Class07 Machine Learning 1

Ramola Baviskar (PID A12228297)

2/8/2022

#Clustering methods

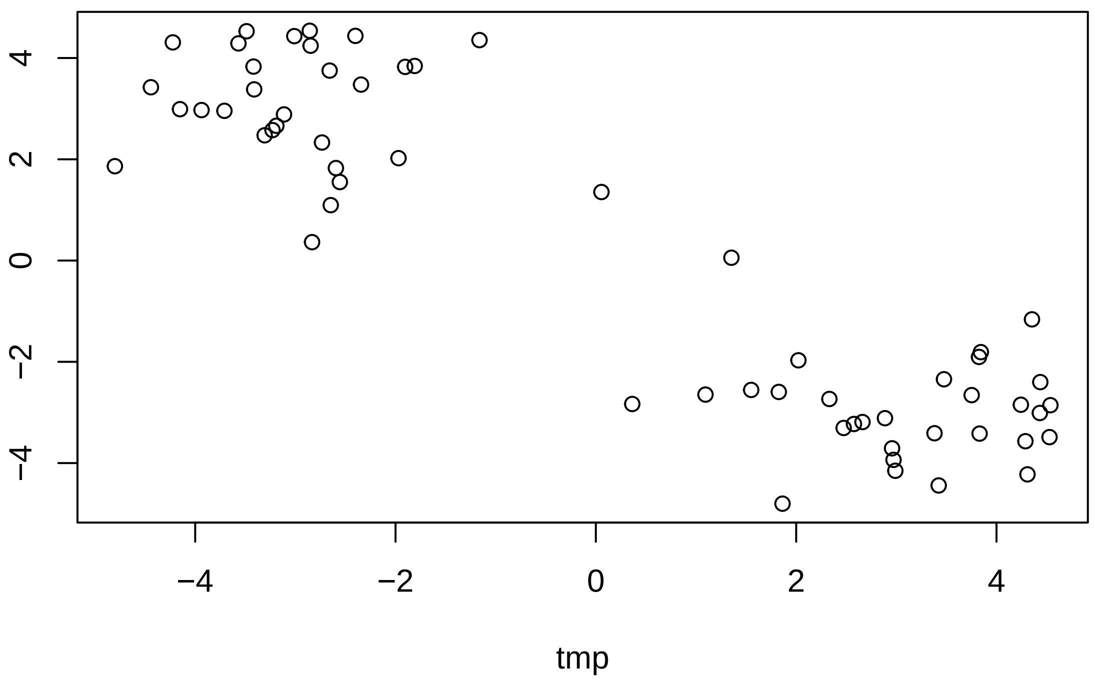
Find groups (aka) clusters in my data

#K-means clustering

Make up some data to test with.

```
#Make up some data w/ 2 clear groups
tmp <- c( rnorm(30, mean = 3), rnorm(30, mean = -3) )
x  <- cbind(tmp, rev(tmp))

plot(x)
```



The 'kmeans()' function does k-means clustering. >Q1. How many points are in each cluster?

```r
k <- kmeans(x, centers=4, nstart=20)
k
```

```
## K-means clustering with 4 clusters of sizes 23, 23, 7, 7
##
## Cluster means:
##         tmp
## 1 -3.172893  3.567209
## 2  3.567209 -3.172893
## 3  1.506122 -2.182678
## 4 -2.182678  1.506122
##
## Clustering vector:
##  [1] 2 2 3 2 2 2 2 2 2 2 2 2 2 3 3 2 3 2 2 3 2 2 2 2 3 2 2 2 3 2 1 4 1 1 1 4 1 1
## [39] 1 1 4 1 1 4 1 4 4 1 1 1 1 1 1 1 1 1 1 4 1 1
##
## Within cluster sum of squares by cluster:
## [1] 30.144489 30.144489  8.859244  8.859244
##  (between_SS / total_SS =  93.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

We can use the dollar syntax to get at the results (components of the returned list).

> Q2. What 'component' pf your result object details: -cluster size? -cluster assignment/membership? -cluster center?

