Class 7 Lab

AUTHOR
Alexis Galano (PID: A17628362)

PUBLISHED
January 30, 2024

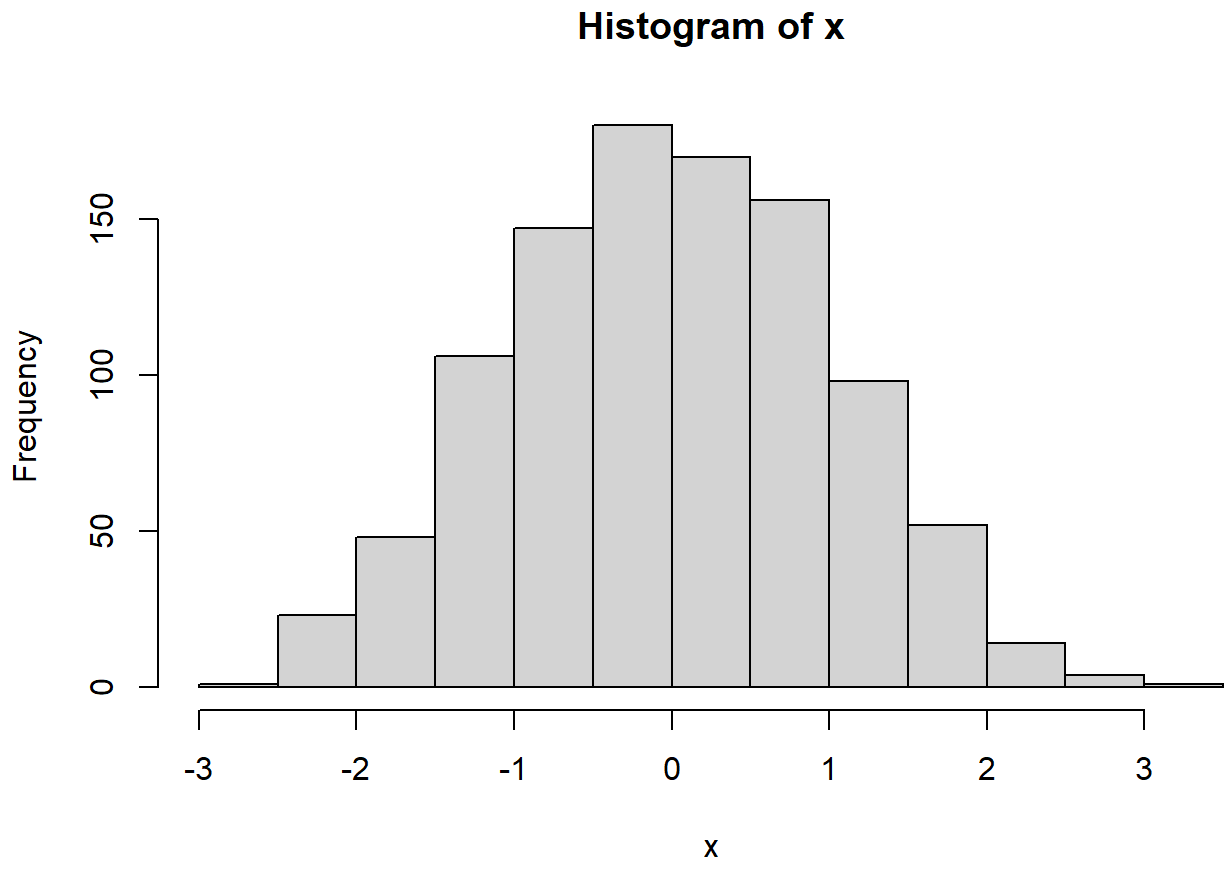
Clustering Methods

The goal is to find groupings (clusters) in your input data.

Using kmeans()

First, let's make up some data to cluster.

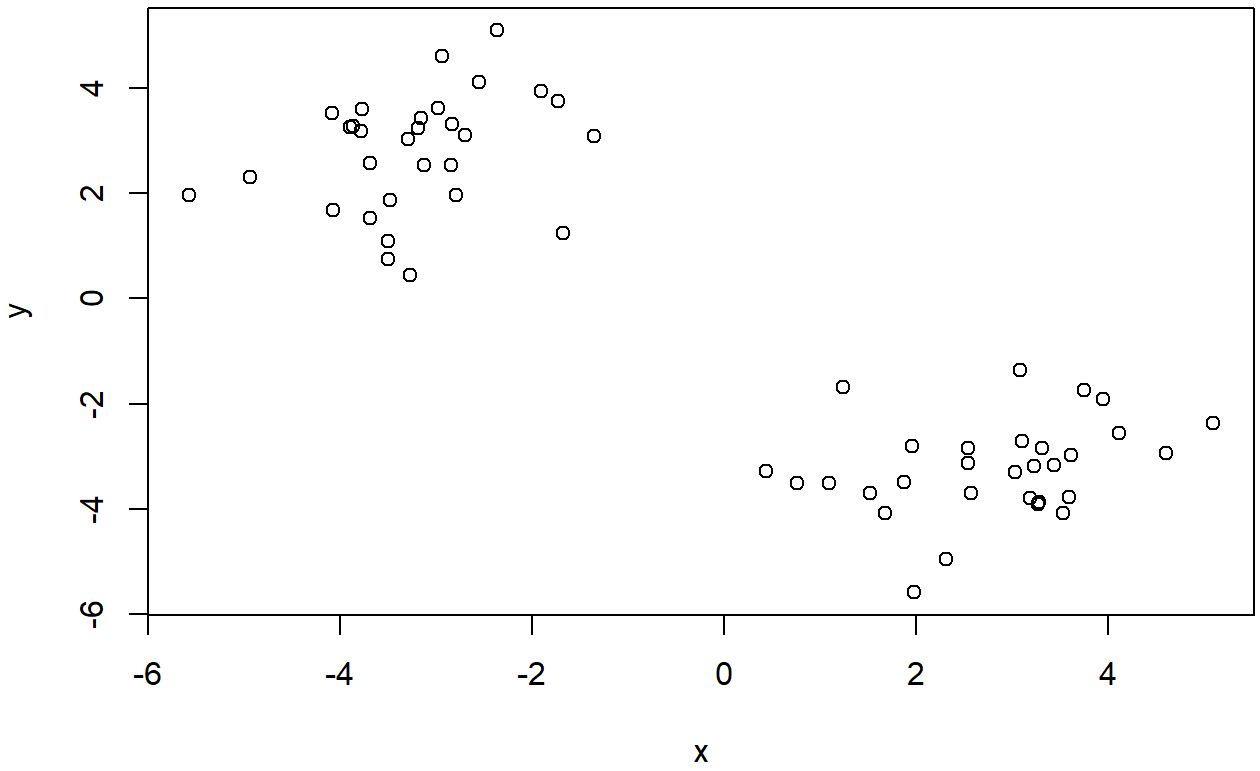
```
x <- rnorm(1000)
hist(x)
```



Make a vector of length 60 with 30 points centered at -3 and 30 points centered at +3.

