Machine Learning 1

Natalie Avina (PID: A15590695)

10/21/2021

Clustering Methods

#Kmeans clustering

The function in base R to do Kmeans clustering is called 'kmeans()'

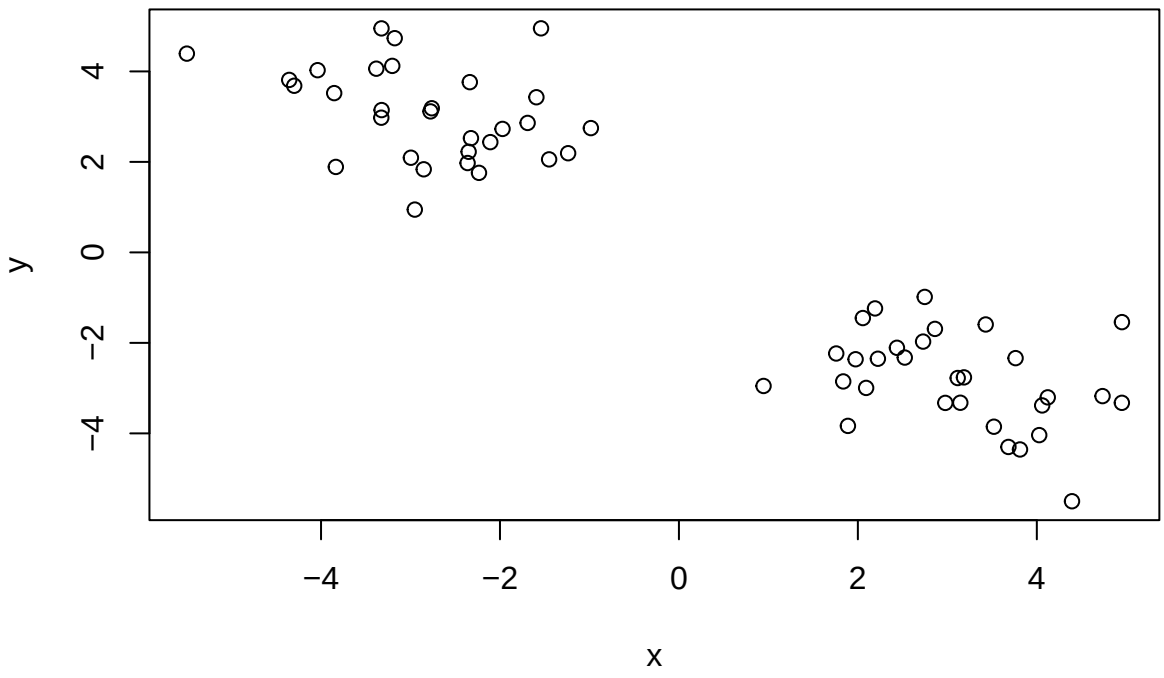
First we make up some data where we know what the answer should be

