Class 7: Machine Learning 1

Macy Hoang

2/8/2022

Clustering methods

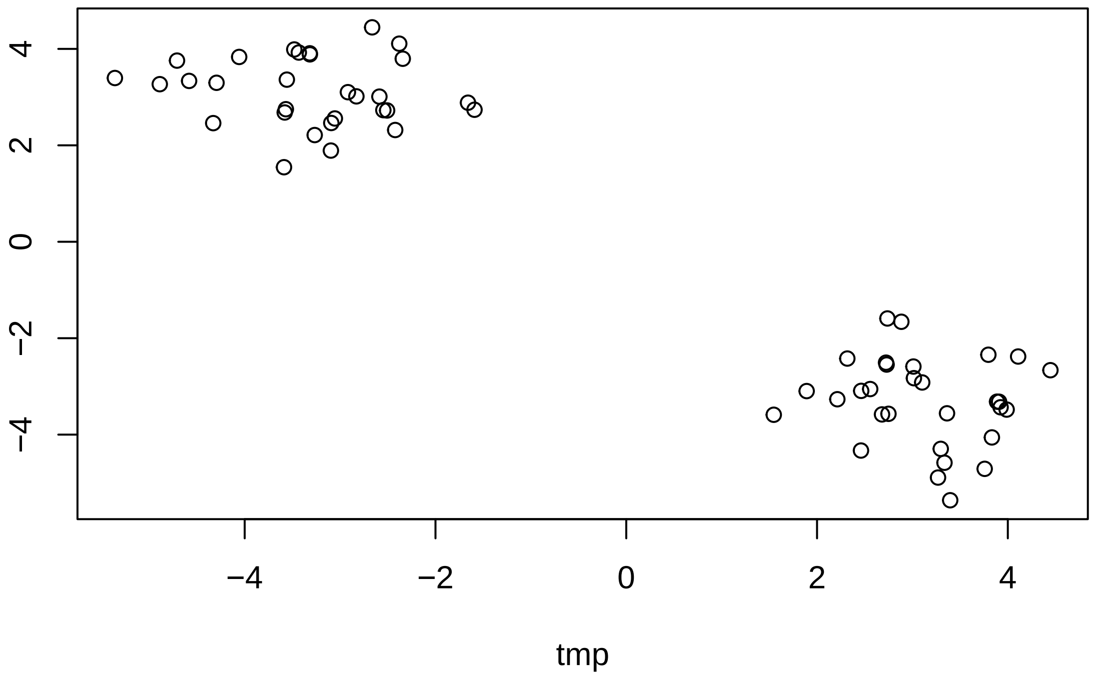
Find groups (aka) clusters in my data

K-means clustering

Make up some data to test with.

```r
# Make up some data with 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

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

```
## K-means clustering with 4 clusters of sizes 13, 17, 13, 17
##
## Cluster means:
##         tmp
## 1  3.605011 -3.999787
## 2 -2.766514  2.736779
## 3 -3.999787  3.605011
## 4  2.736779 -2.766514
##
## Clustering vector:
##  [1] 1 1 4 4 4 4 4 4 4 1 4 1 4 4 1 1 1 1 4 4 1 4 1 1 4 4 1 1 4 4 2 2 3 3 2 2 3 3
## [39] 2 3 2 2 3 3 3 3 2 2 3 2 3 2 2 2 2 2 2 2 3 3
##
## Within cluster sum of squares by cluster:
## [1] 10.08763 11.76362 10.08763 11.76362
##  (between_SS / total_SS =  96.7 %)
##
## 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)

> Q1. How many points are in each cluster?

```
k$size
```

```
## [1] 13 17 13 17
```

> Q2. What 'componeny' of your result object details - cluster size? - cluster assignment/membership? - cluster center?

```
k$size
```

```
## [1] 13 17 13 17
```

```
k$cluster
```