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

```
tmp <- c(( rnorm(30, mean=-3)), rnorm(30, mean=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)
k
```

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

Cluster means:
          x         y
1 -3.221103  2.786055
2  2.786055 -3.221103

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] 60.58288 60.58288
 (between_SS / total_SS =  89.9 %)
```

```
Available components:

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

kmeans returns an object of class “kmeans” which has a print and a fitted method

> Q. How many points are in each cluster?

```
k$size
```

```
[1] 30 30
```

> Q. How do we get 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 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
```

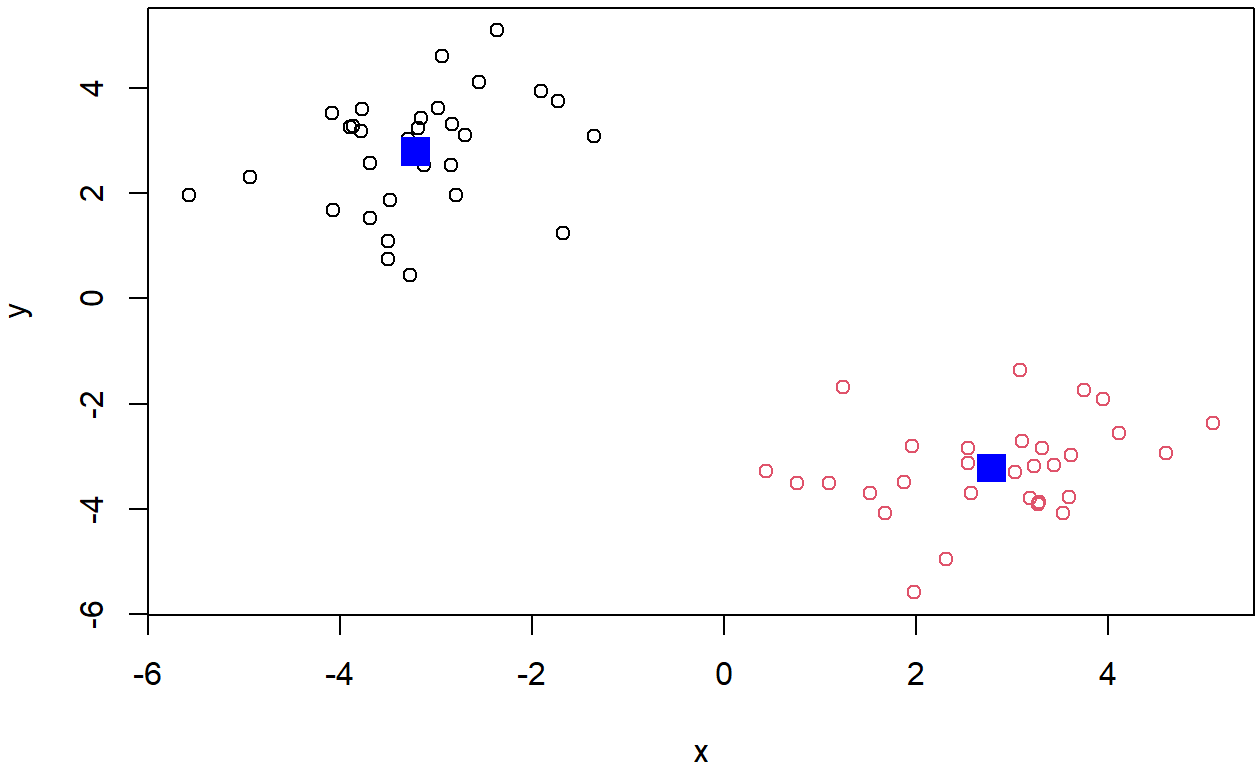
> Q. What about cluster centers?

```
k$centers
```

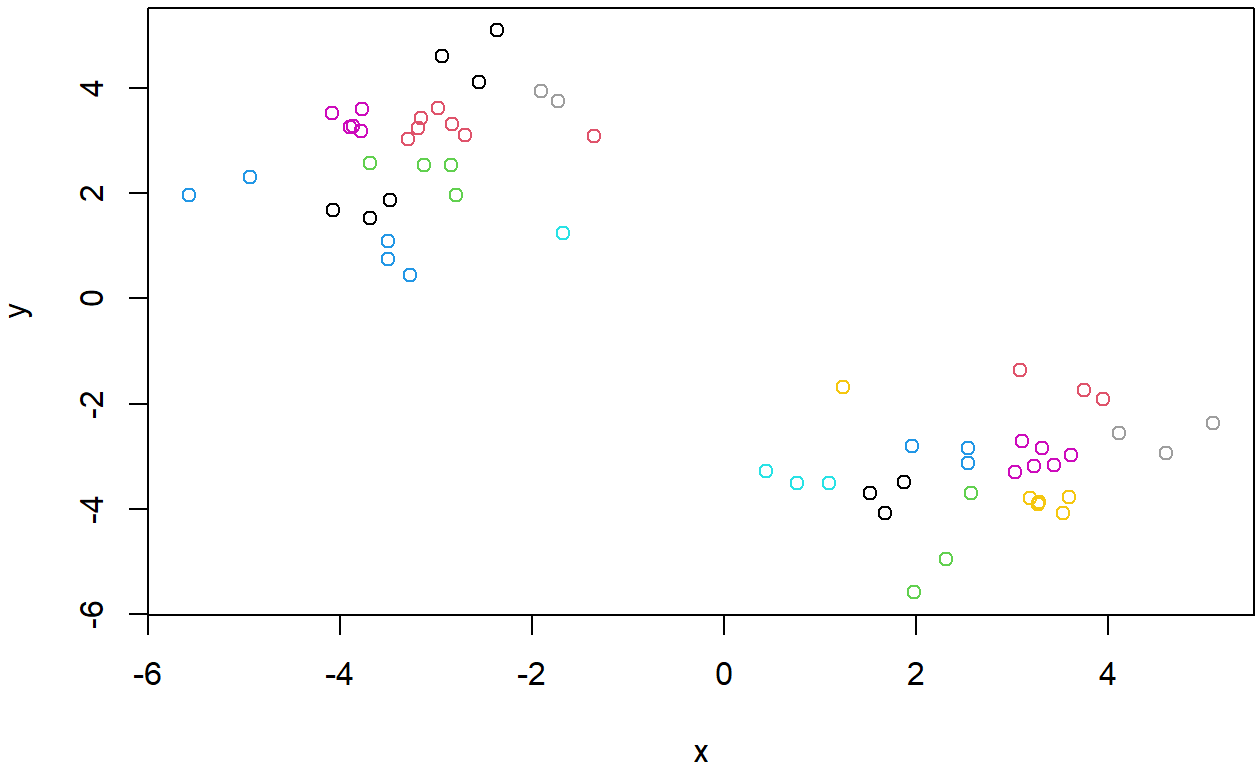
```
          x         y
1 -3.221103  2.786055
2  2.786055 -3.221103
```

Now we got to the main results lets use them to plot our data with the kmeans result.

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