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

```
##                x          y
##  [1,] -2.3618043  1.9746104
##  [2,] -2.1079441  2.4367328
##  [3,] -4.3555117  3.8121394
##  [4,] -3.3825412  4.0599358
##  [5,] -5.4998455  4.3932989
##  [6,] -2.3243747  2.5241796
##  [7,] -3.3244469  4.9507217
##  [8,] -1.5929508  3.4288347
##  [9,] -0.9840591  2.7470775
## [10,] -3.8529167  3.5201743
## [11,] -2.3502714  2.2238016
## [12,] -4.2998285  3.6826973
## [13,] -3.1764222  4.7343931
## [14,] -2.9520634  0.9458956
## [15,] -3.8346407  1.8888255
## [16,] -4.0383207  4.0269618
## [17,] -2.9954975  2.0924507
## [18,] -3.3228907  3.1451120
## [19,] -1.2384260  2.1917071
## [20,] -1.6911466  2.8611765
## [21,] -2.7764560  3.1158355
## [22,] -2.8520947  1.8363363
## [23,] -1.4508469  2.0555968
## [24,] -3.3263693  2.9770681
## [25,] -1.9715510  2.7288733
## [26,] -2.2345324  1.7581757
## [27,] -1.5412409  4.9515995
## [28,] -2.7622361  3.1857095
## [29,] -2.3370534  3.7627385
## [30,] -3.2039381  4.1221974
```

```
## [31,]  4.1221974 -3.2039381
## [32,]  3.7627385 -2.3370534
## [33,]  3.1857095 -2.7622361
## [34,]  4.9515995 -1.5412409
## [35,]  1.7581757 -2.2345324
## [36,]  2.7288733 -1.9715510
## [37,]  2.9770681 -3.3263693
## [38,]  2.0555968 -1.4508469
## [39,]  1.8363363 -2.8520947
## [40,]  3.1158355 -2.7764560
## [41,]  2.8611765 -1.6911466
## [42,]  2.1917071 -1.2384260
## [43,]  3.1451120 -3.3228907
## [44,]  2.0924507 -2.9954975
## [45,]  4.0269618 -4.0383207
## [46,]  1.8888255 -3.8346407
## [47,]  0.9458956 -2.9520634
## [48,]  4.7343931 -3.1764222
## [49,]  3.6826973 -4.2998285
## [50,]  2.2238016 -2.3502714
## [51,]  3.5201743 -3.8529167
## [52,]  2.7470775 -0.9840591
## [53,]  3.4288347 -1.5929508
## [54,]  4.9507217 -3.3244469
## [55,]  2.5241796 -2.3243747
## [56,]  4.3932989 -5.4998455
## [57,]  4.0599358 -3.3825412
## [58,]  3.8121394 -4.3555117
## [59,]  2.4367328 -2.1079441
## [60,]  1.9746104 -2.3618043
```

```
plot(x)
```



Q. Can we use kmeans() to cluster this data setting k to 2 and nstart to 20

```
km <- kmeans(x, center=2, nstart= 20)
km
```

```
## K-means clustering with 2 clusters of sizes 30, 30
##
## Cluster means:
##           x         y
## 1 -2.804741  3.071162
## 2  3.071162 -2.804741
##
## 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 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 2
##
## Within cluster sum of squares by cluster:
## [1] 62.50699 62.50699
##  (between_SS / total_SS =  89.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Q. How many points are in each cluster?

```
km$size
```

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

q. What 'component of your result object details cluster assignment/membership?

```
km$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 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 2
```

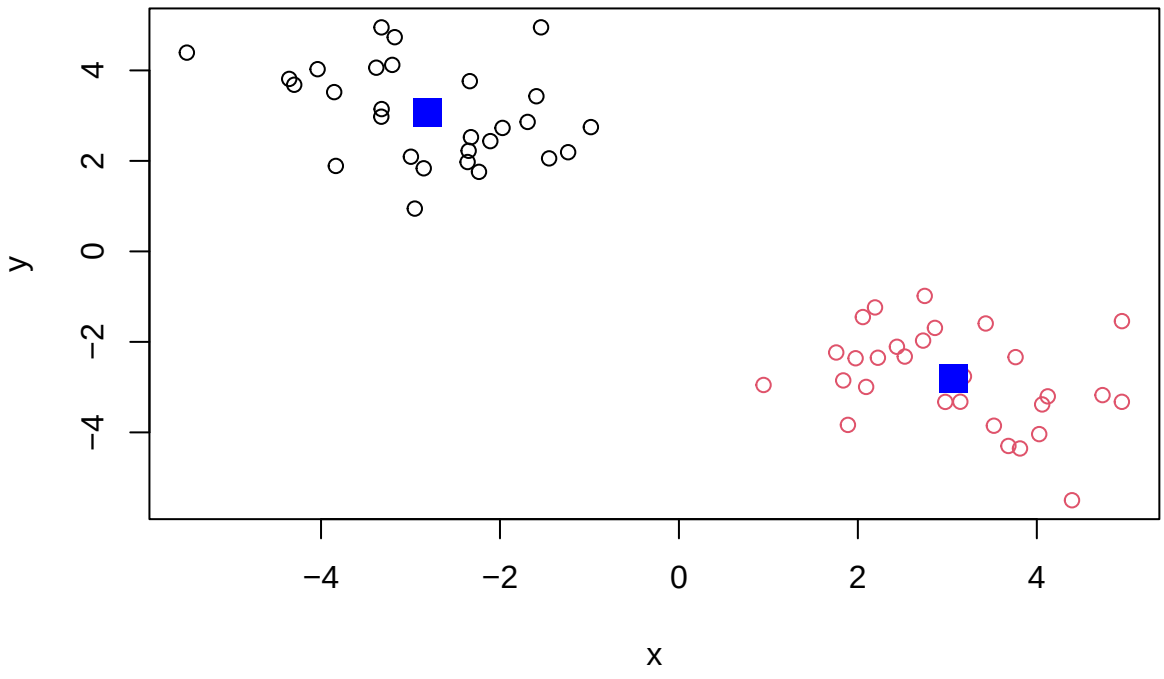
Q. What component of your result object details cluster center?

