Class07: Machine Learning 1

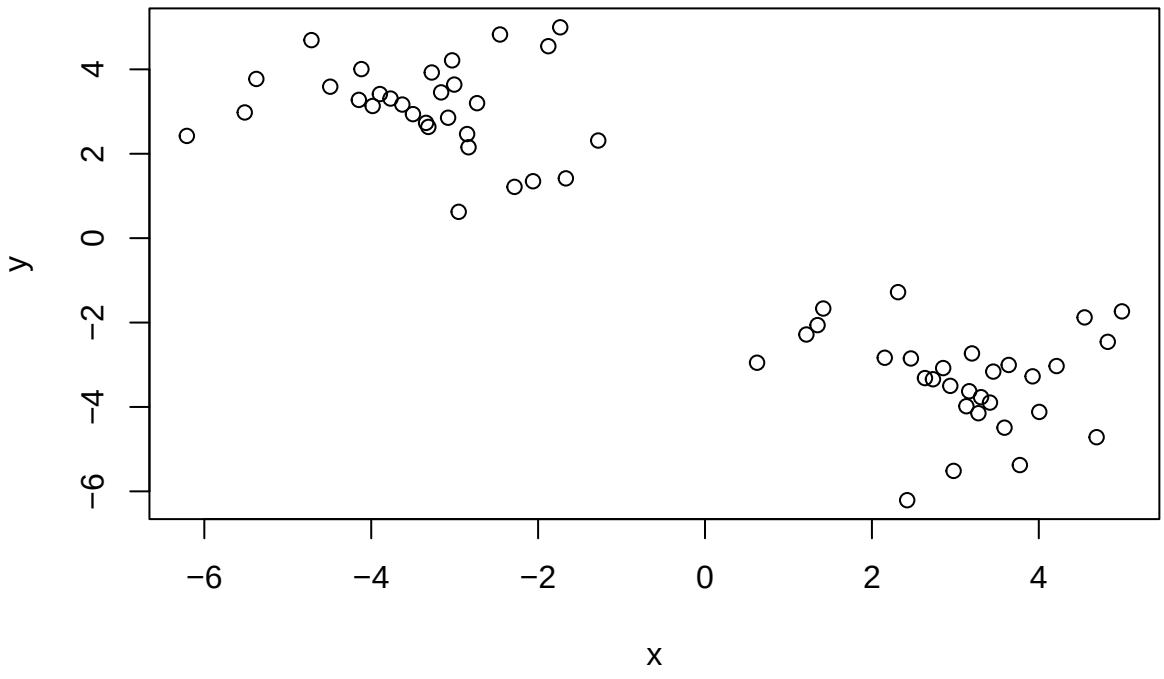
Karis Kim (A16378938)

2024-04-23

First Up kmeans()

Demo of using kmeans() function in base R. First we will make up data with a known structure.

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



Now we have some made up data in `x` let's see how kmeans works with this data.

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

```
## K-means clustering with 2 clusters of sizes 30, 30
##
## Cluster means:
##           x         y
## 1 -3.342890  3.108644
## 2  3.108644 -3.342890
##
## 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 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] 72.88392 72.88392
##  (between_SS / total_SS =  89.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
k$size
```