```
# kmeans
k4 <- kmeans(x, center=20)
#plot
plot(x, col=k4$cluster)
```



A big limitation of kmeans is that it does what you ask even if you ask for silly clusters.

Hierarchical Clustering

Now for hclust()

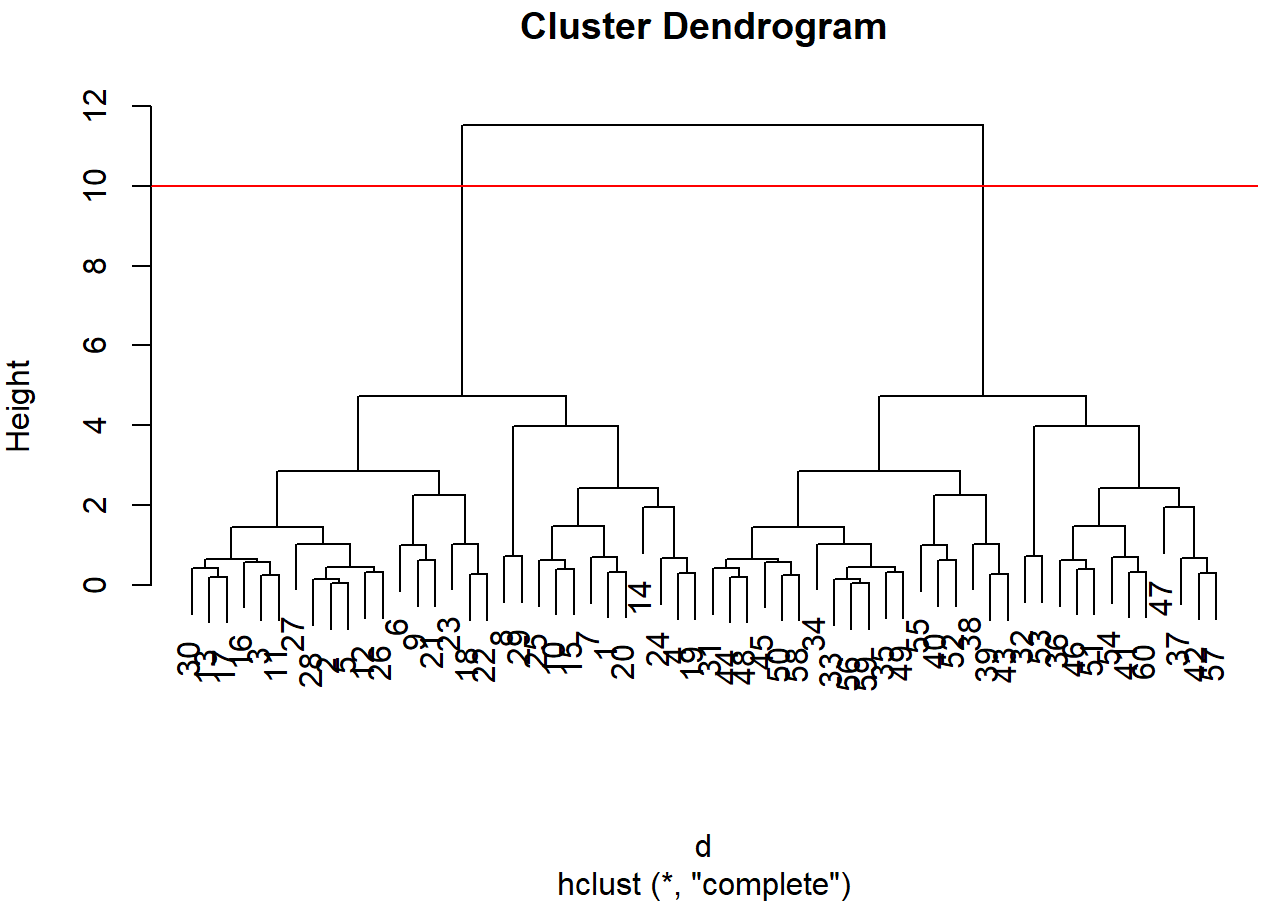
We will cluster the same data 'x' with the 'hclust()'. In this case 'hclust()' requires a distance matric as input.

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

```
Call:
hclust(d = d)

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

Now plot results:

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



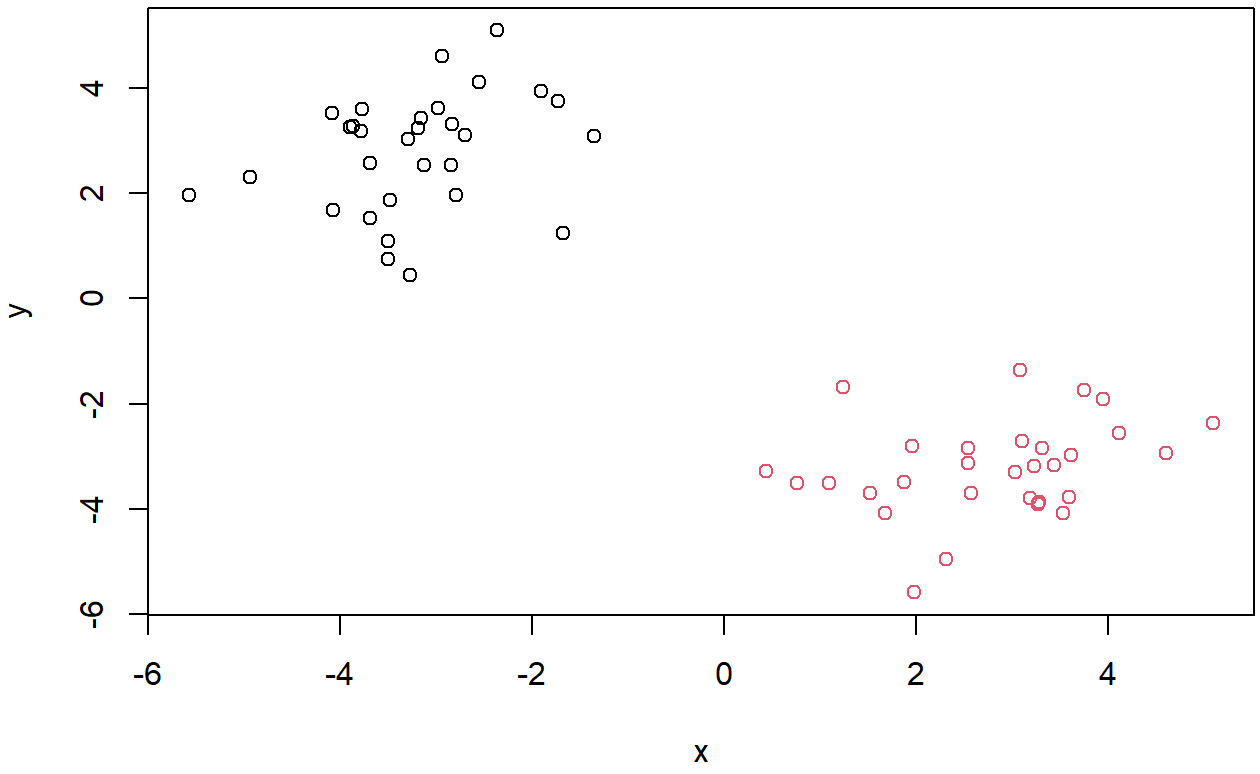
To get our cluser membership vector we need to "cut" the tree with the 'cutree()'

```
grps <- cutree(hc, h=10)
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
```