```r
k$size
```

```
## [1] 23 23  7  7
```

```r
k$cluster
```

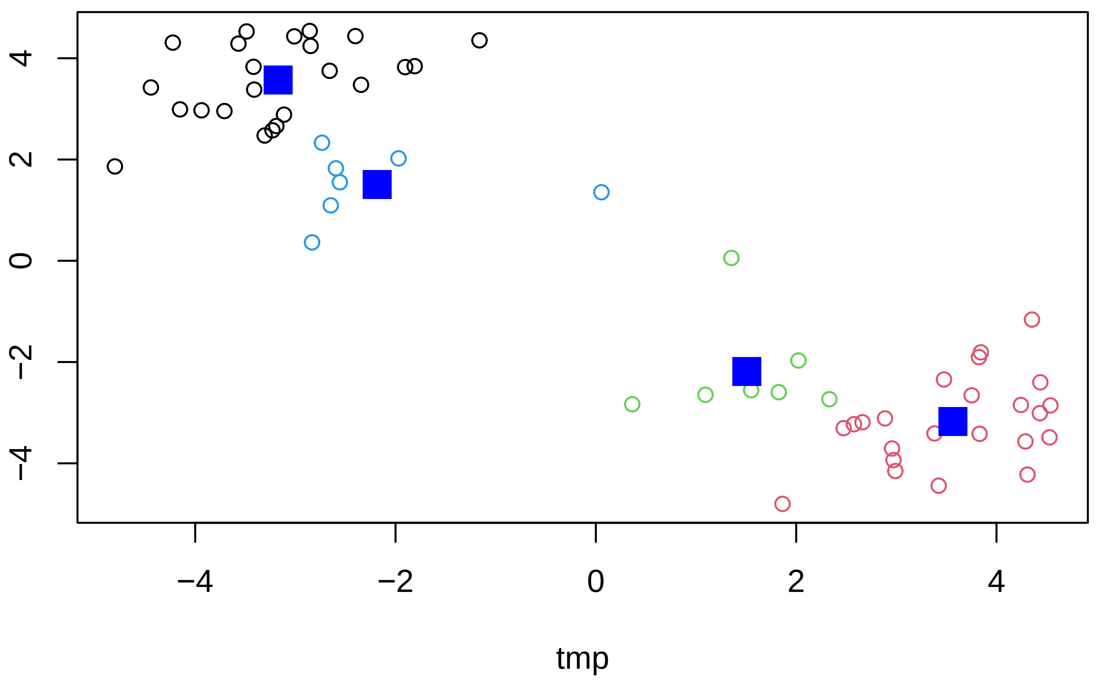
```
##  [1] 2 2 3 2 2 2 2 2 2 2 2 2 2 3 3 2 3 2 2 3 2 2 2 2 3 2 2 2 3 2 1 4 1 1 1 4 1 1
## [39] 1 1 4 1 1 4 1 4 4 1 1 1 1 1 1 1 1 1 1 4 1 1
```

```r
k$centers
```

```
##         tmp
## 1 -3.172893  3.567209
## 2  3.567209 -3.172893
## 3  1.506122 -2.182678
## 4 -2.182678  1.506122
```

> Q3. Plot x colored by the kmeans cluster assigment and add cluster centers as blue points.