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

Q. How do we go to the cluster membership/assignment?

```
k$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
```

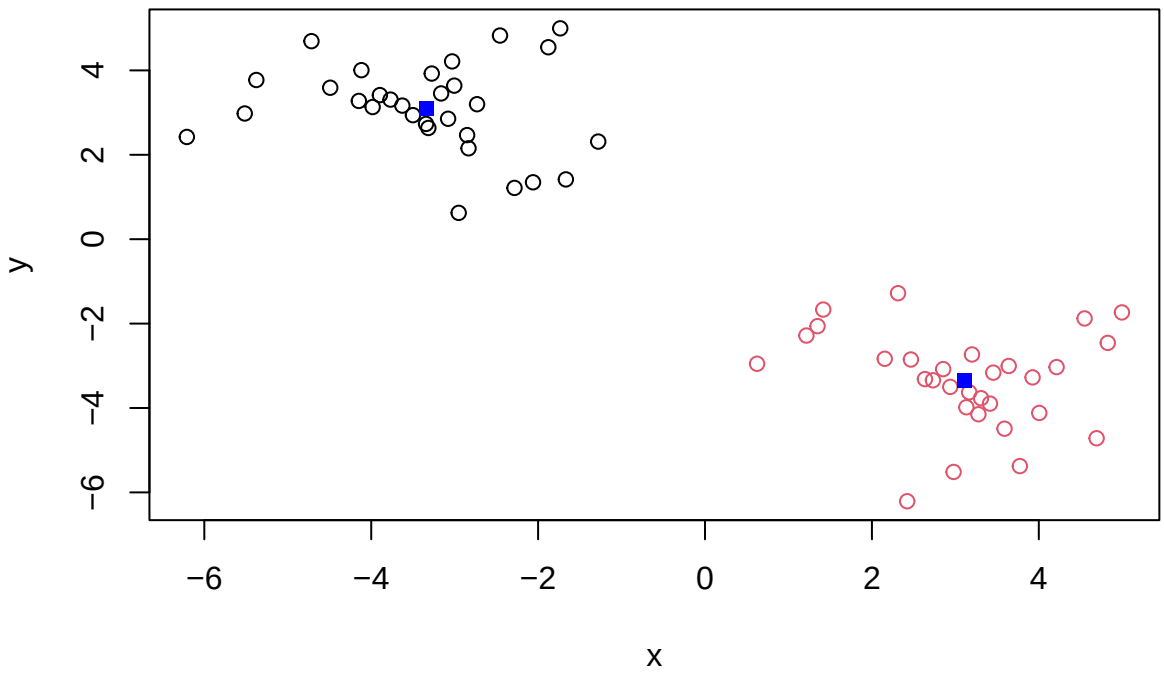
Q. What about the cluster centers?

```
k$centers
```

```
##           x         y
## 1 -3.342890  3.108644
## 2  3.108644 -3.342890
```

Now we go to the main results and use them to plot our data with the kmeans result.

```
plot(x, col=k$cluster)
points(k$centers, col="blue", pch=15)
```



Now for Hierarchical Clustering

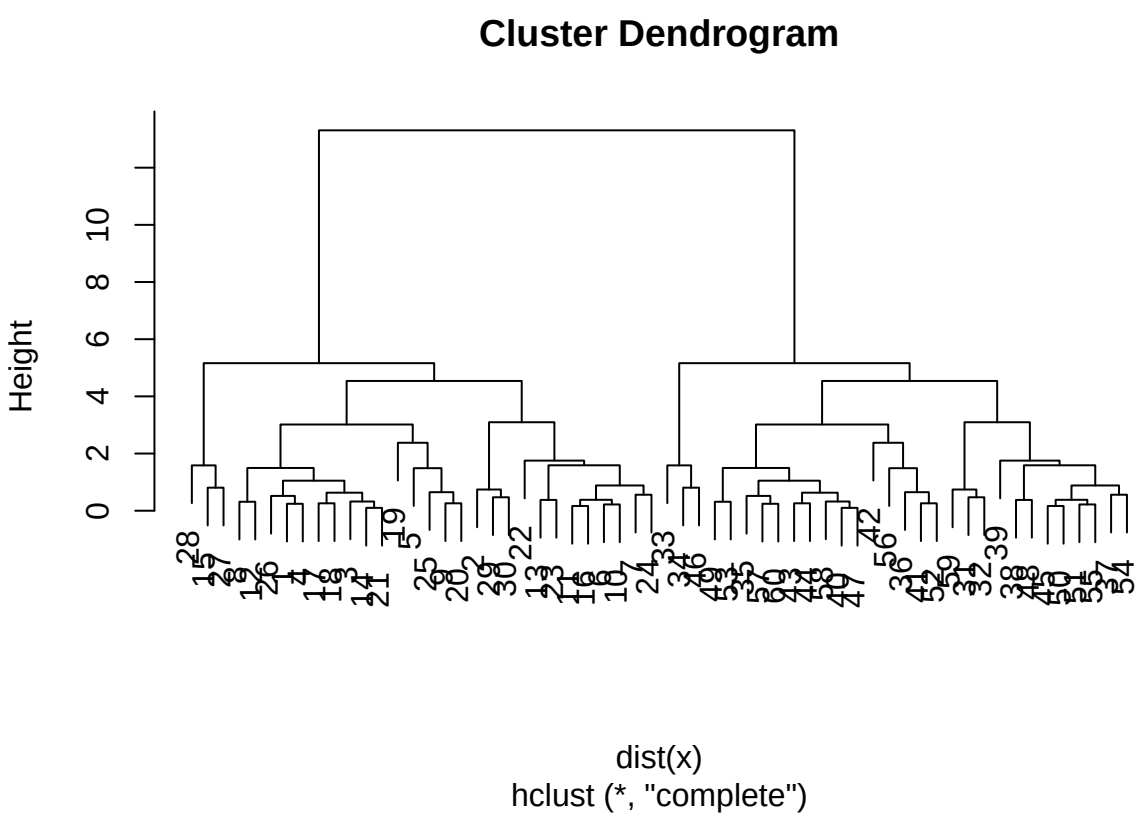
We will cluster the same data `x` with the `hclust()`. In this case `hclust()` requires a distance matrix as input.