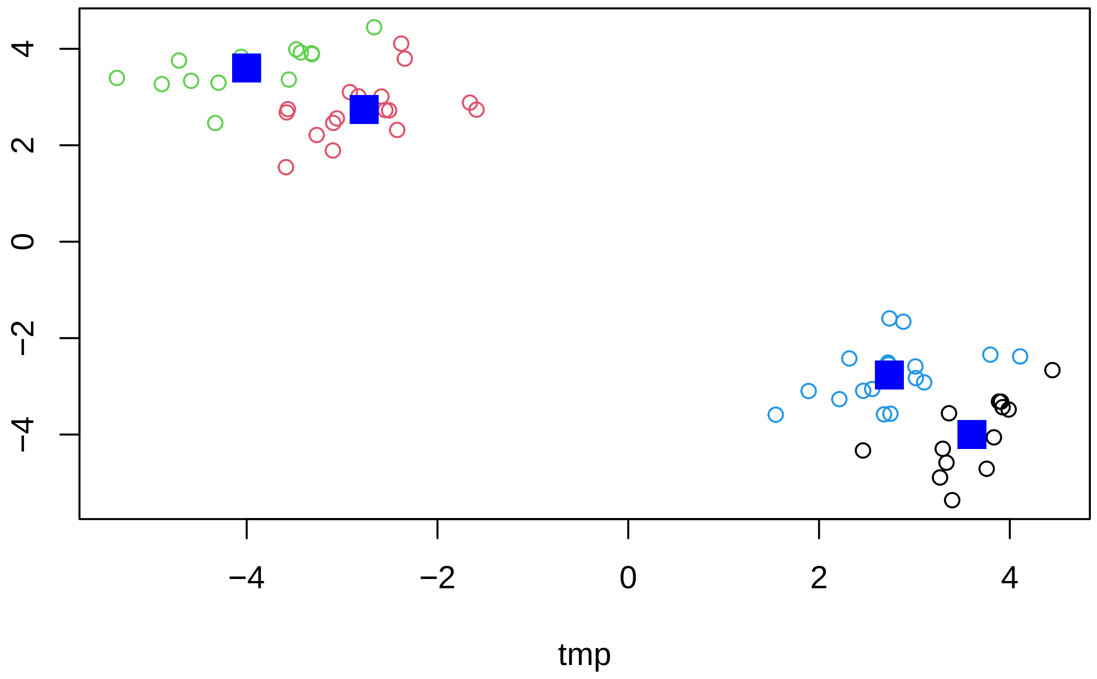
```
##  [1] 1 1 4 4 4 4 4 4 4 1 4 1 4 4 1 1 1 1 4 4 1 4 1 1 4 4 1 1 4 4 2 2 3 3 2 2 3 3
## [39] 2 3 2 2 3 3 3 3 2 2 3 2 3 2 2 2 2 2 2 2 3 3
```

```
k$centers
```

```
##         tmp
## 1  3.605011 -3.999787
## 2 -2.766514  2.736779
## 3 -3.999787  3.605011
## 4  2.736779 -2.766514
```