Now plot our data with the hclust() results.

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



Principal Component Analysis (PCA)

PCS of UK food data

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

> 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
```

> A1. There are 17 rows and 4 columns in this data frame.

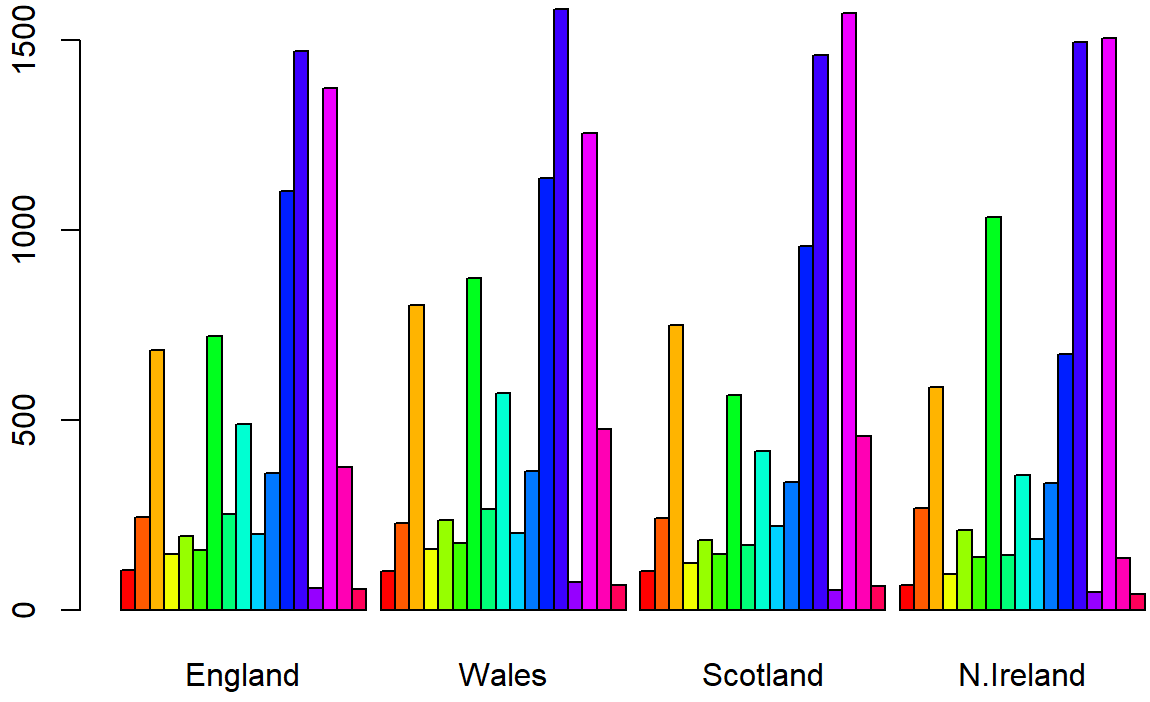
```
x <- read.csv(url, row.names=1)
head(x)
```