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

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

Let's plot our hclust result

```
plot(hc)
```



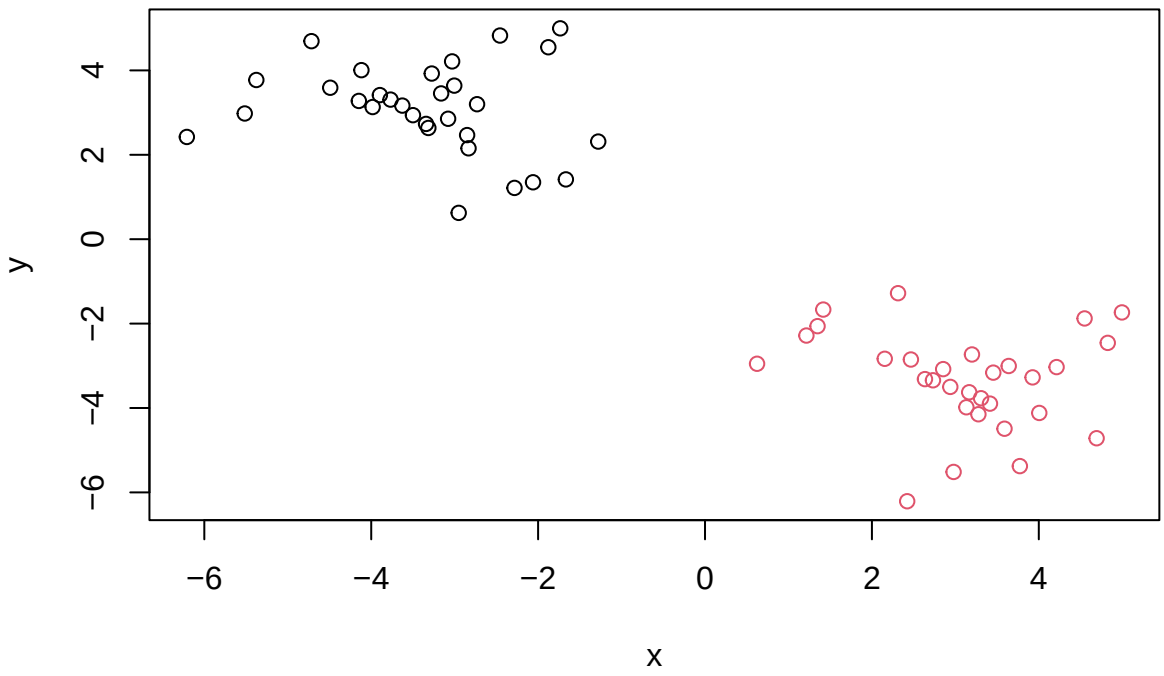
To get our cluster membership vector, we need to "cut" the tree with the `cutree()`

```
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 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
```

Now plot our data with the hclust() results.