```
plot(x, col=k$cluster)
points(k$centers, col="blue", pch=15, cex=2)
```



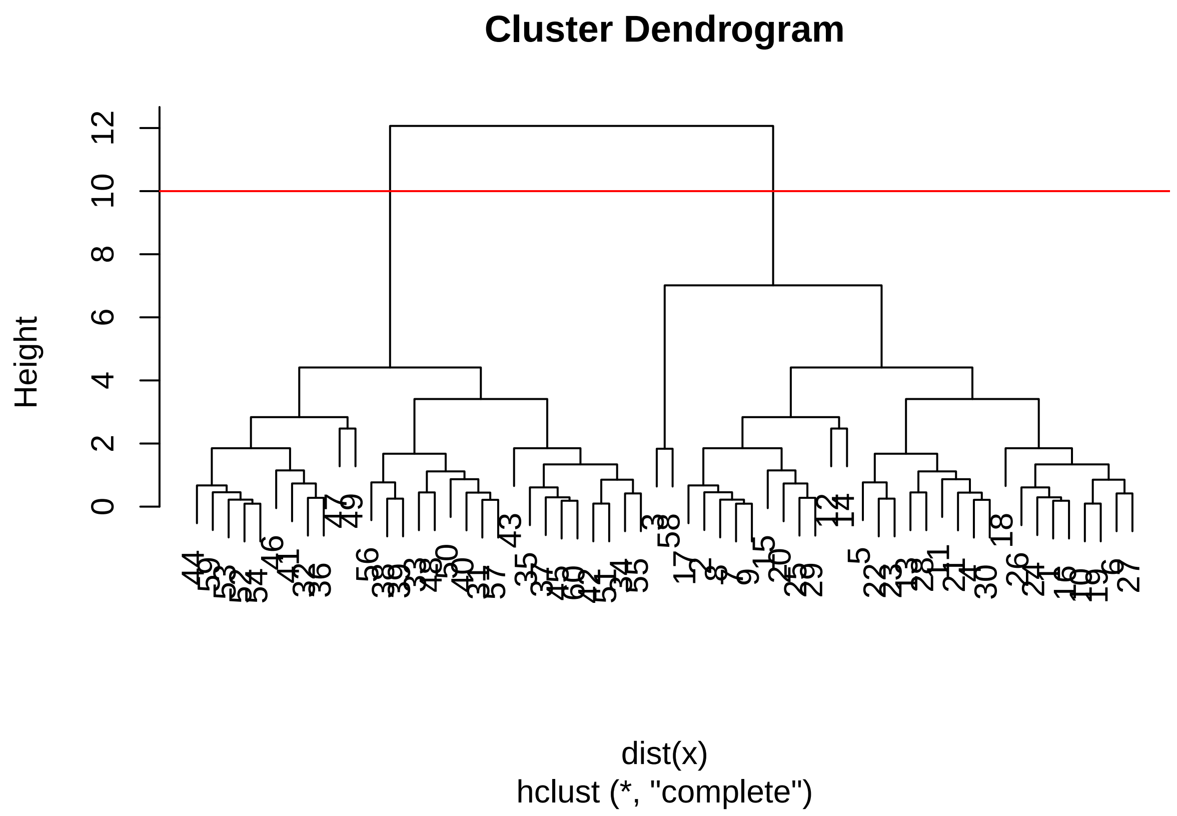
##Hierarchical Clustering

The hclust() fucntion needs a distance matrix as input nor our original data.For this we use the ‘dist(’ function

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

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

```
plot(hc)
abline(h=10, col="red")
```



To get our cluster membership vector we must cut our tree. For this we use 'cutree()'.

```
cutree(hc, h=10)
```

```
##  [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 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 1 2 2
```

We can cut by a given height (h=) or into a given number of k groups w/ k=.

```
cutree(hc, k=2)
```

```
##  [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 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 1 2 2
```

#Principal Component Analysis

##PCA of UK food data

Let's read the data about the stuff people in the UK eat and drink:

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

Look at the first bit of file:

```
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
```

How many columns in dataset:

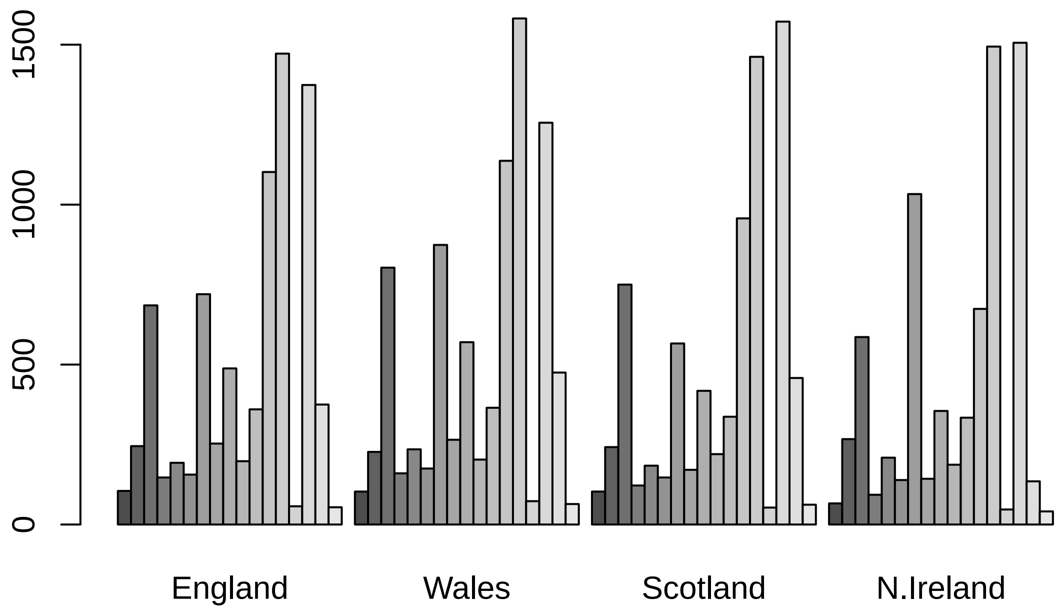
```
ncol(x)
```

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

Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions? 4 columns and 17 rows 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? Using the codeblock and running it multiple times eventually led to the removal of all the columns, so it's not my preferred approach as I feel like it's error-prone.

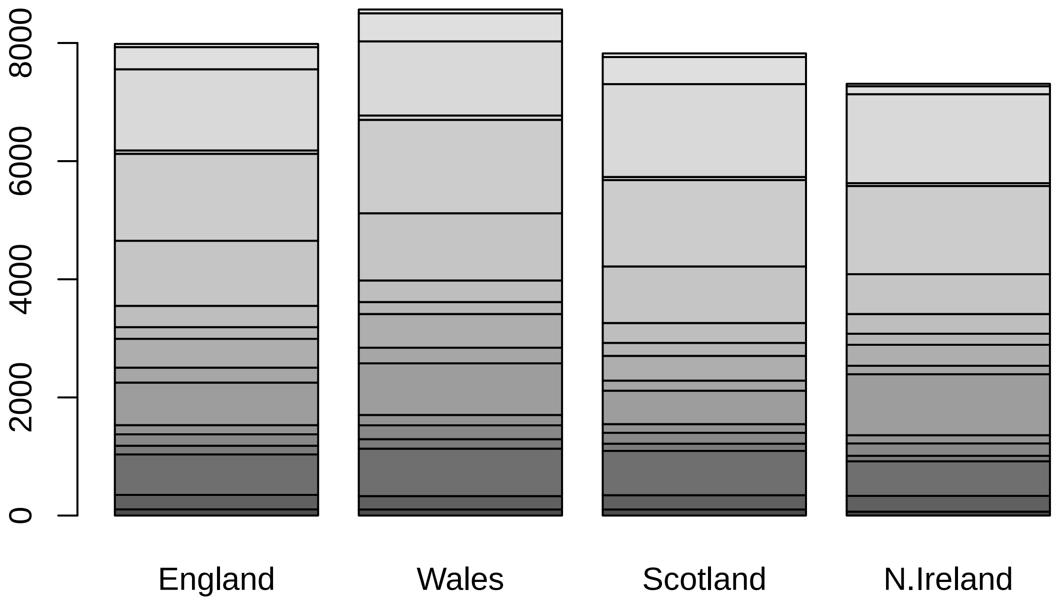
We can make some plots to try to understand this data a bit more. for example barplots:

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



Q3: Changing what optional argument in the above barplot() function results in the following plot? Changing the 'beside' function from TRUE to FALSE would result in a stacked plot.