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

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



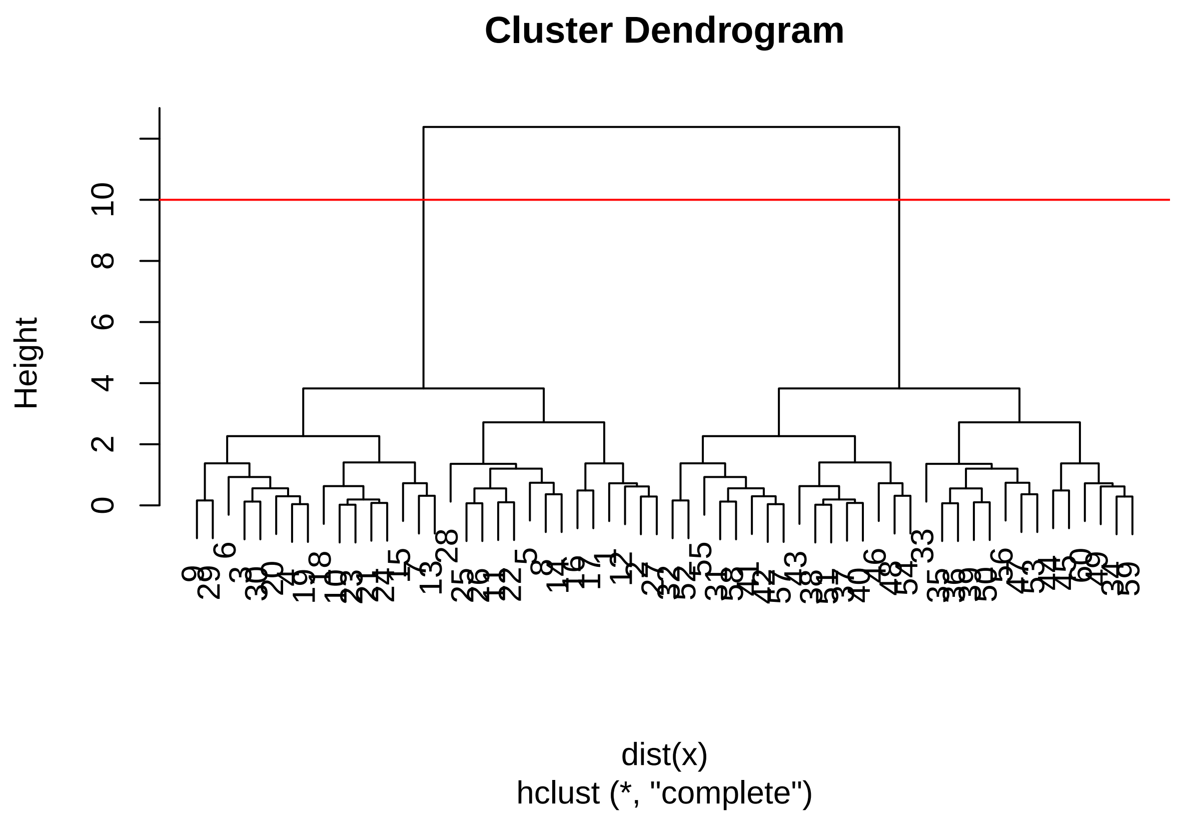
Hierarchial Clustering

The hclust() needs a distance matrix as input not 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
```

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



To get our cluster membership vector, we need to cut our tree and for this we use the `cutree()`

```r
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 2 2 2 2
```

You can cut by a different height h= or into a given number of k groups with k=

```r
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 2 2 2 2
```

Principal Component Analysis

PCA of UK food data

Let's read our data about the weird stuff folks from the UK eat and drink:

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

> Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

```r
dim(x)
```

```
## [1] 17  5
```

Look at the first bit of the file:

```r
## Preview the first 6 rows
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
```

> 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 prefer setting the argument with row.names because each time we use the minus indexing, it does not show all the data when you run it multiple times

Well let's set the rownames() and remove the first column

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

```r
# Just looking at columns
ncol(x)
```

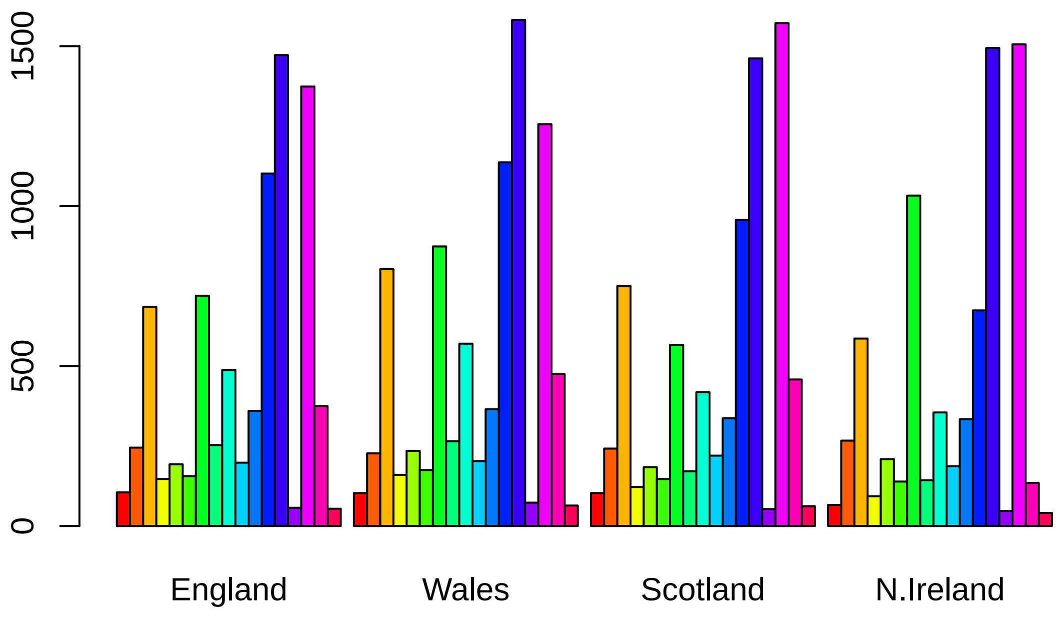
```
## [1] 4
```