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



Principal Component Analysis (PCA)

PCA of UK Food Data

Read data from the website and try a few visualizations.

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url)
```

```
nrow(x)
```

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

```
ncol(x)
```

```
## [1] 5
```

Checking your data. It is always a good idea to examine your imported data to make sure it meets your expectations.

```
#To view the entire data frame
View(x)
#To view the first 6 rows of the data frame
head(x)
```

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

```
#To view the last 6 rows of the data frame
tail(x)
```

```
##                    X England Wales Scotland N.Ireland
## 12      Fresh_fruit    1102  1137      957       674
## 13          Cereals    1472  1582     1462      1494
## 14         Beverages      57    73       53        47
## 15      Soft_drinks    1374  1256     1572      1506
## 16 Alcoholic_drinks     375   475      458       135
## 17    Confectionery      54    64       62        41
```

It appears that the data is not set properly, as the first column is labeled as 'X', giving us 5 variables not 4. To fix this we use the function rownames().

```
#To class for the first column
rownames(x) <- x[,1]
#To remove the first column
x <- x[,-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
```

Another way to do it is by calling read.csv()

x <- read.csv(url, row.names=1) head(x)

```
#To find out the dimensions (x, y) of the data frame: dim()
dim(x)
```

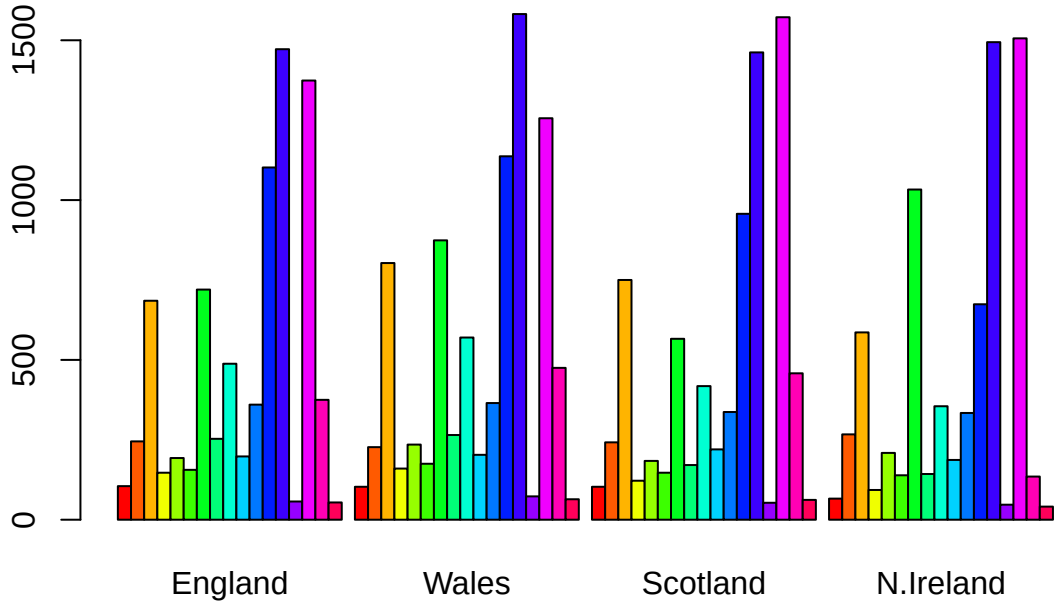
```
## [1] 17  4
```

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?

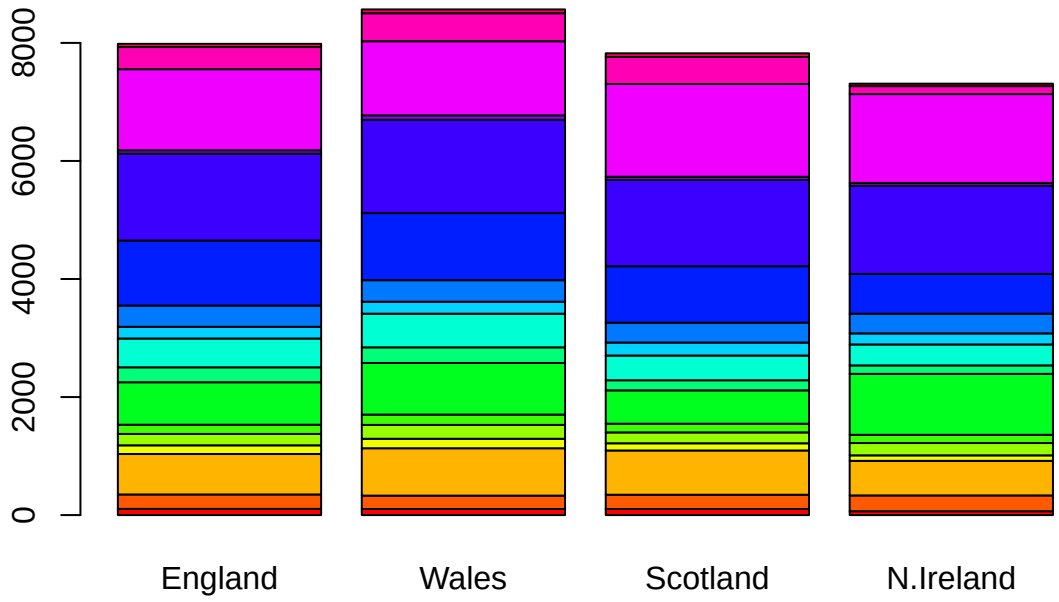
I think the solution for the 'row-names problem' that I prefer is the: 'x <- read.csv(url, row.names=1)' approach, as you have more control as to which column that you are changing. I think using the x[,-1] method would work if you only had to adjust the first column, because you might continue to erase more variables.

Spotting major differences and trends

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

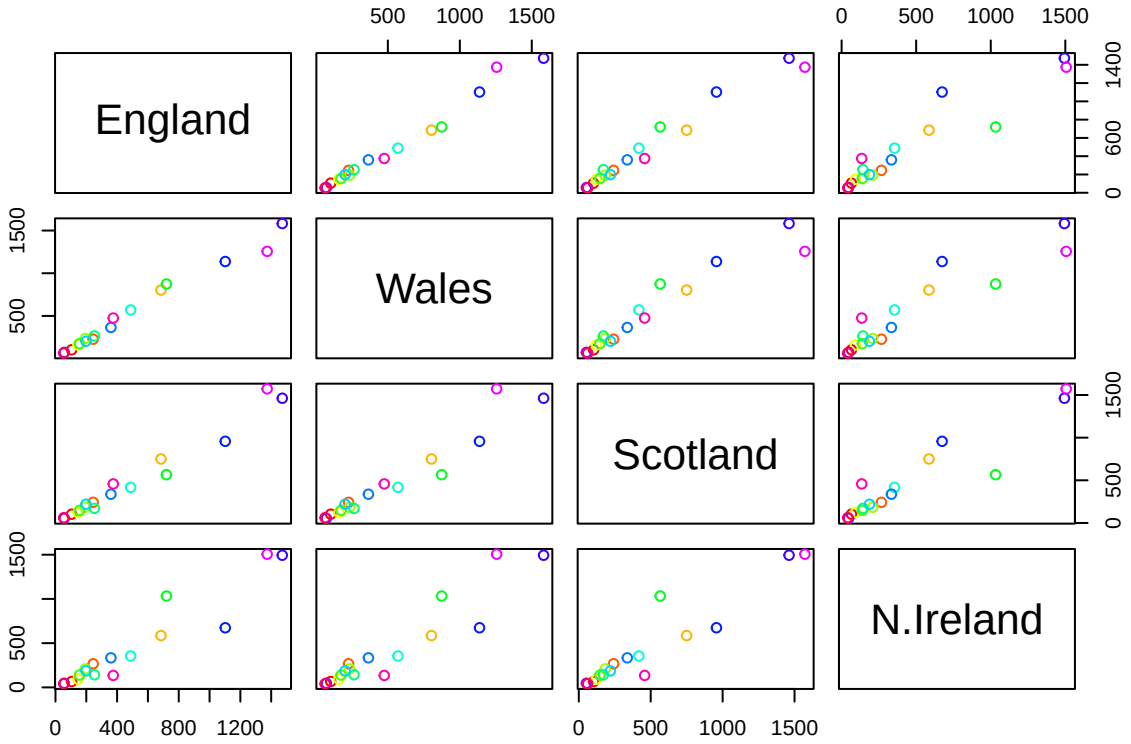