```
km$centers
```

```
##           x         y
## 1 -2.804741  3.071162
## 2  3.071162 -2.804741
```

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

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



Hierarchical Clustering

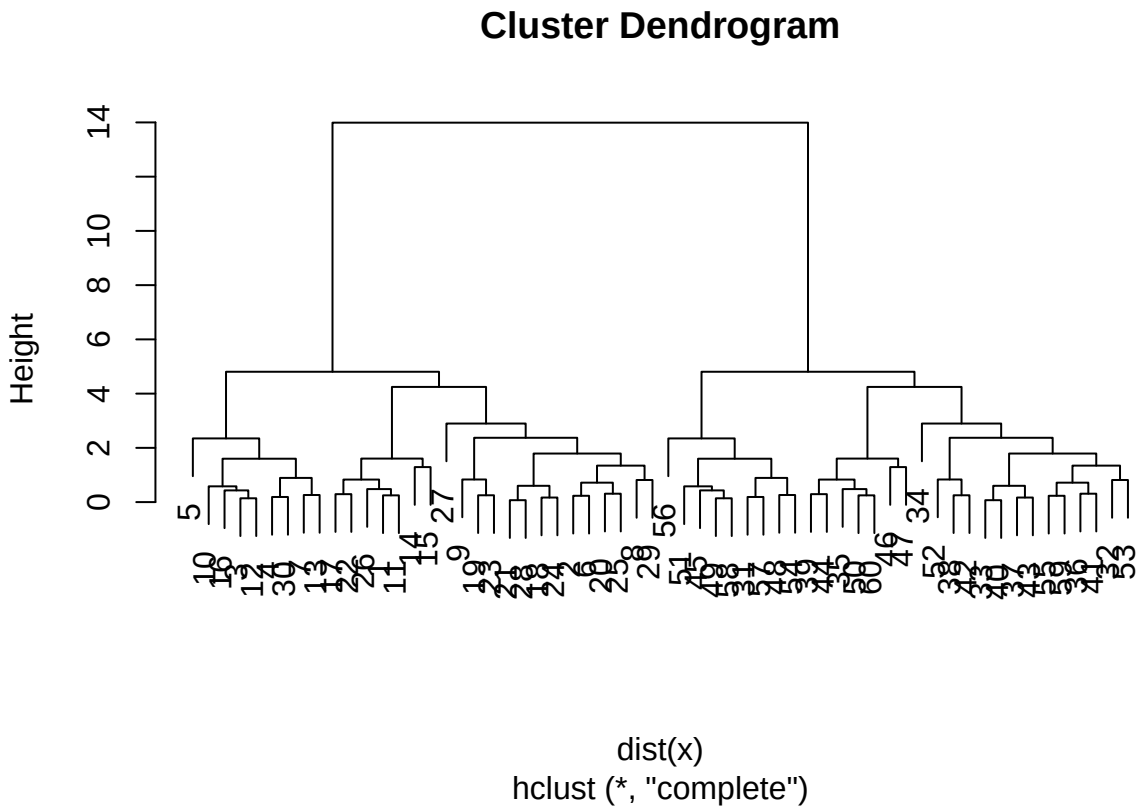
A limitation with k-means is that we have to specify K (or the number of clusters we want).

