Class 7: Machine Learning 1

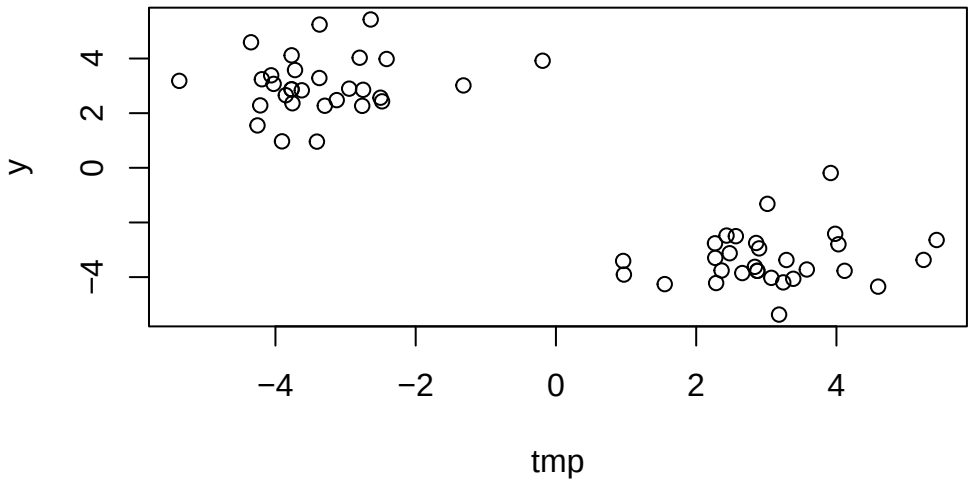
Safiya Sayd (A18027139)

2025-02-05

#kmeans()

Demo af using kmeans() function in base R. First make up some data with a known structure

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



Made up data in `x` lets see how kmeans works with this data

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

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

Cluster means:
        tmp         y
1 -3.334045  3.040037
2  3.040037 -3.334045

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 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

Within cluster sum of squares by cluster:
[1] 59.54232 59.54232
 (between_SS / total_SS =  91.1 %)

Available components:

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

Q How many points are in each cluster?

```r
k$size
```

```
[1] 30 30
```

Q. How do we find which cluster the data point belongs to?

```r
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 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
```

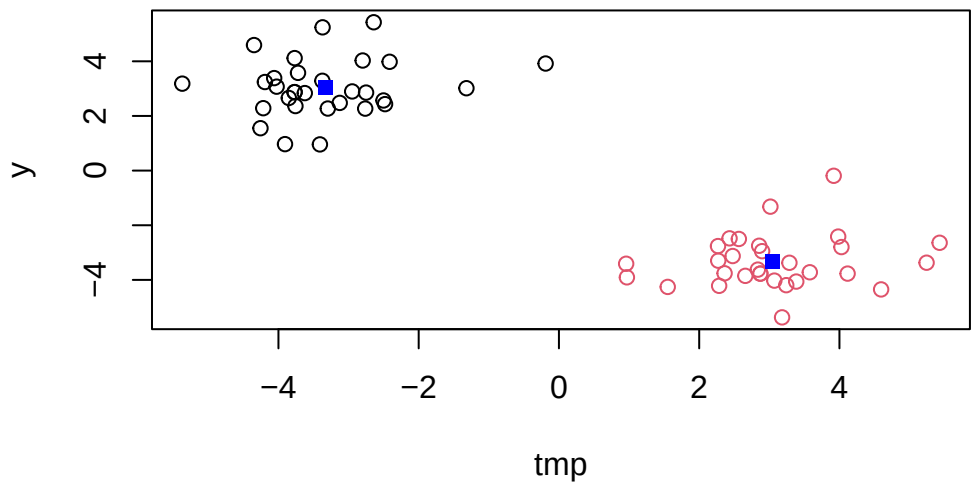
Q. What is the center of the cluster?

```r
k$centers
```

```
        tmp         y
1 -3.334045  3.040037
2  3.040037 -3.334045
```

Lets use main results to plot data with kmeans results

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



##hierarchical Clustering, hclust()

Cluster the same data in `x` with hierarchical clustering (hclust())

d is the distance of one point from all the other points. `hclust()` requires a distnace matrix as input.