```r
## Complete the following code to find out how many rows and columns are in x?
dim(x)
```

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

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

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



PCA to the rescue

The main base R function for PCA is called `prcomp()`

> Q3. Changing what optional argument in the above barplot() function results in the following plot? - adding the argument beside = T

```
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
```

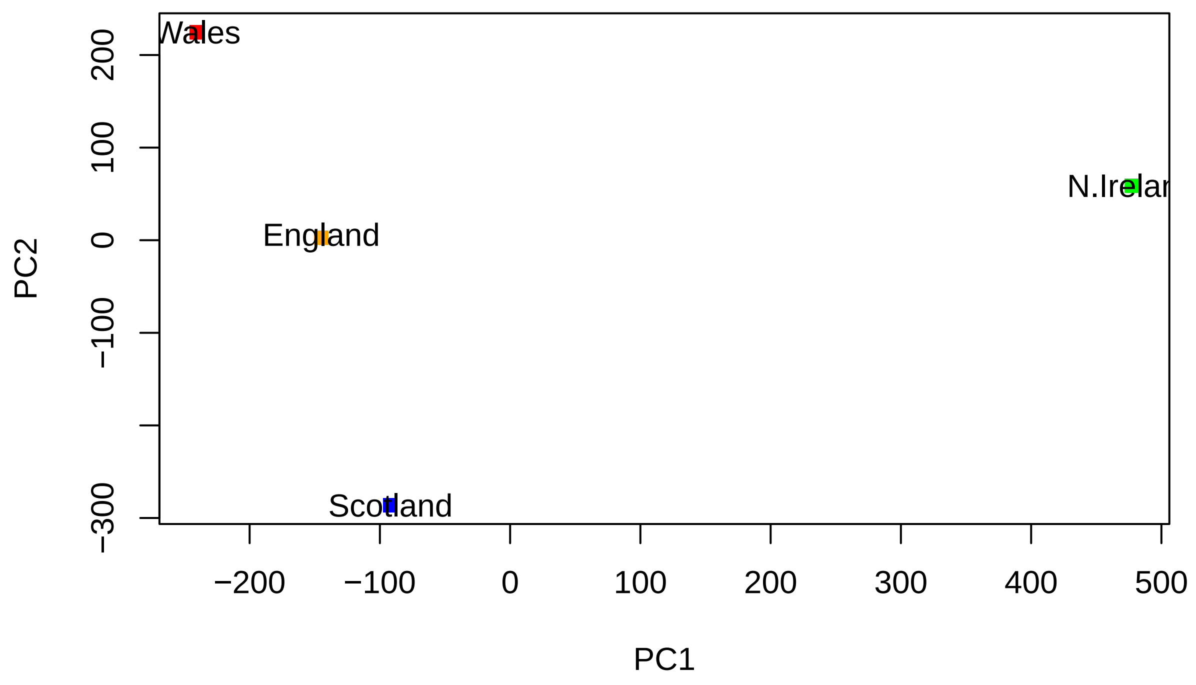
What is this returned pca object?