```
              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
```

> 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?

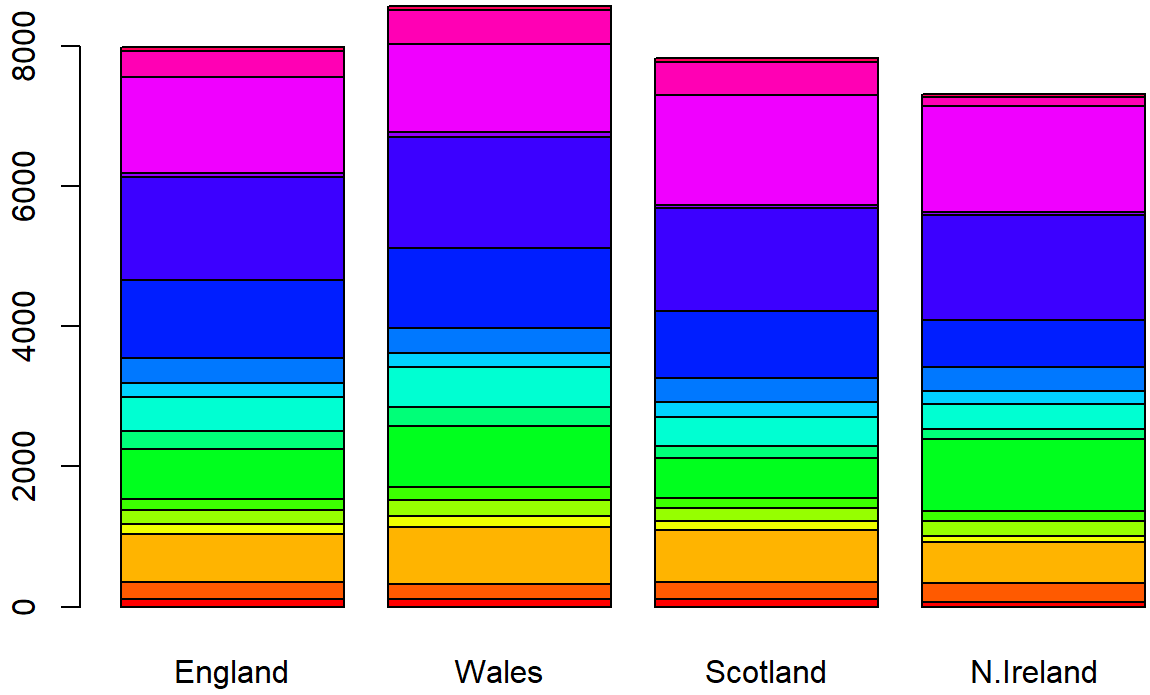
> > A2. I prefer adding row.names=1 to the read.csv() function because running x <- x[,-1] can cause data to disappear when you run it multiple times.

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



> Q3: Changing what optional argument in the above barplot() function results in the following plot?

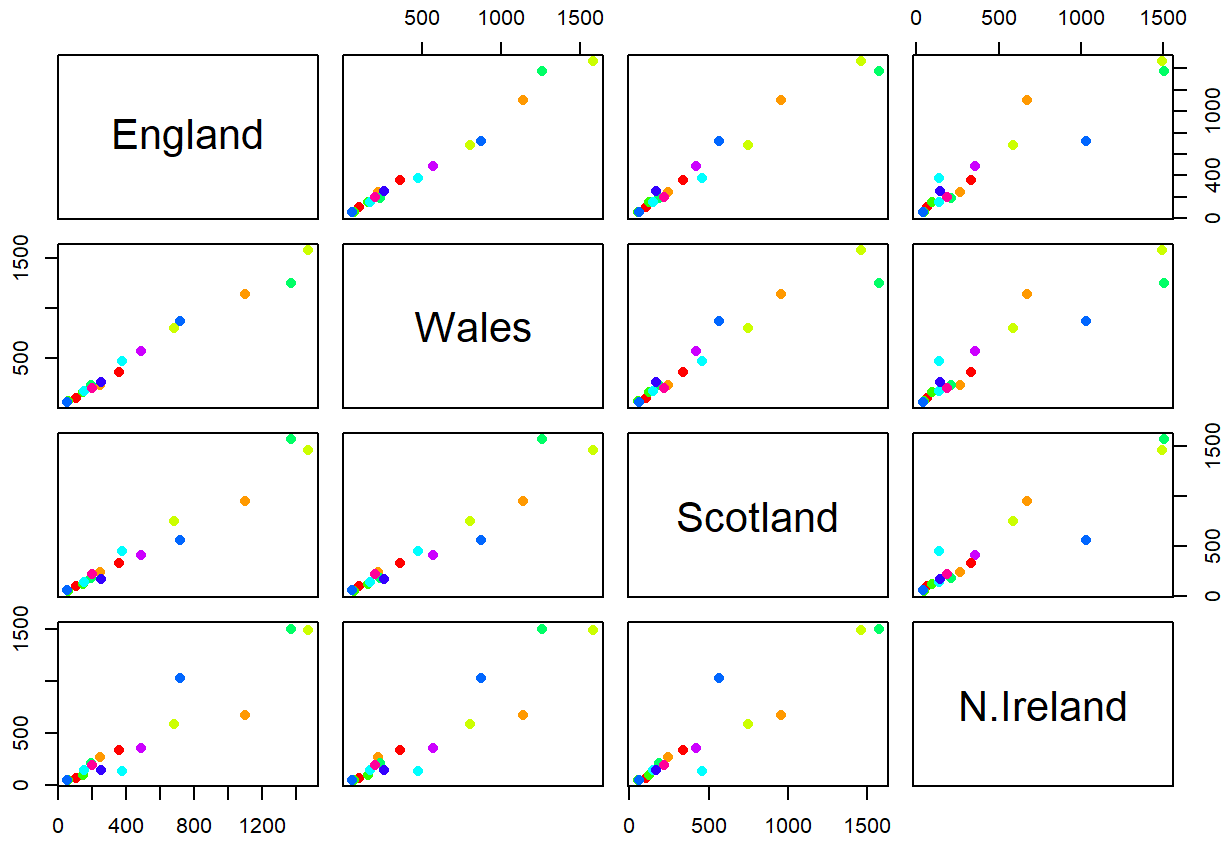
```
barplot( as.matrix(x), col=rainbow(nrow(x)), beside=FALSE)
```



> A3. Changing beside=T to beside=F will change the data layout from laying beside one another to instead being stacked.

> 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?

```
pairs(x, col=rainbow(10), pch=16)
```



> A5. A point that lies on the diagonal of the plot shows the distribution and relationship between a pair of variables (ex. England and Wales).

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

- A6. The main difference is that N. Ireland has more fresh potatoes and less alcoholic drinks.

PCA to the rescue

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

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

```
Importance of components:
                            PC1      PC2      PC3         PC4
Standard deviation     324.1502 212.7478 73.87622 3.176e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

How much variance is captured in 2 PCs? - 96.5%

> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
attributes(pca)
```

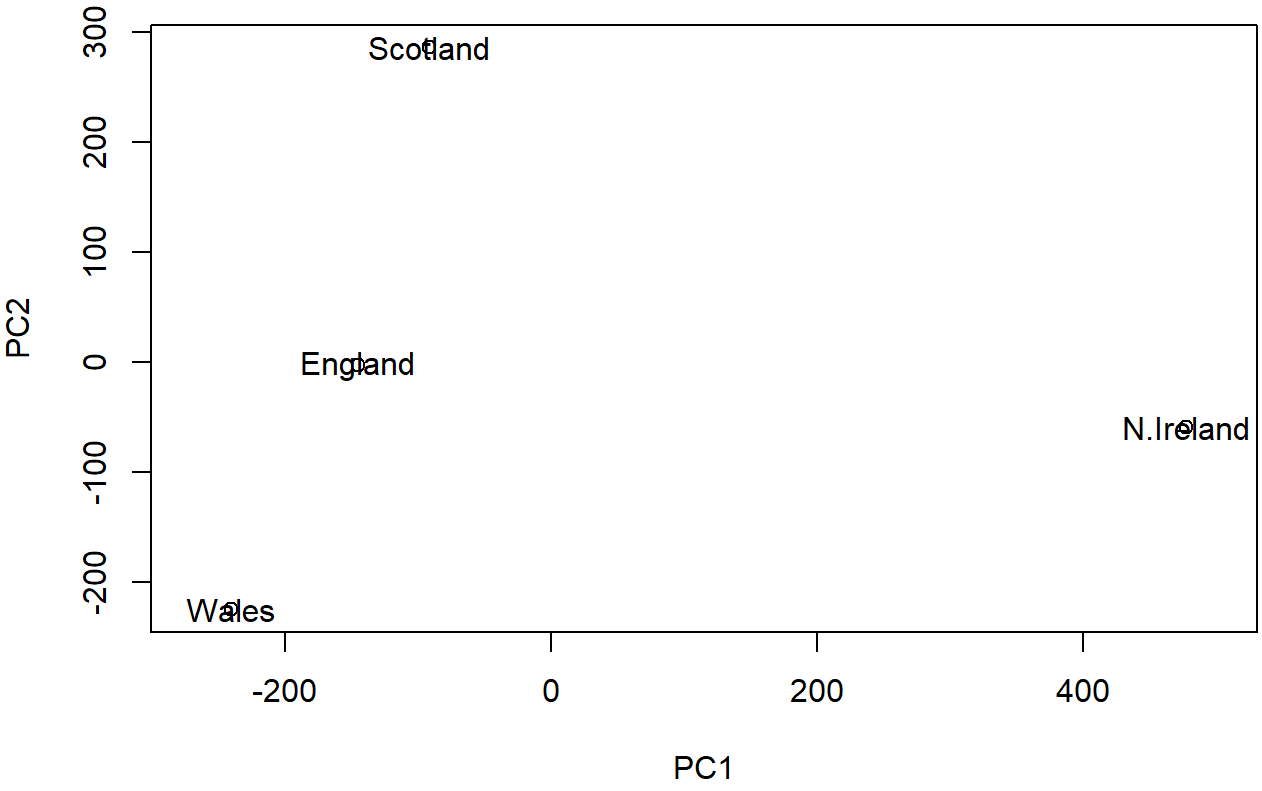
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