```
#hclust(d, method= , members=) where d is the output of 'dist()'

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

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

The plot method for hclust() is a dendrogram.

```
plot(hc)
```



To get our cluster membership vector we have to "cut" the tree where we want.

```
cutree(hc, h=6)
```

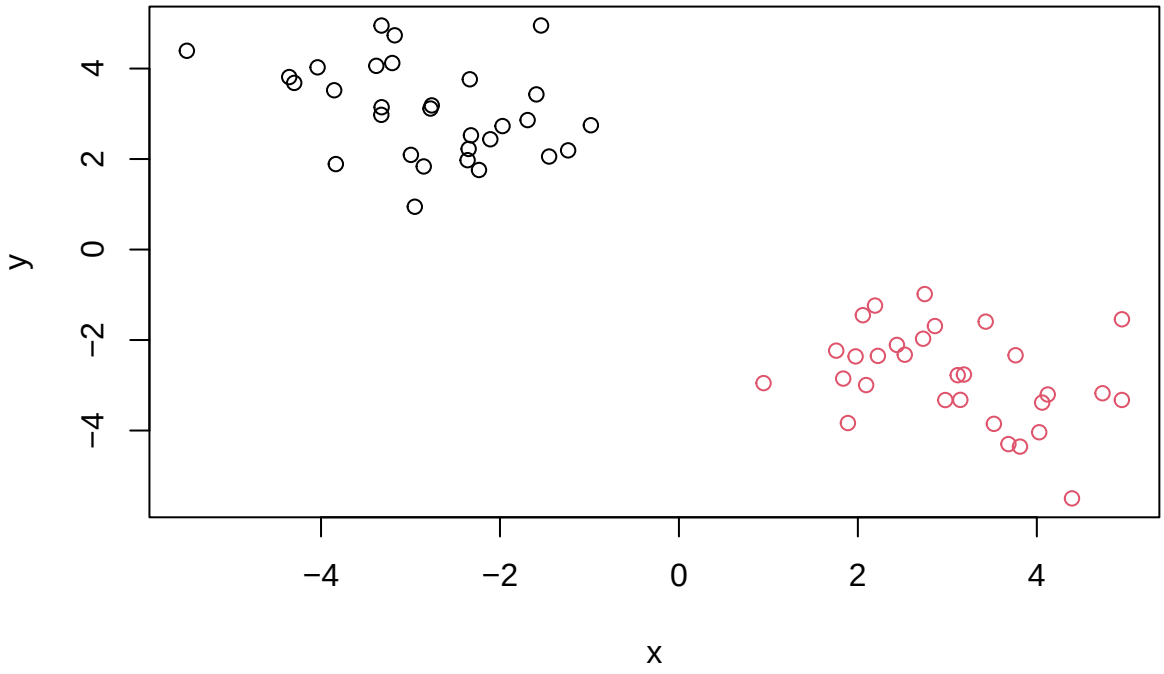
```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

#if you use cutree(hc, k= 'number') you can set k to a number of different groups so that the tree is cut at those groups rather than at a given height

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

Make results plot

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



PART 2 #Principal Component Analysis

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

```
y
```

```
##                     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
```

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(y)
```

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

```
ncol(y)
```

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

```
nrow(y)
```

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

We have one extra column that we need to fix so we have to fix that.

```
head(y)
```

```
##                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
```

```
 #views first six rows, tail() views last six rows
```

Be careful because certain commands will remove one column each time you run them such as:

```
#rownames(x) <- x[,1]
#x <- x[,-1]
#head(x)
```