```
barplot(as.matrix(x), beside=FALSE)
```



#PCA

The main base R function for PCA is 'prcomp()'.

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

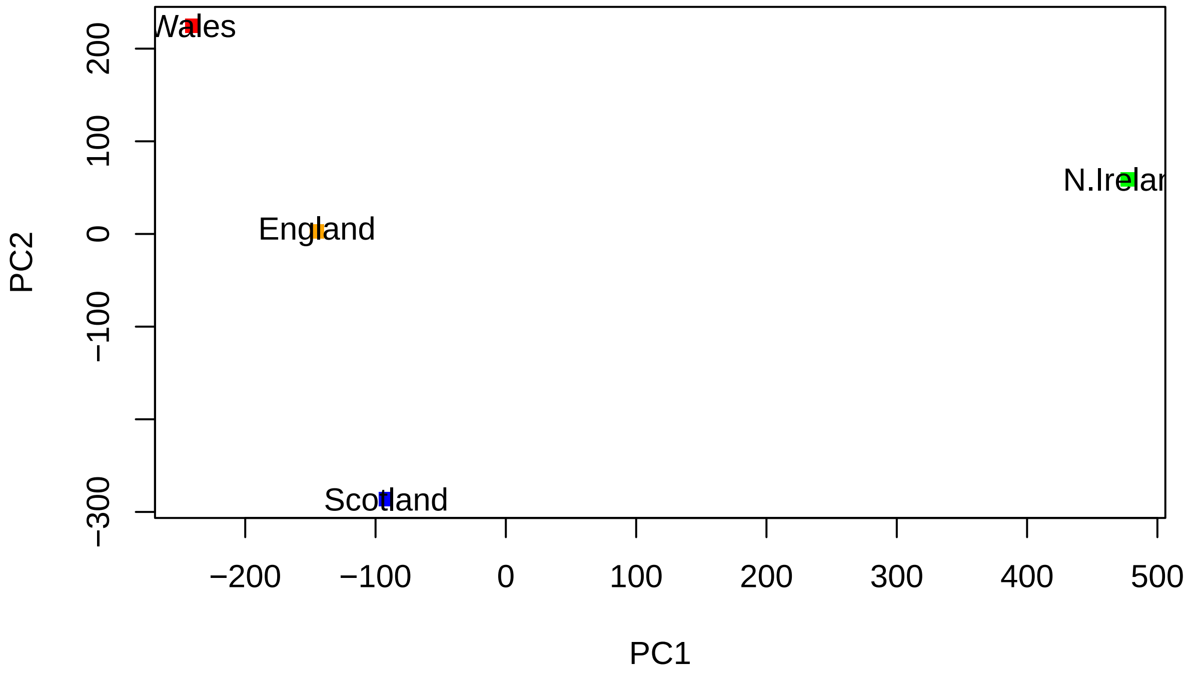
```
## Importance of components:
##                              PC1      PC2      PC3       PC4
## Standard deviation     324.1502 212.7478 73.87622 4.189e-14
## Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
## Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

What's in this returned PCA object?

```
attributes(PCA)
```

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

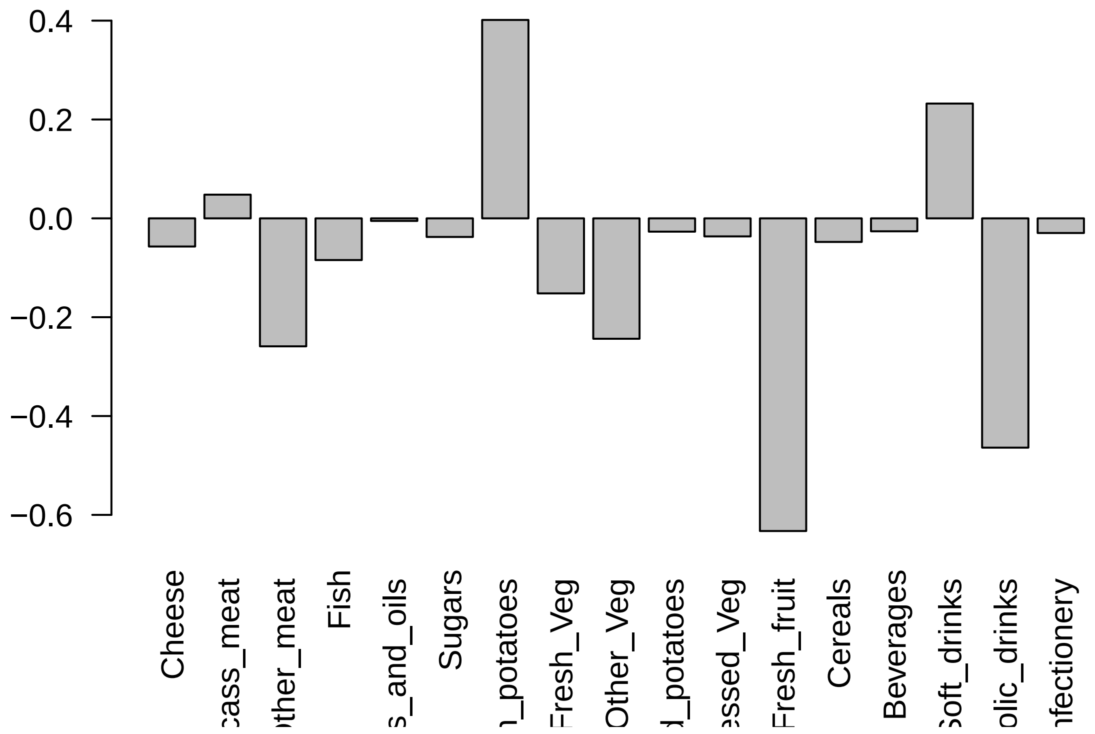
```
plot(PCA$x[,1:2], col=c("orange", "red", "blue", "green"), pch=15)
text(PCA$x[,1], PCA$x[,2], labels=colnames(x))
```