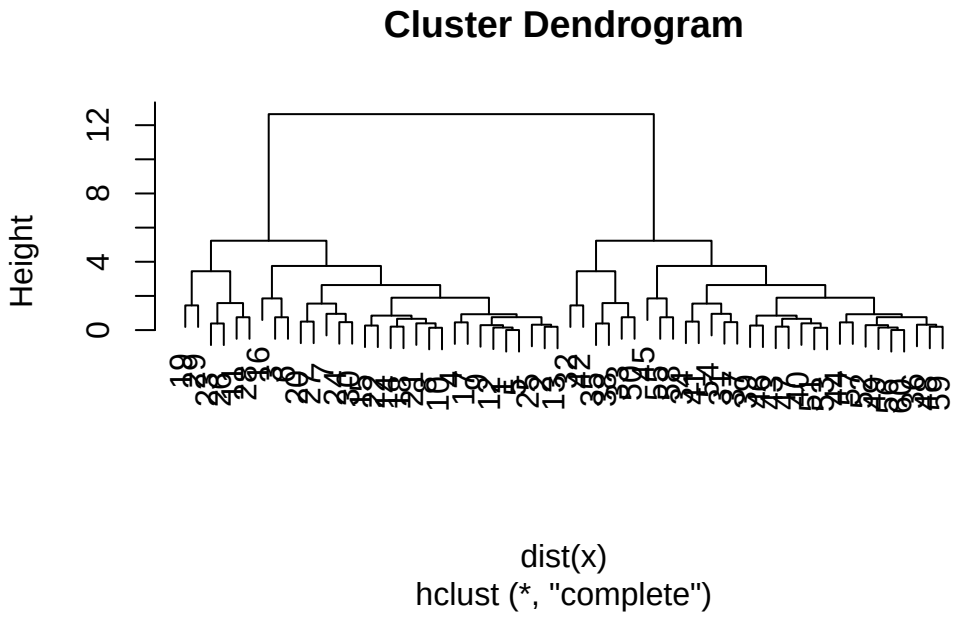
```
hc <- hclust( dist(x))
hc
```

```
Call:
hclust(d = dist(x))
```

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

Plot of hc results

```
plot(hc)
```



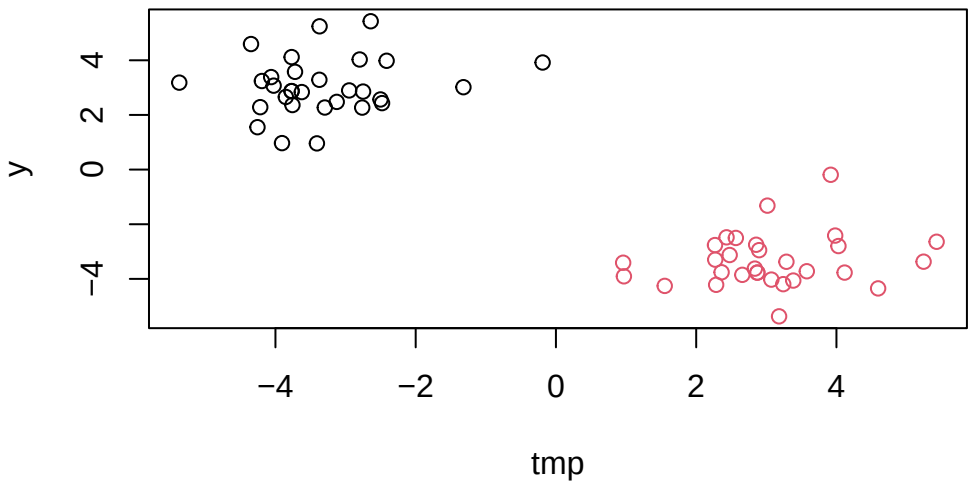
To get cluster membership vector the tree has to be "cut" horizonally at the top of the tree, `cutree()`.

```
groups <- cutree(hc, k=2)
groups
```

```
 [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
```

