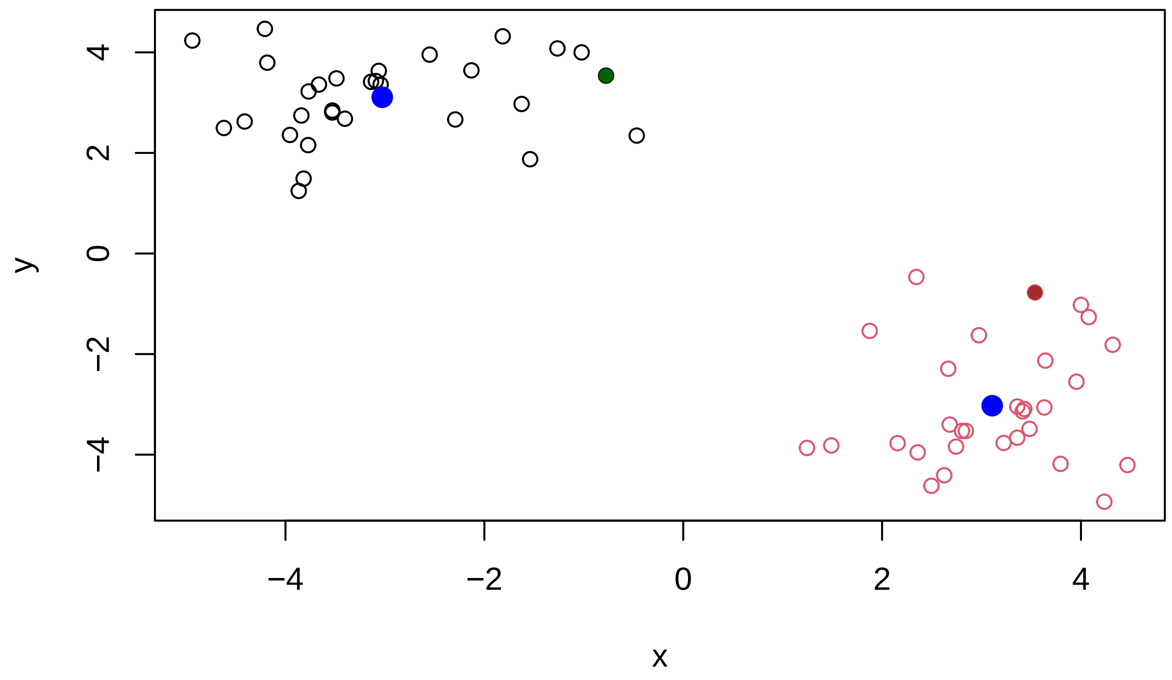
Q. Membership vector

```
k$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 1 1
```

Plot our data with the clustering result

```
plot(x, col=k$cluster)
points(k$centers, col="blue", pch=16, cex=1.5)
points(x[42,1], x[42,2], col="darkgreen", pch = 16)
points(x[19,1], x[19,2], col="brown", pch = 16)
```