Q7. Complete the code to generate a plot of PC1 vs PC2. T Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

We can look at how the variables contribute to our new PCs by examining the 'PCA$rotation' component of our results.

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



```r
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
```

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

```
## [1] 100
```

```r
ncol(rna.data)
```

```
## [1] 10
```

Q10: How many genes and samples are in this data set? 100 genes; 10 samples

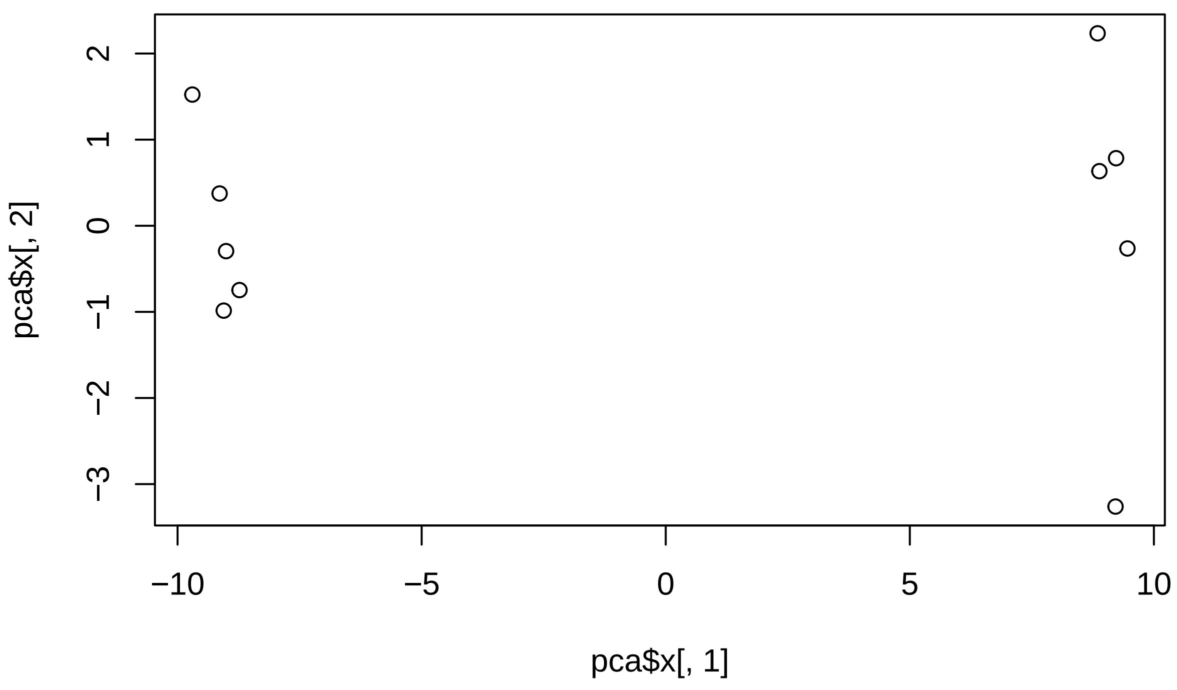
Let's do PCA of htis dataset; first take the transpose as that's what hte prcomp() function wants.

```
pca <- prcomp(t(rna.data), scale=TRUE)
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     9.6237 1.5198 1.05787 1.05203 0.88062 0.82545 0.80111
## Proportion of Variance 0.9262 0.0231 0.01119 0.01107 0.00775 0.00681 0.00642
## Cumulative Proportion  0.9262 0.9493 0.96045 0.97152 0.97928 0.98609 0.99251
##                            PC8     PC9      PC10
## Standard deviation     0.62065 0.60342 3.348e-15
## Proportion of Variance 0.00385 0.00364 0.000e+00
## Cumulative Proportion  0.99636 1.00000 1.000e+00
```