Groups can be used to plot data

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



#Principal Component Analysis (PCA)

PCA of UK food data

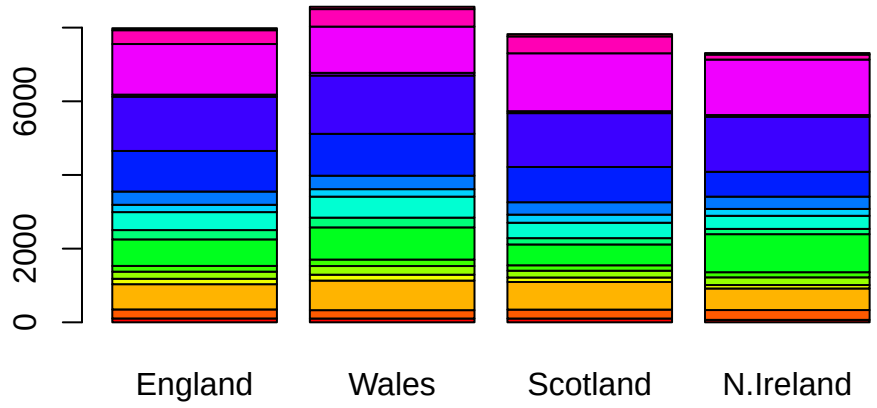
Read data from website and try a few visualizations

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

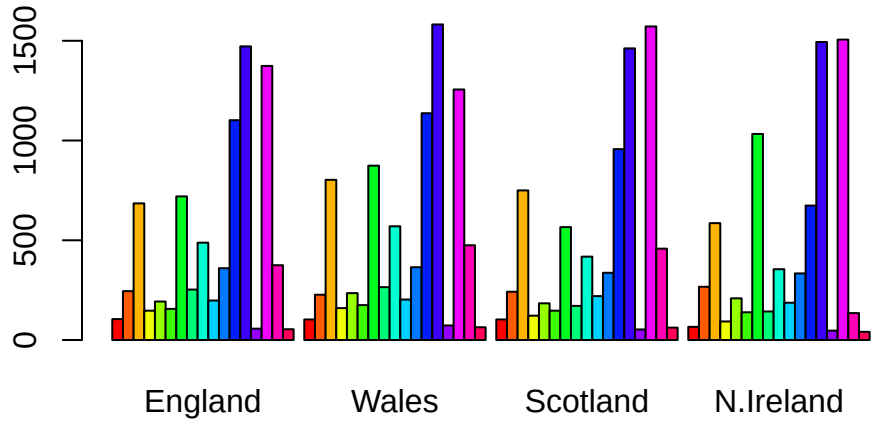
```
                   England Wales Scotland N.Ireland
Cheese                 105   103      103        66
Carcass_meat           245   227      242       267
Other_meat             685   803      750       586
Fish                   147   160      122        93
Fats_and_oils          193   235    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

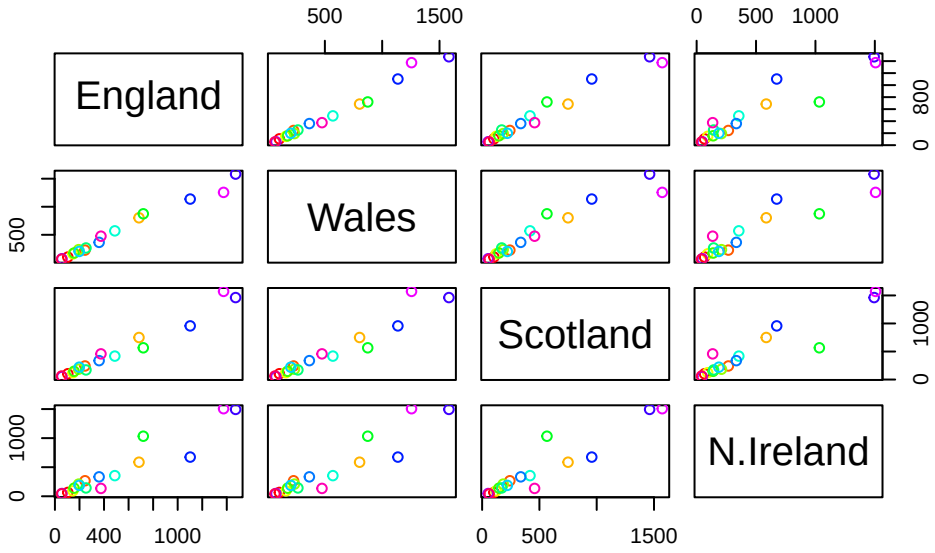
```
cols <- rainbow(nrow(x))
barplot( as.matrix(x), col=cols)
```




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




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



PCA to the rescue! The main base R PCA function is called `prcomp()`