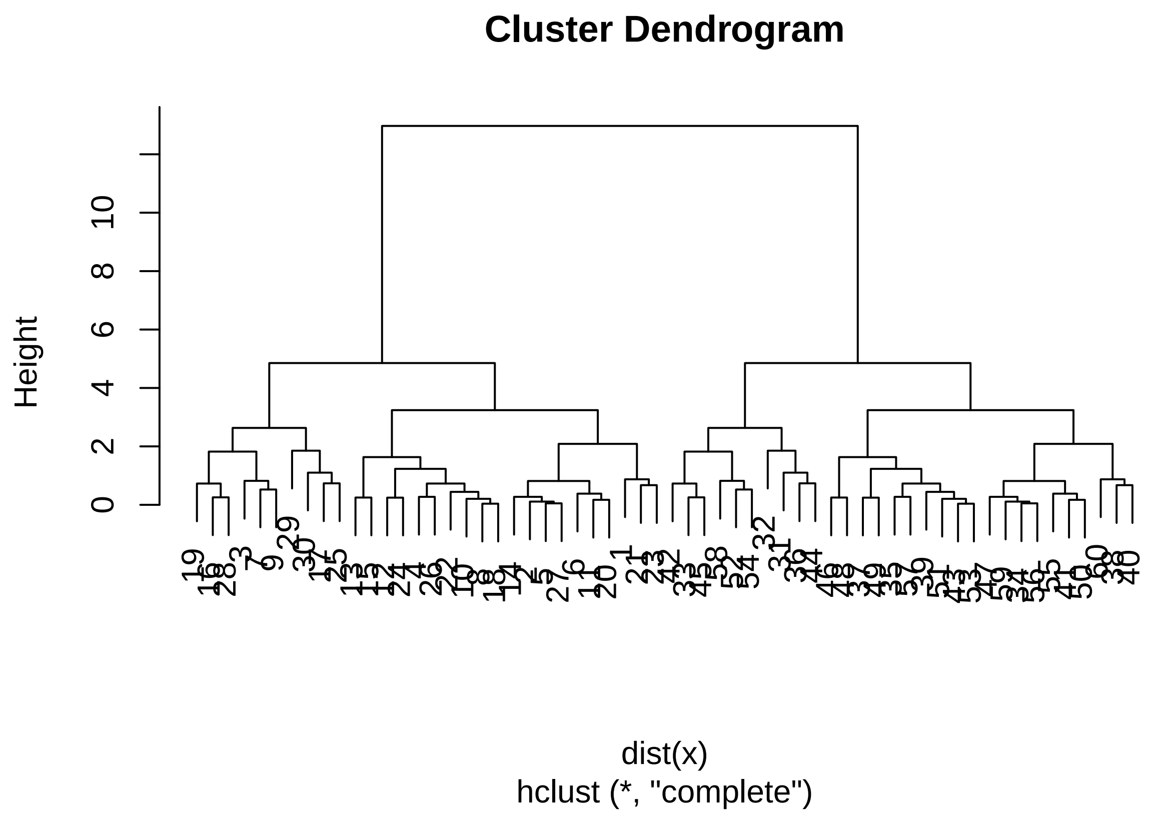
hclust()

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

```
##
## Call:
## hclust(d = dist(x))
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 60
```

There is a cool and useful plot method for hclust()

```
plot(hc)
```



Principal Component Analysis

Data import

```r
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
```

```
ncol(x)
```