We can make our score plot (aka PCA plot) from the 'pca$x'.

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



Make a color vector by wt and ko

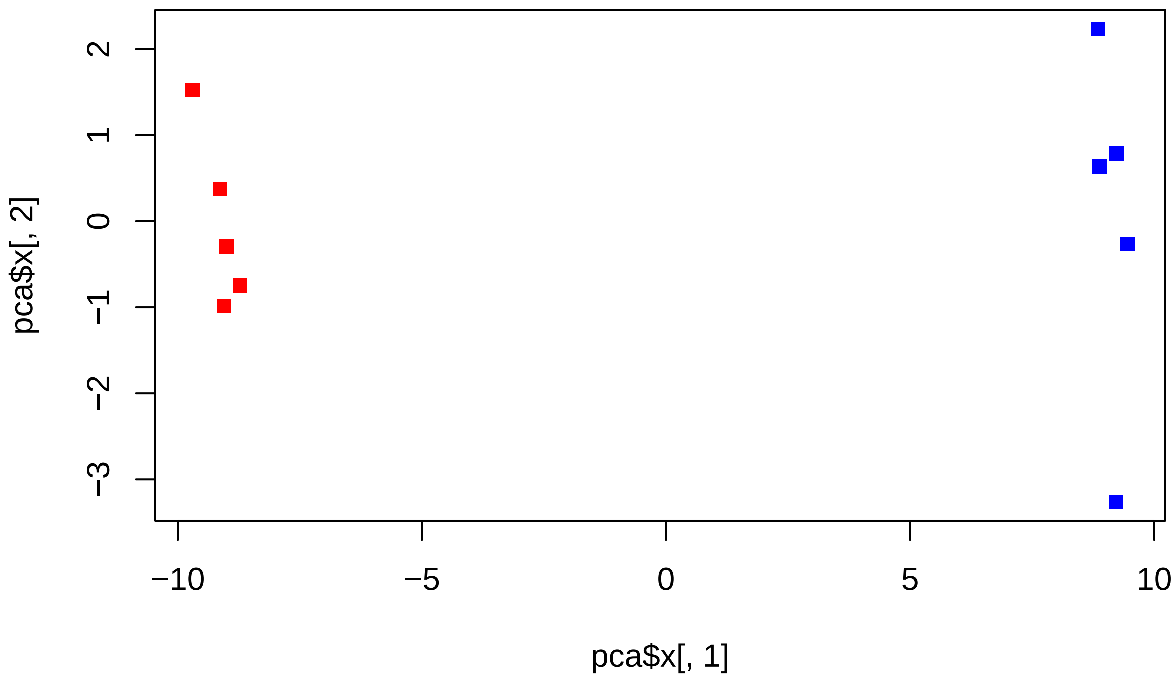
```
rep("red", 5)
```

```
## [1] "red" "red" "red" "red" "red"
```

```
rep("blue", 5)
```

```
## [1] "blue" "blue" "blue" "blue" "blue"
```

```
colvec <- c(rep("red", 5), rep("blue", 5))
plot(pca$x[,1], pca$x[,2], col=colvec, pch=15)
```


pca$x[, 2]
pca$x[, 1]