```
attributes(pca)
```

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

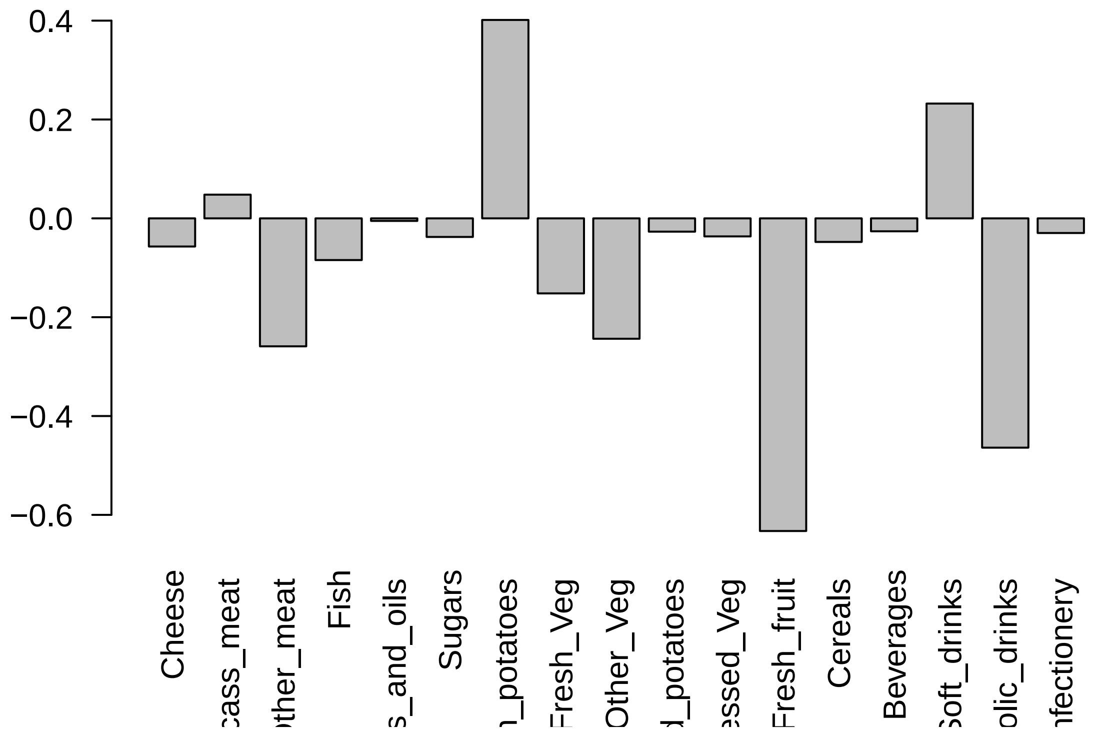
```
##
## $class
## [1] "prcomp"
```

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



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)
```



An RNA-Seq PCA example

Read the data first

```
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
```

Q10. How many genes and samples are in this data

How many genes (how many rows)?

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

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

How many experiments (columns)?

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

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

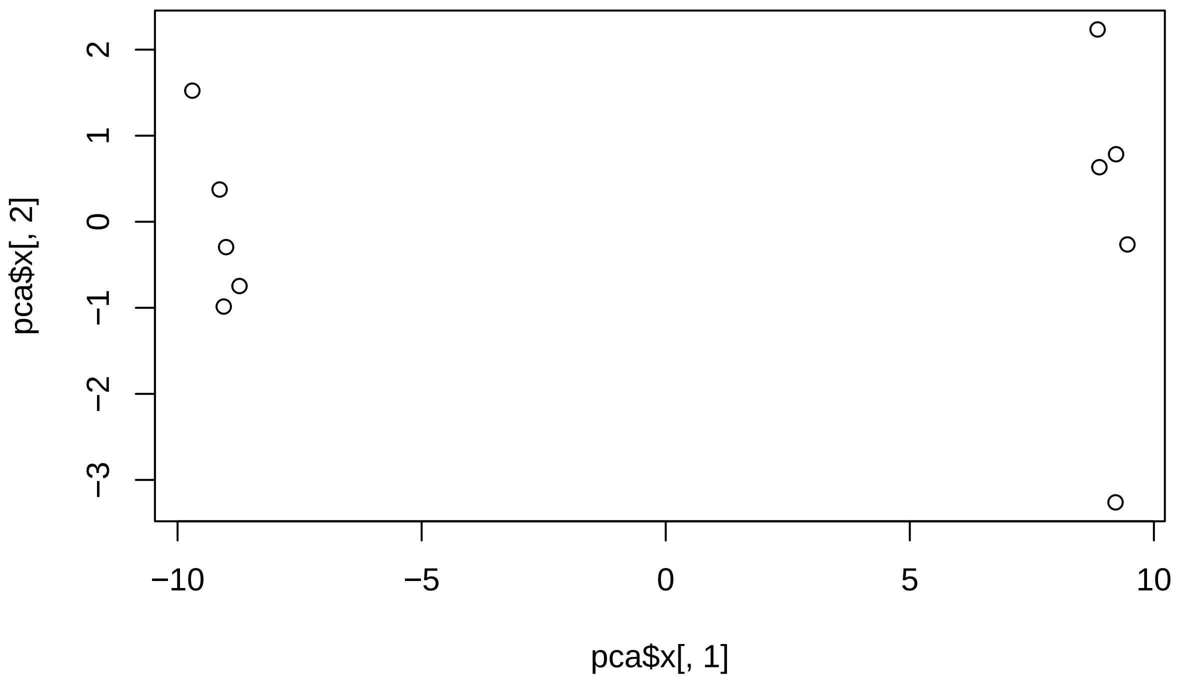
Let's do PCA of this data set First take the transpose as that is what the `prcomp()` function wants

```
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
```