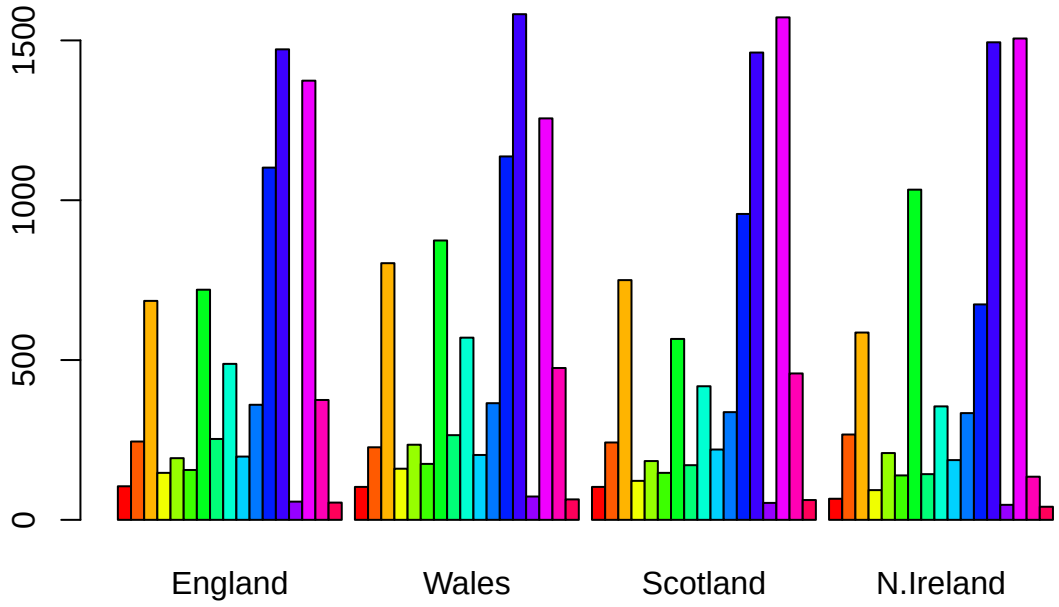
To fix this we use read.csv(url, row.names=1) instead back up top

> 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?

read.csv(url, row.names=1) because it will not remove a row every time you run the code. It is more robust because it can apply to other circumstances and won't eliminate data.