```
cols <- rainbow(nrow(x))
barplot( as.matrix(x), col=cols)
```




```
#It is hard to compare the data in this format.
```

```
pairs(x, col=cols)
```



PCA to the rescue! The main R PCA function is called `prcomp()`. We will need to give it the transpose of our input data.

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

```
## Standard deviations (1, .., p=4):
## [1] 3.241502e+02 2.127478e+02 7.387622e+01 3.175833e-14
##
## Rotation (n x k) = (17 x 4):
##                              PC1          PC2         PC3          PC4
## Cheese               -0.056955380  0.016012850  0.02394295 -0.694538519
## Carcass_meat          0.047927628  0.013915823  0.06367111  0.489884628
## Other_meat           -0.258916658 -0.015331138 -0.55384854  0.279023718
## Fish                 -0.084414983 -0.050754947  0.03906481 -0.008483145
## Fats_and_oils        -0.005193623 -0.095388656 -0.12522257  0.076097502
## Sugars               -0.037620983 -0.043021699 -0.03605745  0.034101334
## Fresh_potatoes        0.401402060 -0.715017078 -0.20668248 -0.090972715
## Fresh_Veg            -0.151849942 -0.144900268  0.21382237 -0.039901917
## Other_Veg            -0.243593729 -0.225450923 -0.05332841  0.016719075
## Processed_potatoes   -0.026886233  0.042850761 -0.07364902  0.030125166
## Processed_Veg        -0.036488269 -0.045451802  0.05289191 -0.013969507
## Fresh_fruit          -0.632640898 -0.177740743  0.40012865  0.184072217
## Cereals              -0.047702858 -0.212599678 -0.35884921  0.191926714
## Beverages            -0.026187756 -0.030560542 -0.04135860  0.004831876
## Soft_drinks           0.232244140  0.555124311 -0.16942648  0.103508492
```

```
## Alcoholic_drinks    -0.463968168  0.113536523 -0.49858320 -0.316290619
## Confectionery       -0.029650201  0.005949921 -0.05232164  0.001847469
```

```
#Like kmeans(), there are different attributes for prcomp()
attributes(pca)
```

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