```
#t() switches rows for columns and columns for rows
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
```

#Attributes() tells you whats inside the PCA object

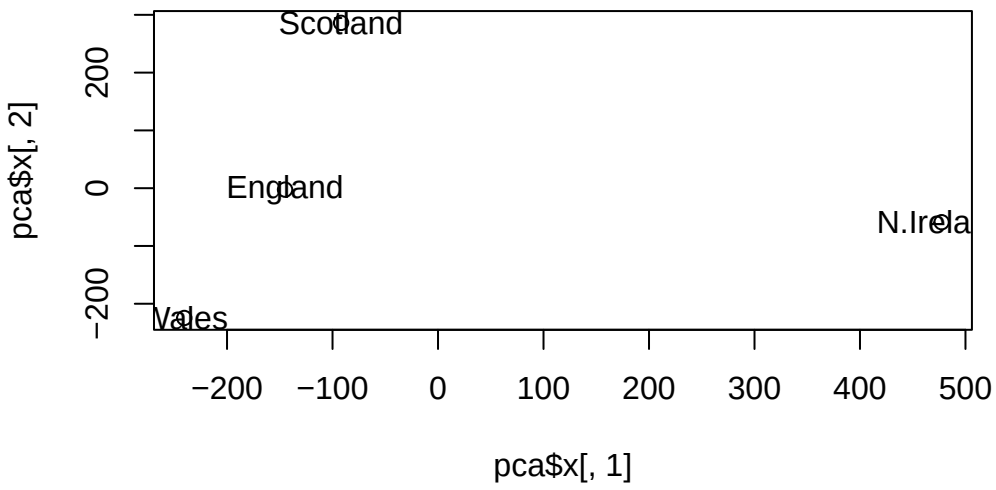
```
attributes(pca)
```

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

$class
[1] "prcomp"
```

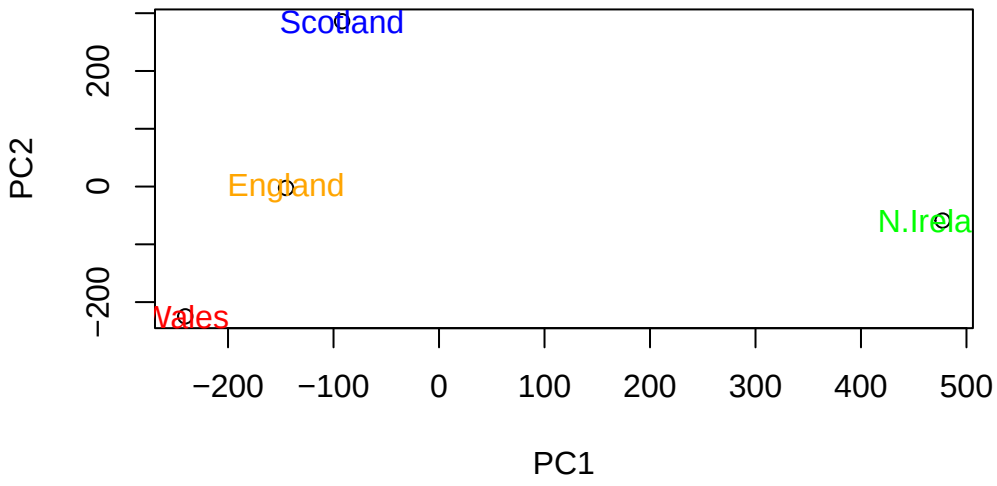
TO make our PCA plot (a.k.a PCA scores plot) we access `pca$x`

```
plot(pca$x[,1], pca$x[,2])
text(pca$x[,1], pca$x[,2], colnames(x))
```