```
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
```

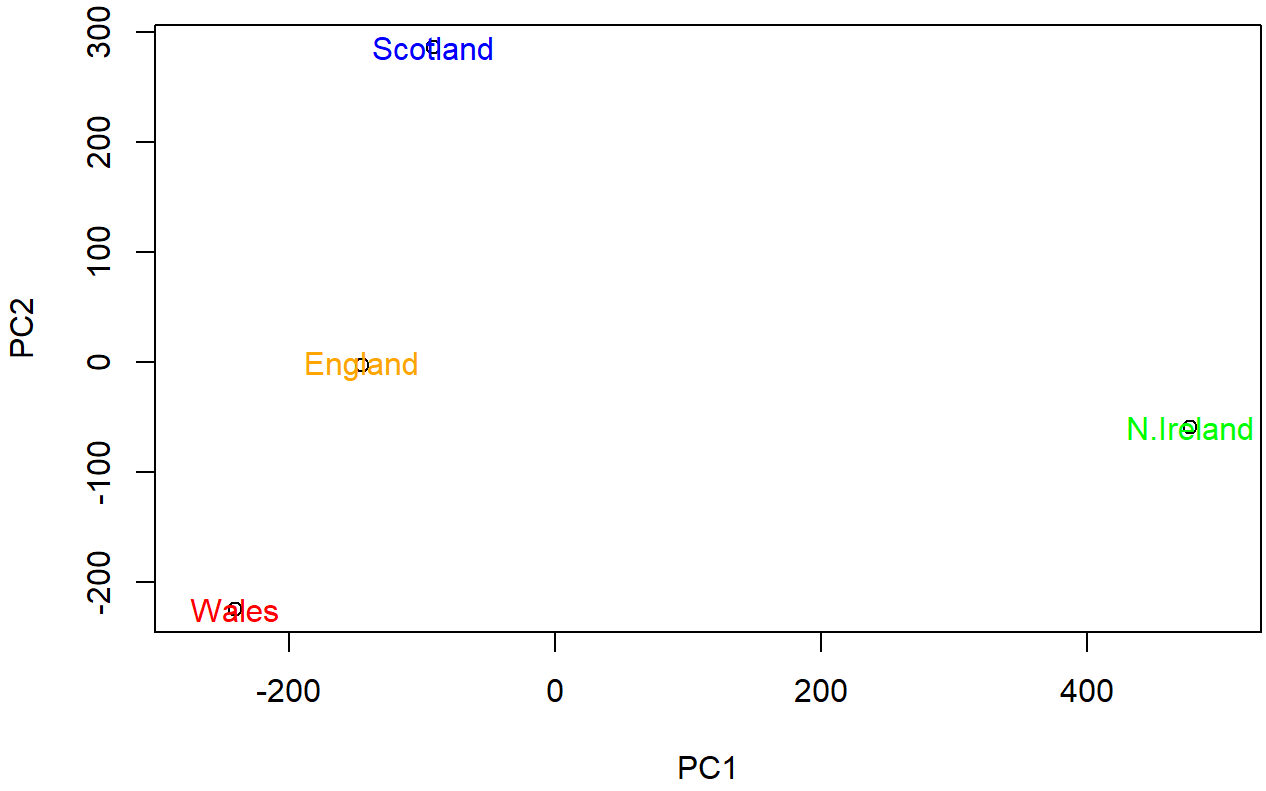
> A7. See completed code below

```
# Plot PC1 vs PC2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x))
```



> 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.

```
country_cols <- c("orange", "red", "blue", "green")
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x),
col=country_cols)
```



Another important result from PCA is how the original variables (in this case, the foods) contribute to the PCs. This is contained in the pca$rotation object - folks often call this the "loadings" or "contributions" to the PCs.

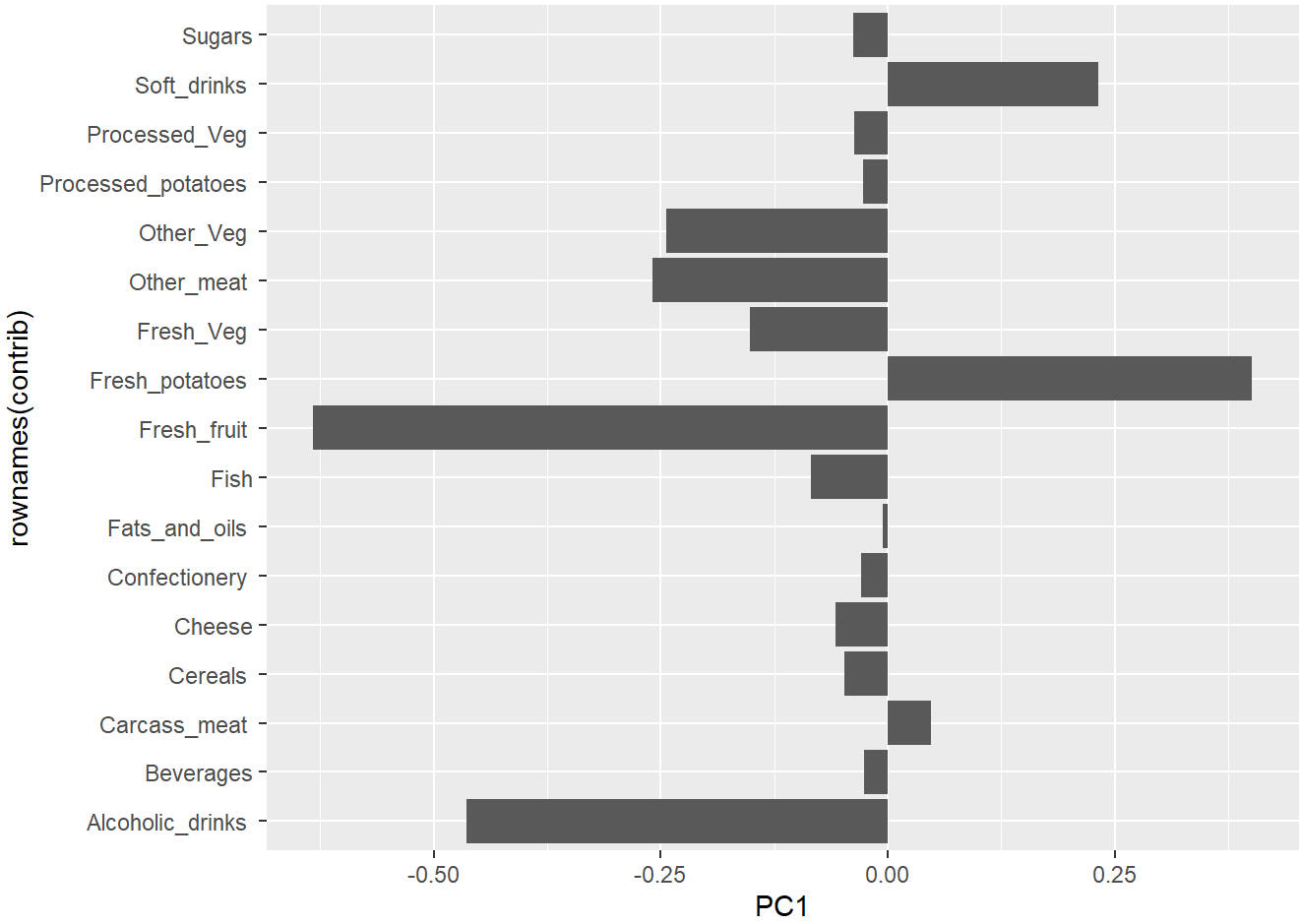
```
pca$rotation[,1]
```