```
#Example: calling the center values.
pca$center
```

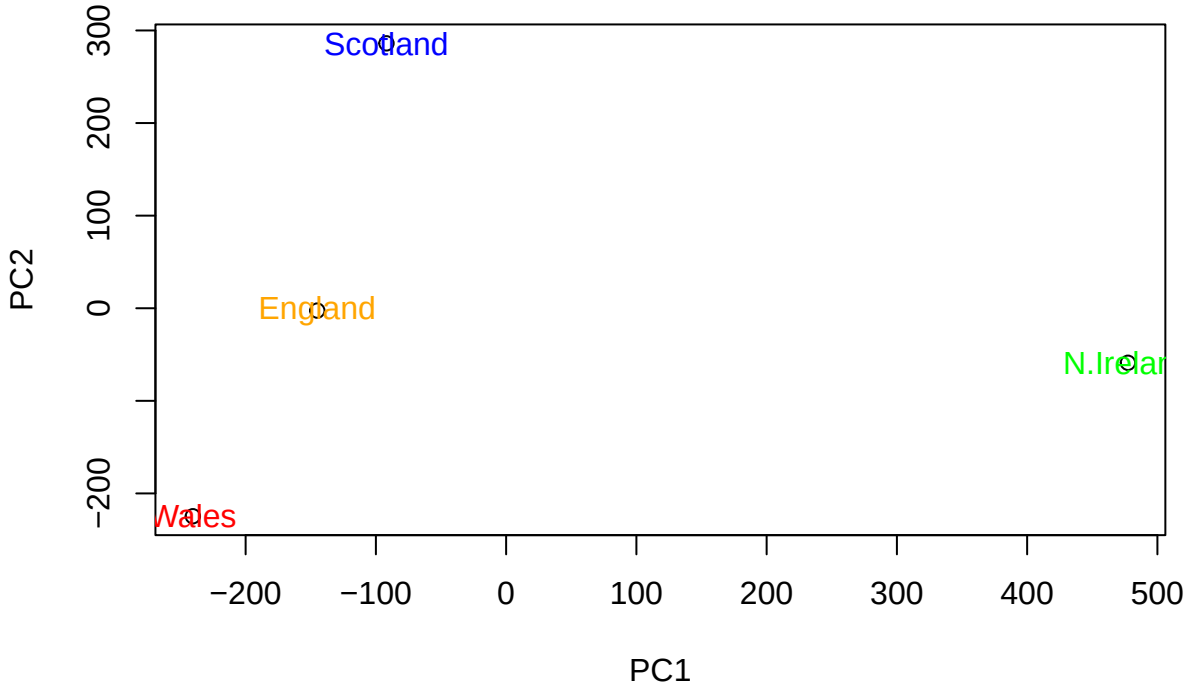
```
##              Cheese       Carcass_meat          Other_meat                Fish
##               94.25             245.25             706.00             130.50
##      Fats_and_oils              Sugars     Fresh_potatoes          Fresh_Veg
##              205.25             154.25             798.25             208.00
##          Other_Veg  Processed_potatoes       Processed_Veg         Fresh_fruit
##              457.75             202.00             349.00             967.50
##            Cereals            Beverages        Soft_drinks    Alcoholic_drinks
##             1502.50              57.50            1427.00             360.75
##      Confectionery
##               55.25
```

To make our new PCA plot (PCA score plot) we access `pca$x`

```
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
```

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



PCA of RNA-seq Data

First, we have to read our data.

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

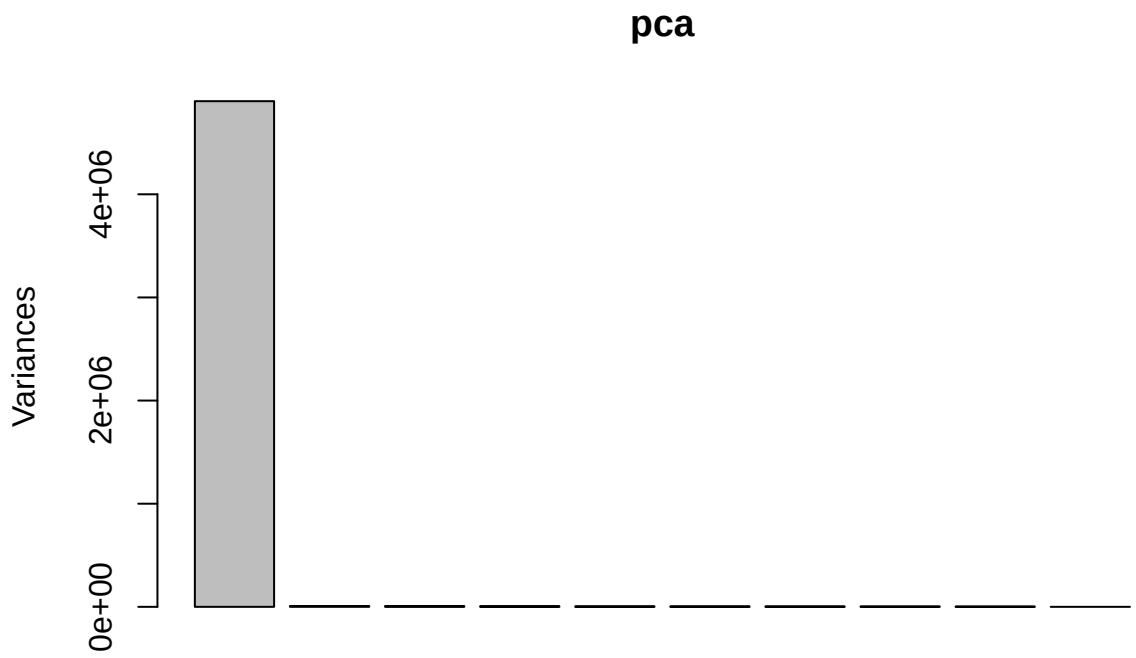
```
##        wt1 wt2  wt3  wt4 wt5 ko1 ko2 ko3 ko4 ko5
## gene1  439 458  408  429 420  90  88  86  90  93
## gene2  219 200  204  210 187 427 423 434 433 426
## gene3 1006 989 1030 1017 973 252 237 238 226 210
## gene4  783 792  829  856 760 849 856 835 885 894
## gene5  181 249  204  244 225 277 305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