Color up plot

```
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)
```



Use standard deviation to calculate how much variation of the original data is accounted for in PC

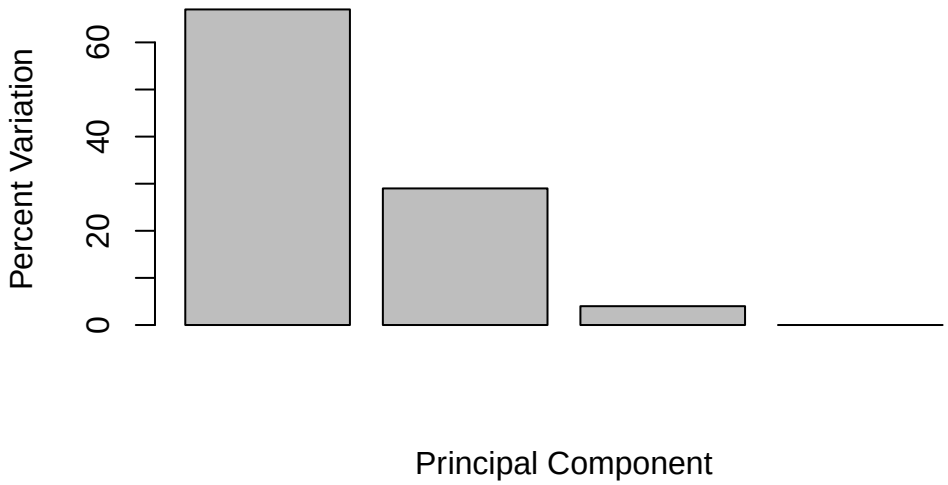
```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

```
[1] 67 29  4  0
```

```
z <- summary(pca)
z$importance
```

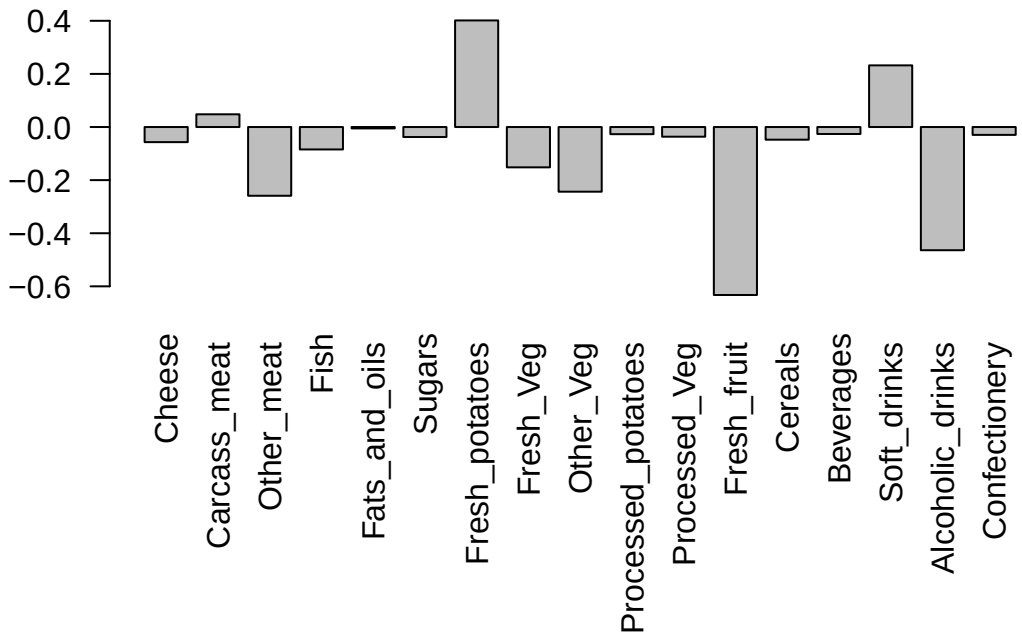
```
                             PC1       PC2      PC3           PC4
Standard deviation     324.15019 212.74780 73.87622 3.175833e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```



Digging deeper (variable loadings) Rotation component tells you how much each variable contributes to PCA

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



##PCA of RNA-Seq 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
```

Q. How many genes are in this data set?