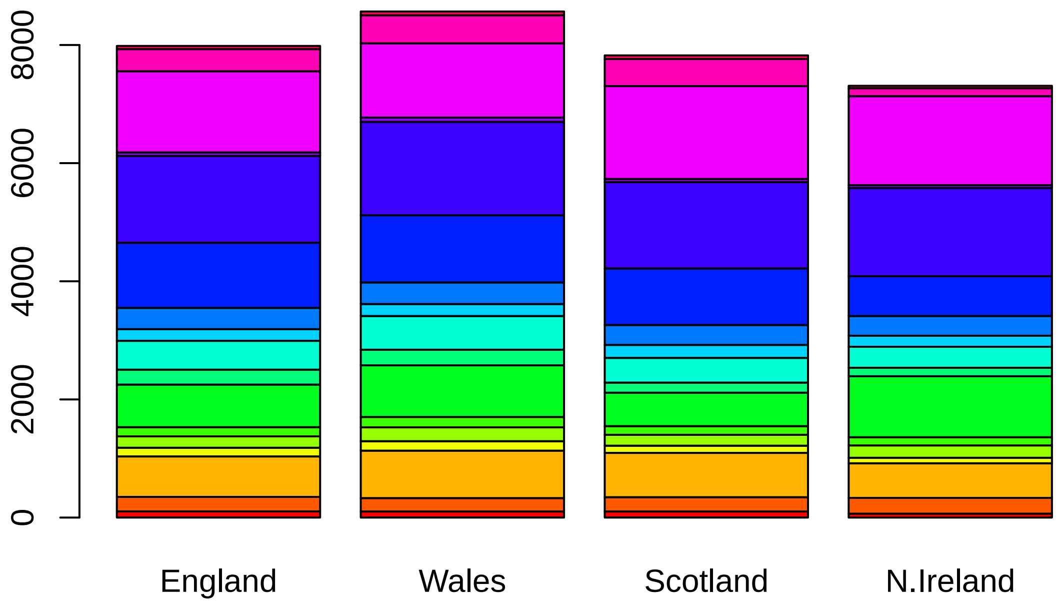
```
## [1] 4
```

```
nrow(x)
```

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

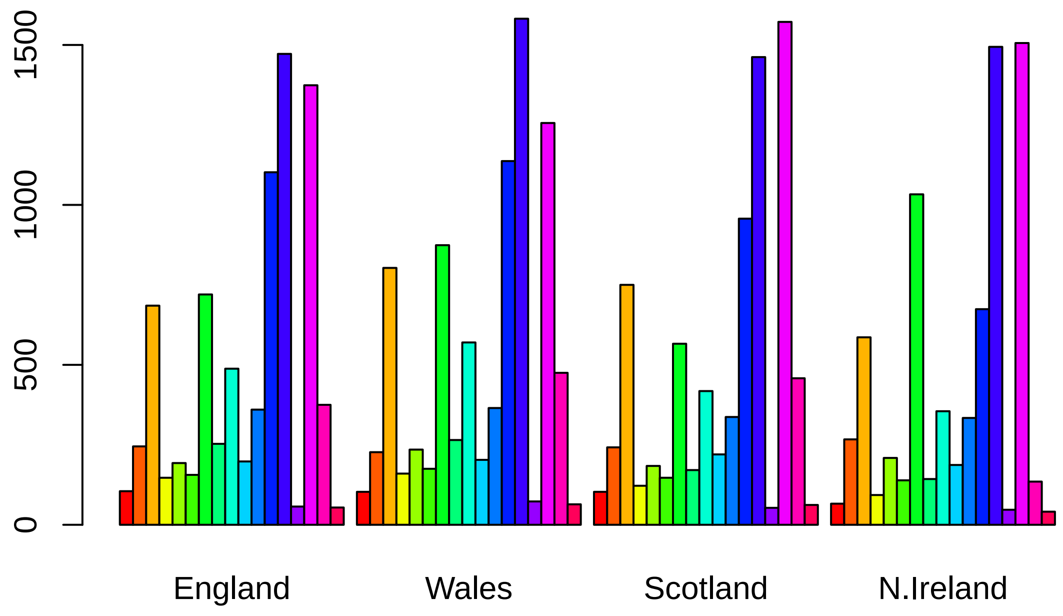
Let's make some typical plots

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



Make it side by side i.e. not stacked

```r
barplot(as.matrix(x), col=mycols, beside=TRUE)
```