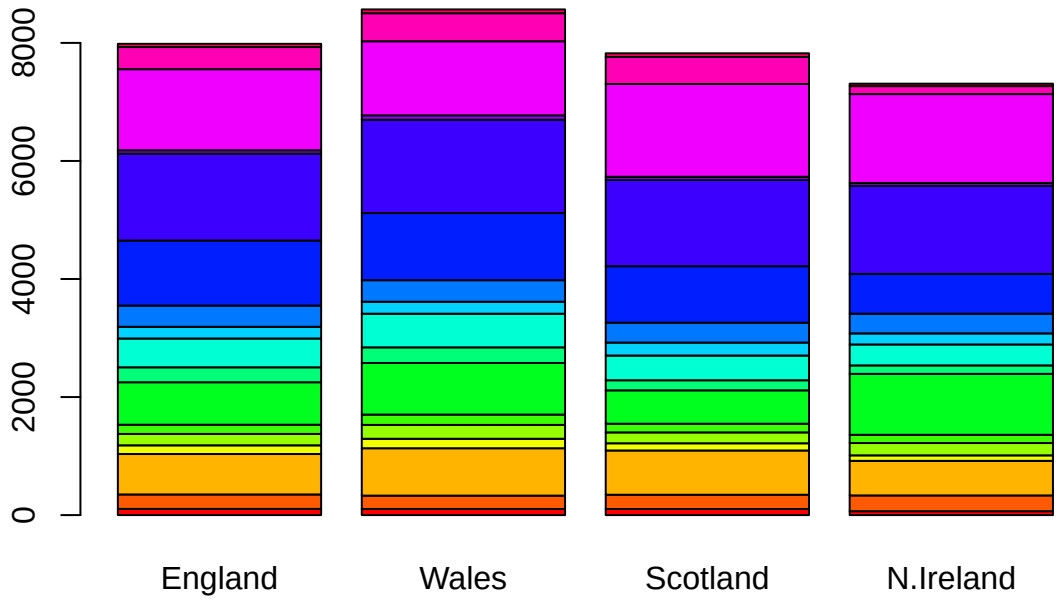
Barplot

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



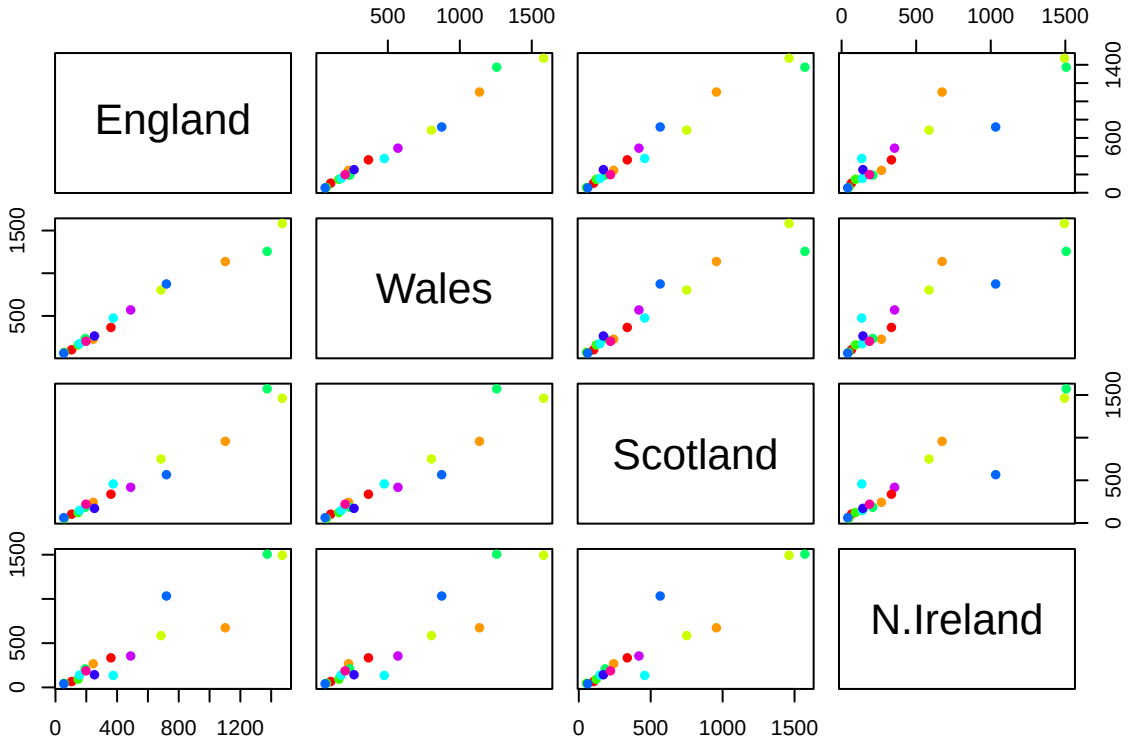
> Q3: Changing what optional argument in the above barplot() function results in the following plot?

```
#Beside=T makes it so that columns are juxtaposed as bars and FALSE makes it so that they are stacked
barplot(as.matrix(y), beside=FALSE, col=rainbow(nrow(y)))
```