```
          Cheese       Carcass_meat         Other_meat               Fish
     -0.056955380        0.047927628       -0.258916658       -0.084414983
   Fats_and_oils             Sugars     Fresh_potatoes          Fresh_Veg
     -0.005193623       -0.037620983        0.401402060       -0.151849942
       Other_Veg  Processed_potatoes      Processed_Veg         Fresh_fruit
     -0.243593729       -0.026886233       -0.036488269       -0.632640898
          Cereals          Beverages       Soft_drinks    Alcoholic_drinks
     -0.047702858       -0.026187756        0.232244140       -0.463968168
Confectionery
     -0.029650201
```

We can make a plot along PC1.

> Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about?

```
library(ggplot2)
contrib <- as.data.frame(pca$rotation)
ggplot(contrib) + aes(PC1, rownames(contrib)) + geom_col()
```



> A9. The two food groups fresh potatoes and soft drinks show to be in higher quantities and demand in N. Ireland whereas fresh fruit and alcoholic drinks show to be of less quantity compared to the other countries.

PCA of RNA-seq data

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

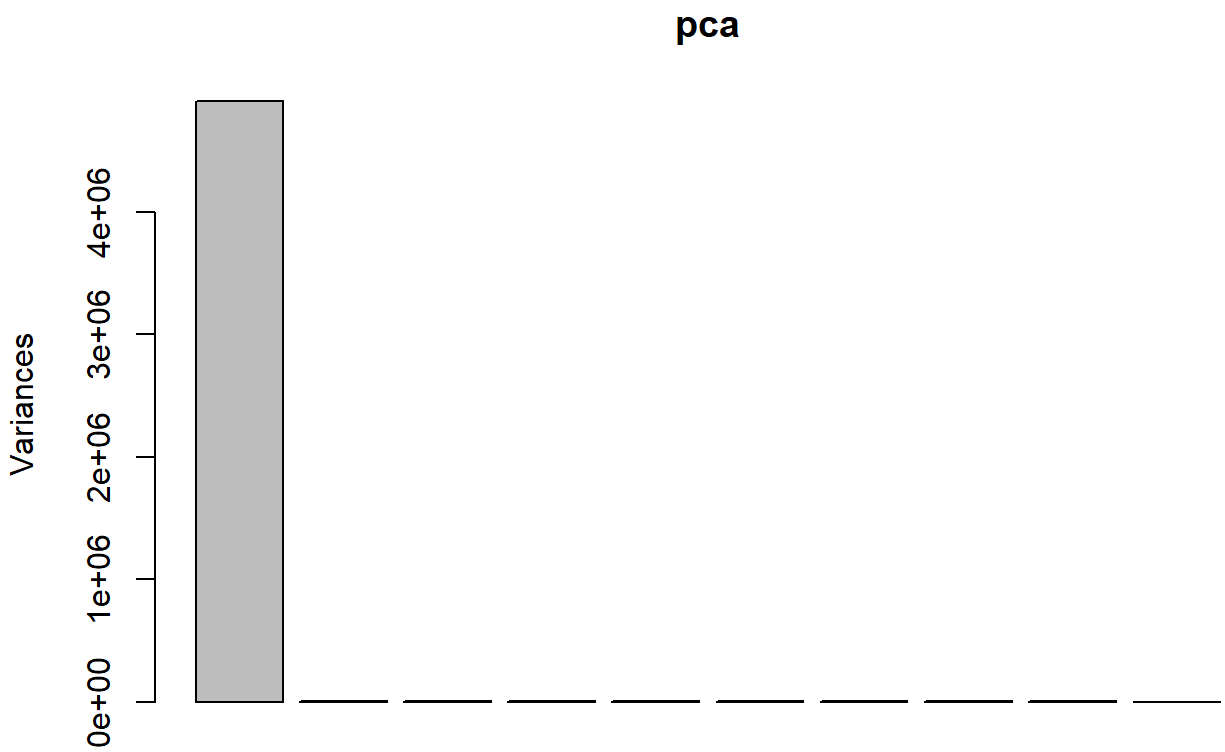
```
       wt1 wt2  wt3  wt4 wt5 ko1 ko2 ko3 ko4 ko5
gene1  439 458  408  429 420  90  88  86  90  93
gene2  219 200  204  210 187 427 423 434 433 426
gene3 1006 989 1030 1017 973 252 237 238 226 210
gene4  783 792  829  856 760 849 856 835 885 894
```