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

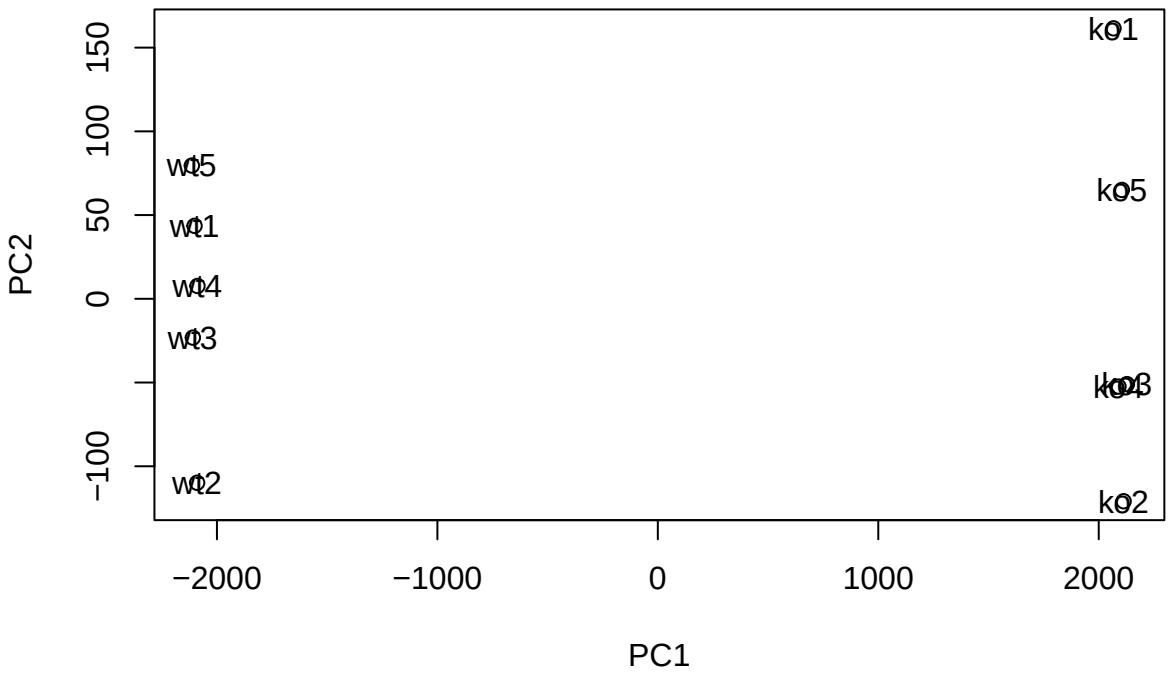
```
## Importance of components:
##                             PC1     PC2      PC3      PC4      PC5      PC6
## Standard deviation     2214.2633 88.9209 84.33908 77.74094 69.66341 67.78516
## Proportion of Variance    0.9917  0.0016  0.00144  0.00122  0.00098  0.00093
```

```
## 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 2.647e-13
## Proportion of Variance  0.00086  0.00073  0.00057 0.000e+00
## Cumulative Proportion   0.99870  0.99943  1.00000 1.000e+00
```

```r
plot(pca)
```




```r
plot(pca$x[,1], pca$x[,2], xlab = "PC1", ylab = "PC2")
text(pca$x[,1], pca$x[,2], colnames(rna.data))
```


PC2
150
100
50
0
−100
wt5
wt1
wt4
wt3
wt2
ko1
ko5
ko4
ko3
ko2
−2000
−1000
0
1000
2000
PC1