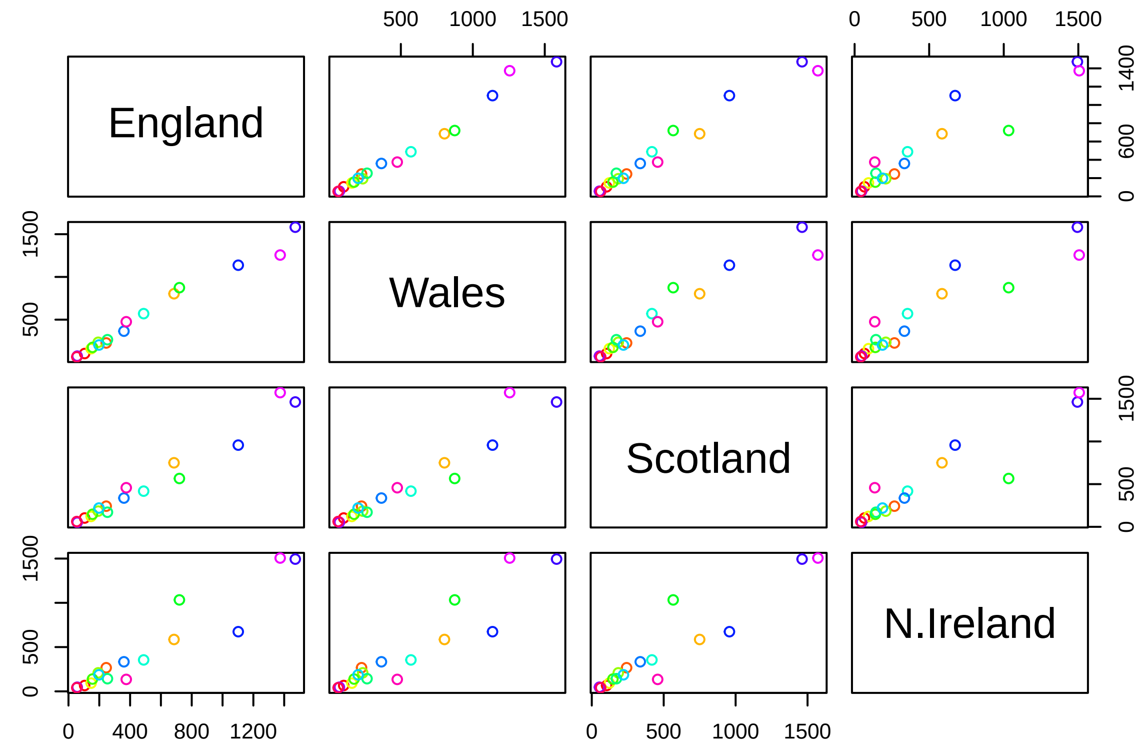


One plot that is helpful here.

#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?

No difference in values between x-axis and y-axis.

```r
pairs(x, col=mycols)
```



#Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set?

PCA to rescue

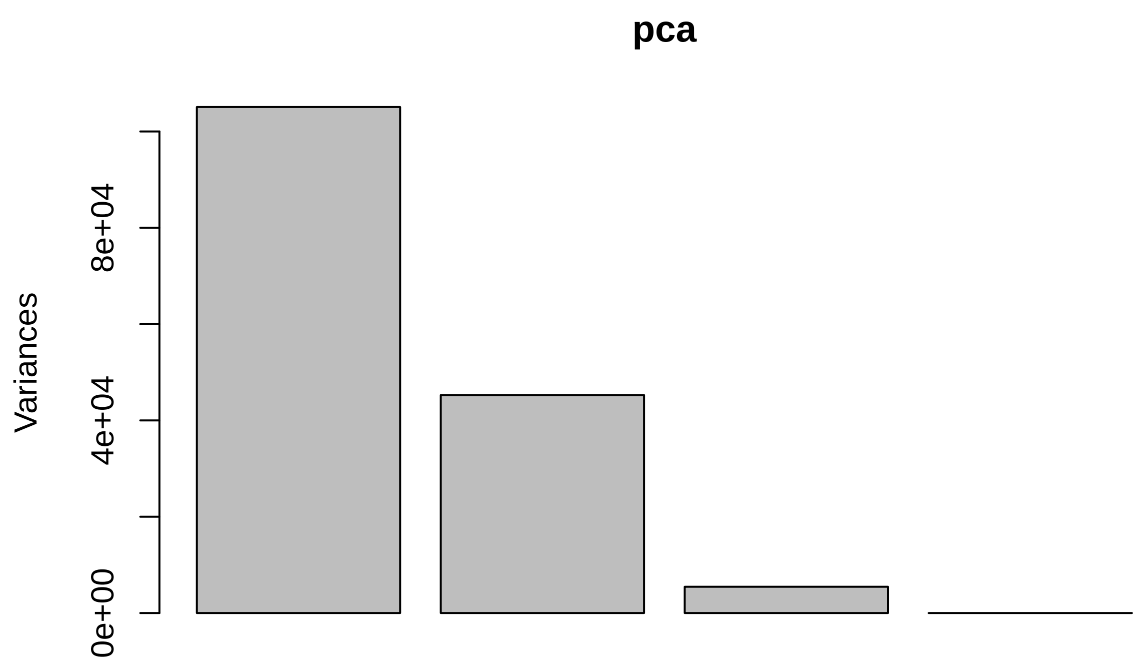
Do PCR of this 17D food data. The main function in base R is called "prcom()". This function requires the transpose of our data in this case…

```
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
```