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(y, col=rainbow(10), pch=16)
```



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

They consume more potatoes and less fresh fruit than other countries

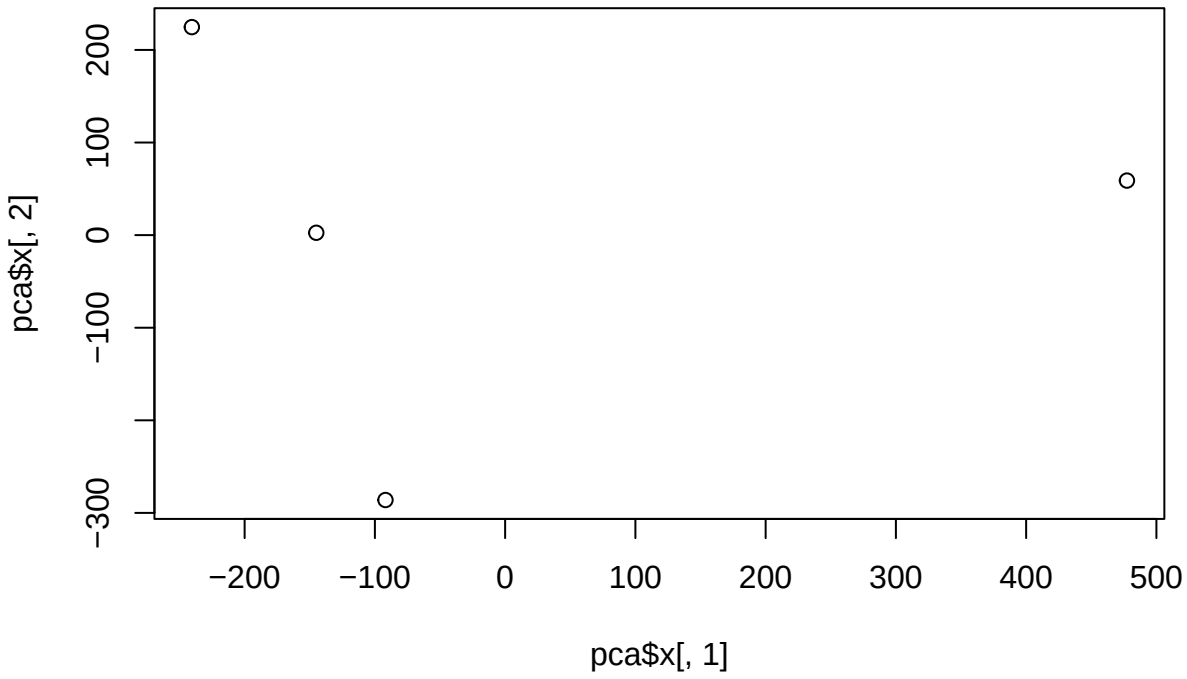
```
pca <- prcomp( t(y) )
summary(pca)
```