```
nrow(rna.data)
```

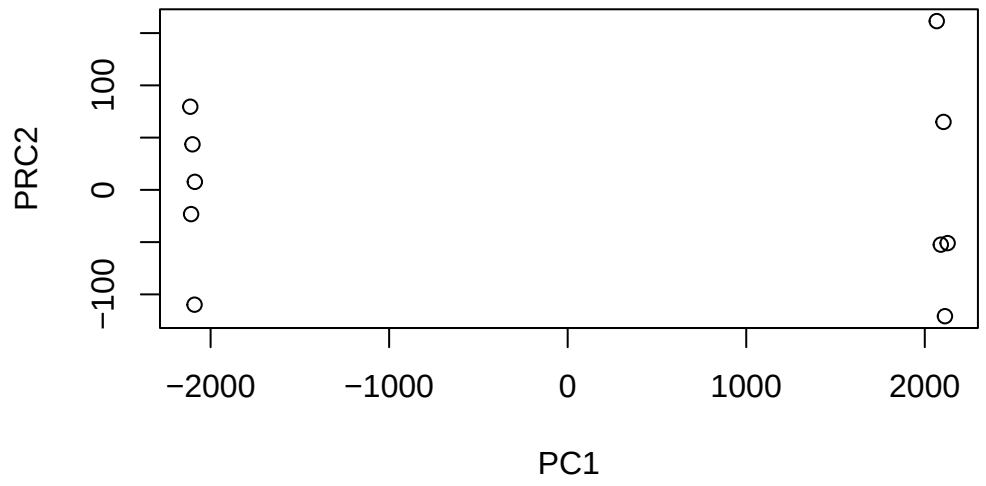
```
[1] 100
```

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

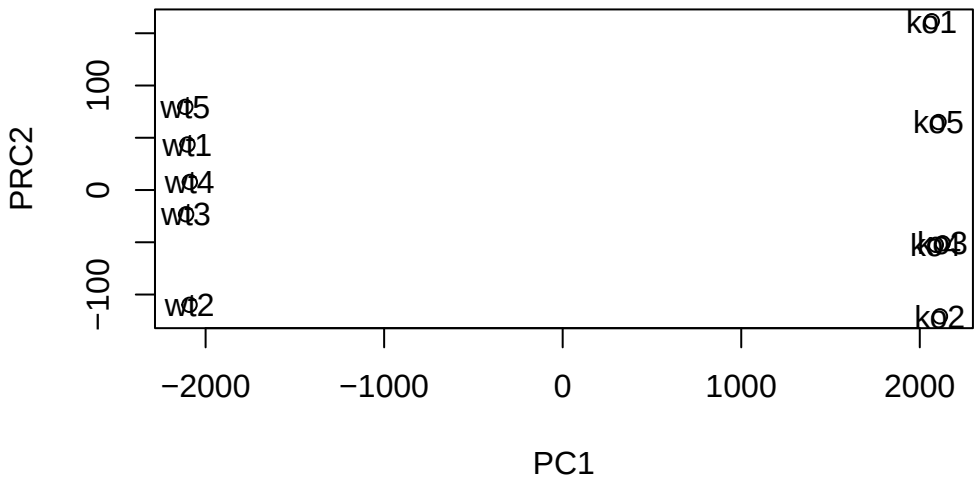
```
Importance of components:
                            PC1     PC2      PC3      PC4      PC5      PC6
Standard deviation     2214.2633 88.9209 84.33908 77.74094 69.66341 67.78516
Proportion of Variance    0.9917  0.0016  0.00144  0.00122  0.00098  0.00093
Cumulative Proportion     0.9917  0.9933  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
```

plot RNA-Seq data

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab= "PRC2")
```




```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab= "PRC2")
text(pca$x[,1], pca$x[,2], colnames(rna.data))
```




```
plot(pca, main="Quick scree plot")
```