The prcom() function returns a list object.

```
plot(pca)
```



The PCA plot a.k.a a PCA score plot is a plot of PC1 vs PV2. Basically using the new PCA axis to view our data.

```
attributes(pca)
```

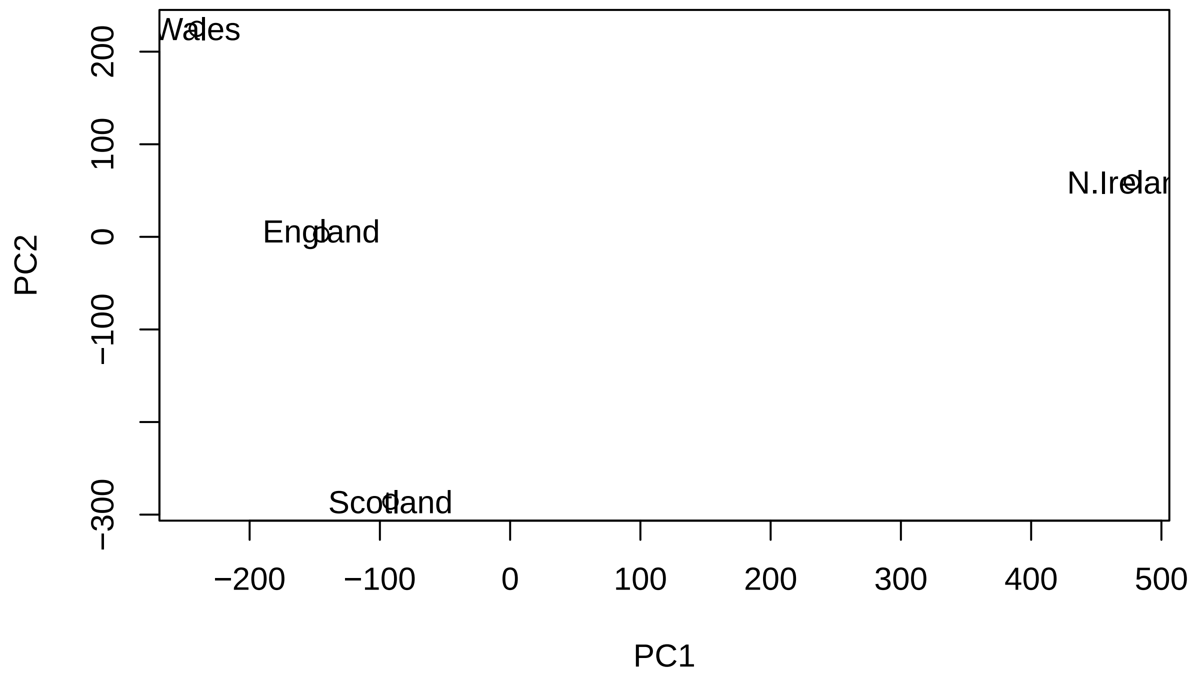
```
## $names
## [1] "sdev"     "rotation" "center"   "scale"    "x"
##
## $class
## [1] "prcomp"
```

We will focus on “pca$x” for this plot

```
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], xlab= "PC1", ylab="PC2")
text(pca$x[,1], pca$x[,2], labels = colnames(x))
```



## PCA of a RNA-Seq

```
url2 <- "https://tinyurl.com/expression-CSV"
rna.data <- read.csv(url2, row.names=1)
head(rna.data)
```