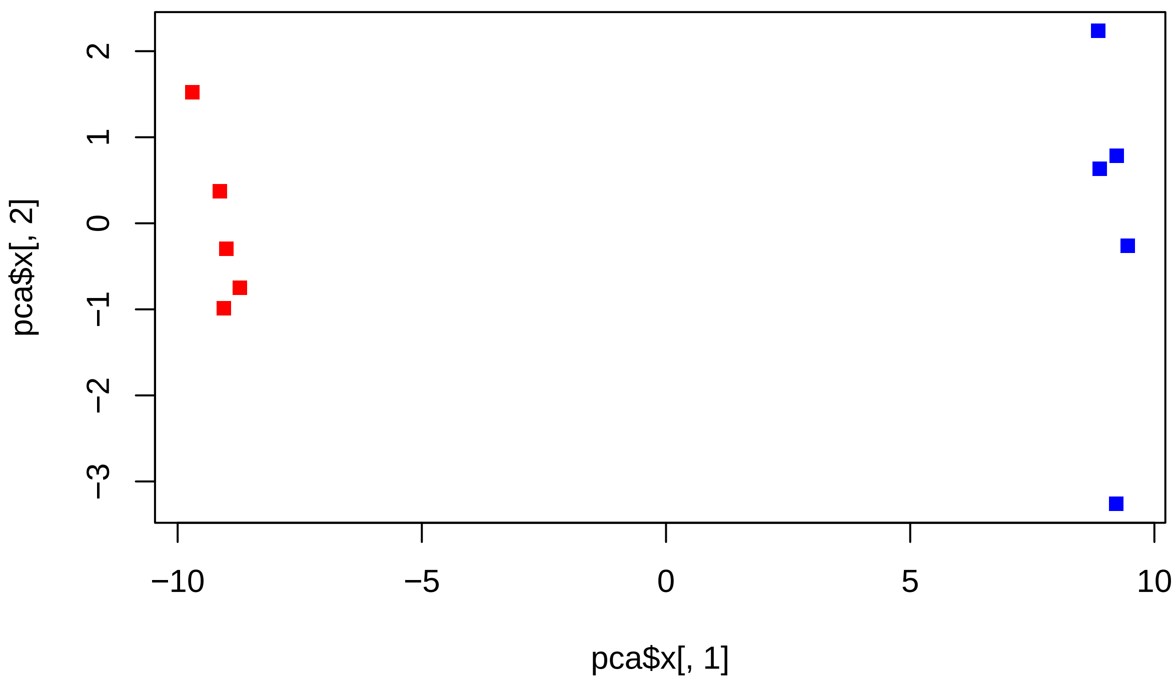
We can make our score plot (aka PCA plot) from the `pca$x`

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



Make a little color vector to color up our plot by wt and ko

```
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]