```
gene5  181 249  204  244 225 277 305 272 270 279
gene6  460 502  491  491 493 612 594 577 618 638
```

> Q10. How many genes and samples are in this data set?

- A10. There are 6 genes.

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

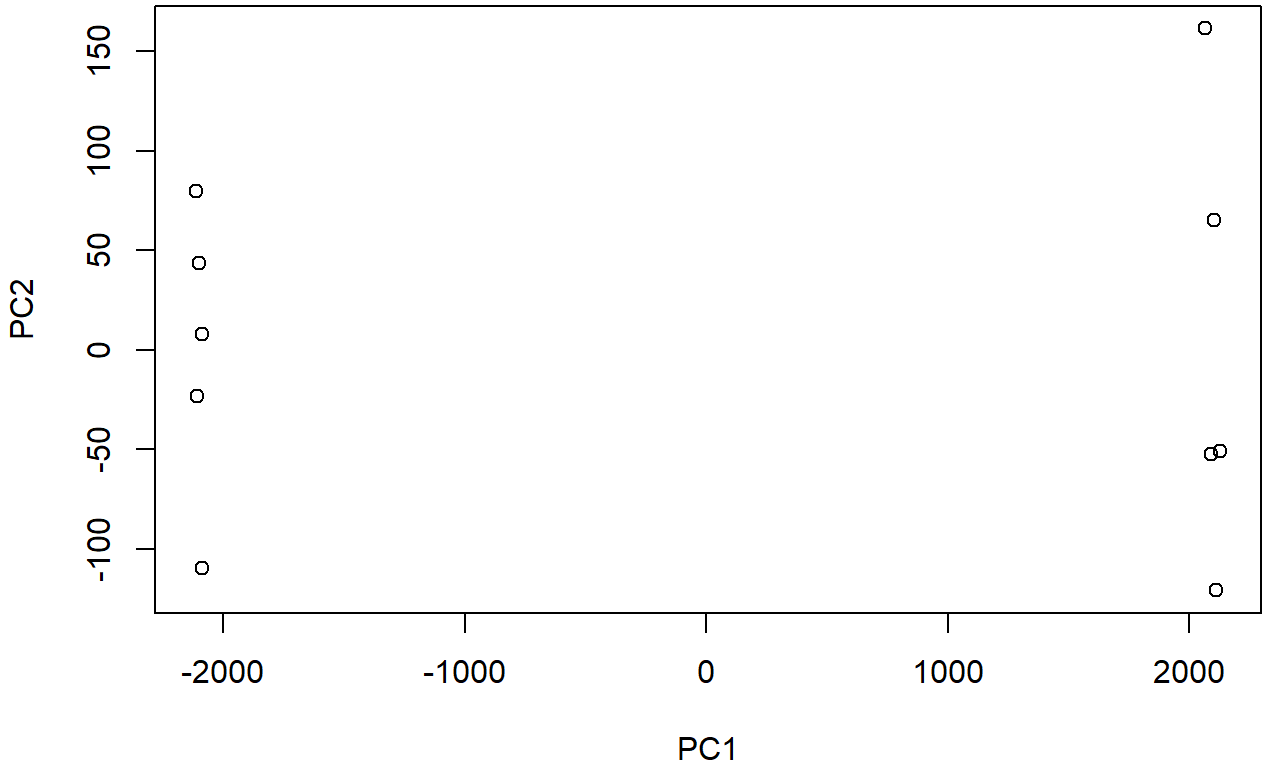



```
summary(pca)
```

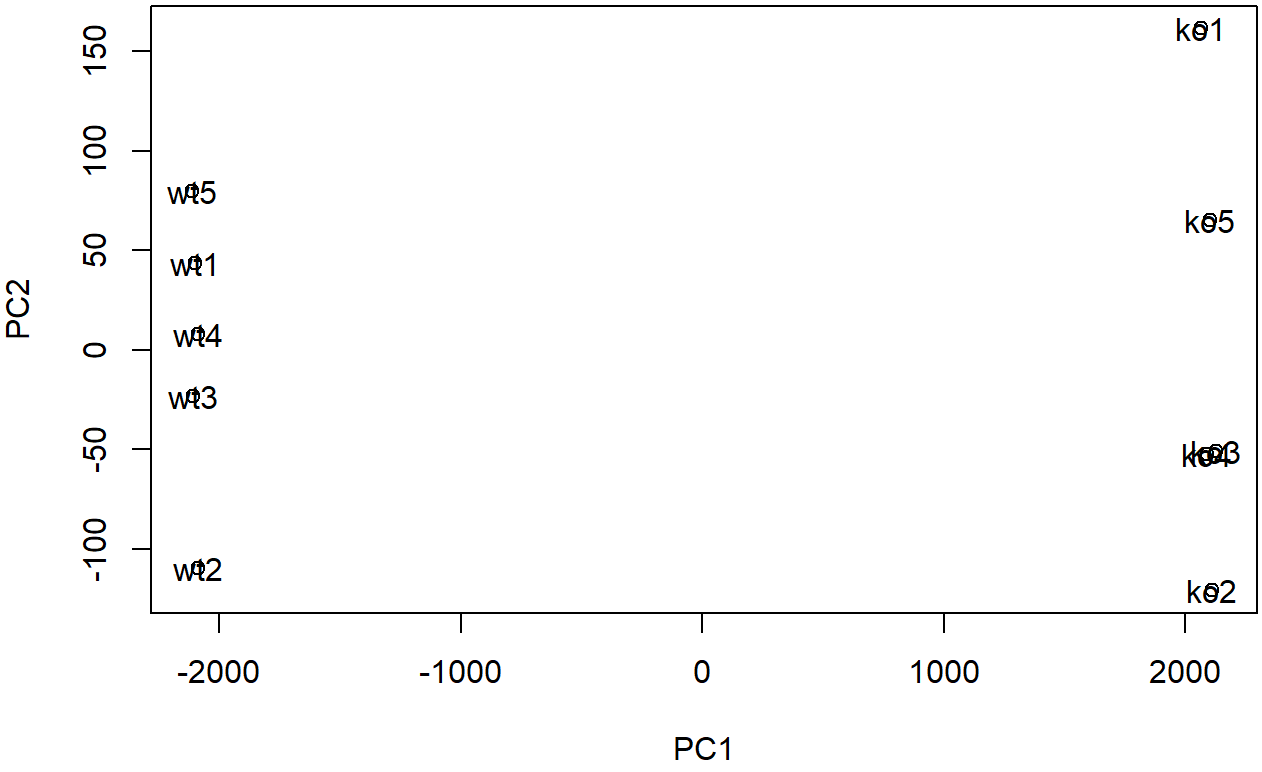
```
Importance of components:
                            PC1     PC2      PC3      PC4      PC5      PC6
Standard deviation     2214.2633 88.9209 84.33908 77.74094 69.66341 67.78516
Proportion of Variance    0.9917  0.0016  0.00144  0.00122  0.00098  0.00093
Cumulative Proportion     0.9917  0.9933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

Do our PCA plot of this RNA-Seq data

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




```
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
-50
-100
wt5
wt1
wt4
wt3
wt2
ko1
ko5
ko3
ko4
ko2
-2000
-1000
0
1000
2000
PC1