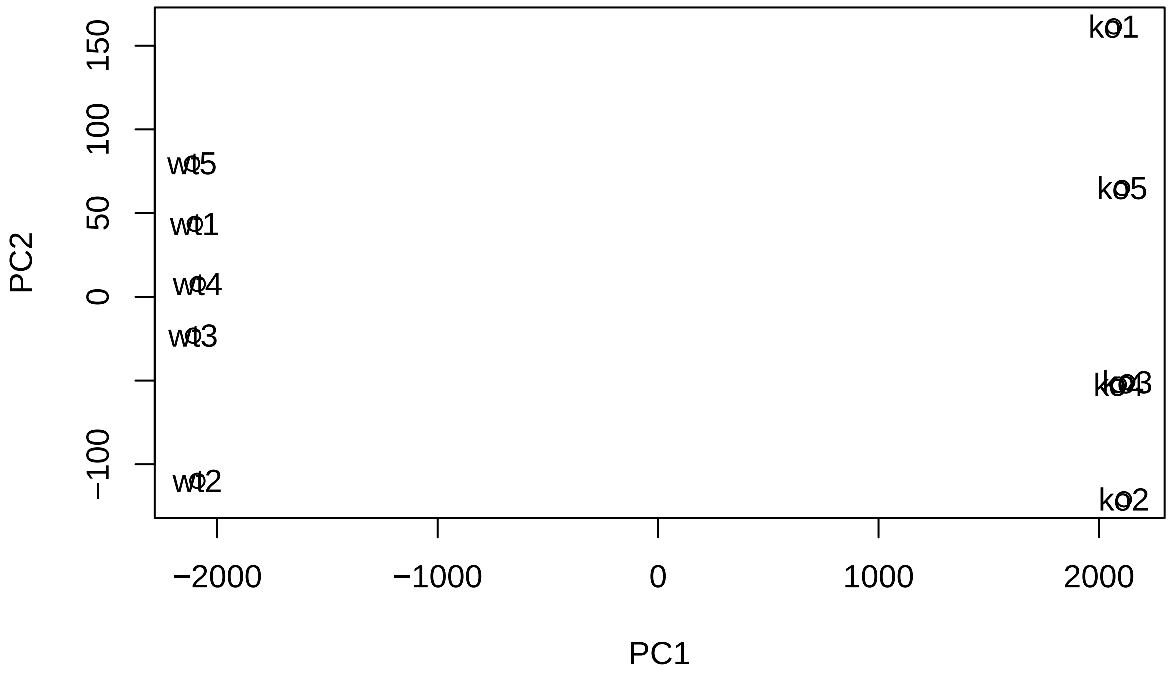
```
##        wt1 wt2  wt3  wt4 wt5 ko1 ko2 ko3 ko4 ko5
## gene1  439 458  408  429 420  90  88  86  90  93
## gene2  219 200  204  210 187 427 423 434 433 426
## gene3 1006 989 1030 1017 973 252 237 238 226 210
## gene4  783 792  829  856 760 849 856 835 885 894
## gene5  181 249  204  244 225 277 305 272 270 279
## gene6  460 502  491  491 493 612 594 577 618 638
```

```
pca <- prcomp(t(rna.data))
summary(pca)
```

```
## Importance of components:
##                              PC1     PC2      PC3      PC4      PC5      PC6
## Standard deviation     2214.2633 88.9209 84.33908 77.74094 69.66341 67.78516
## Proportion of Variance    0.9917  0.0016  0.00144  0.00122  0.00098  0.00093
## Cumulative Proportion     0.9917  0.9933  0.99471  0.99593  0.99691  0.99784
##                             PC7      PC8      PC9      PC10
## Standard deviation     65.29428 59.90981 53.20803 3.142e-13
## Proportion of Variance  0.00086  0.00073  0.00057 0.000e+00
## Cumulative Proportion   0.99870  0.99943  1.00000 1.000e+00
```

Make our PCA score plot

```
plot(pca$x[,1:2])
text(pca$x[,1:2], labels = colnames(rna.data))
```


ko1
wt5
ko5
wt1
wt4
wt3
ko3
ko4
wt2
ko2
PC1
PC2