```
## 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
```

```
attributes(pca)
```

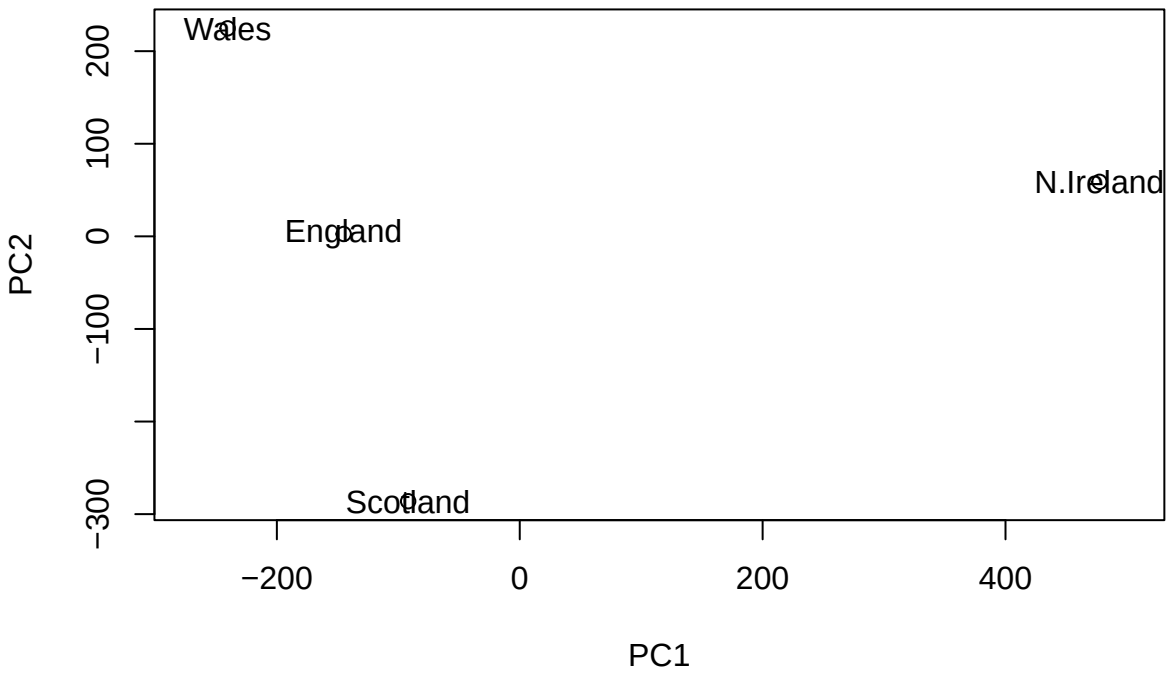
```
## $names
## [1] "sdev"     "rotation" "center"   "scale"    "x"
##
## $class
## [1] "prcomp"
```

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



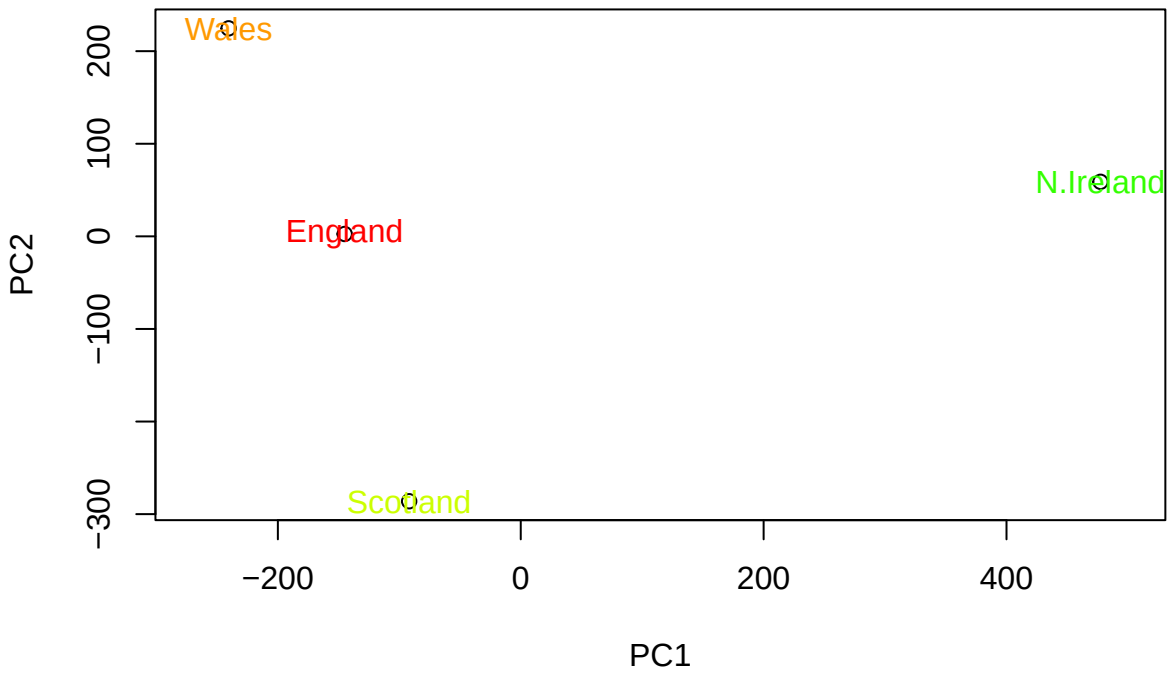
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points

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



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

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(y), col=rainbow(10))
```



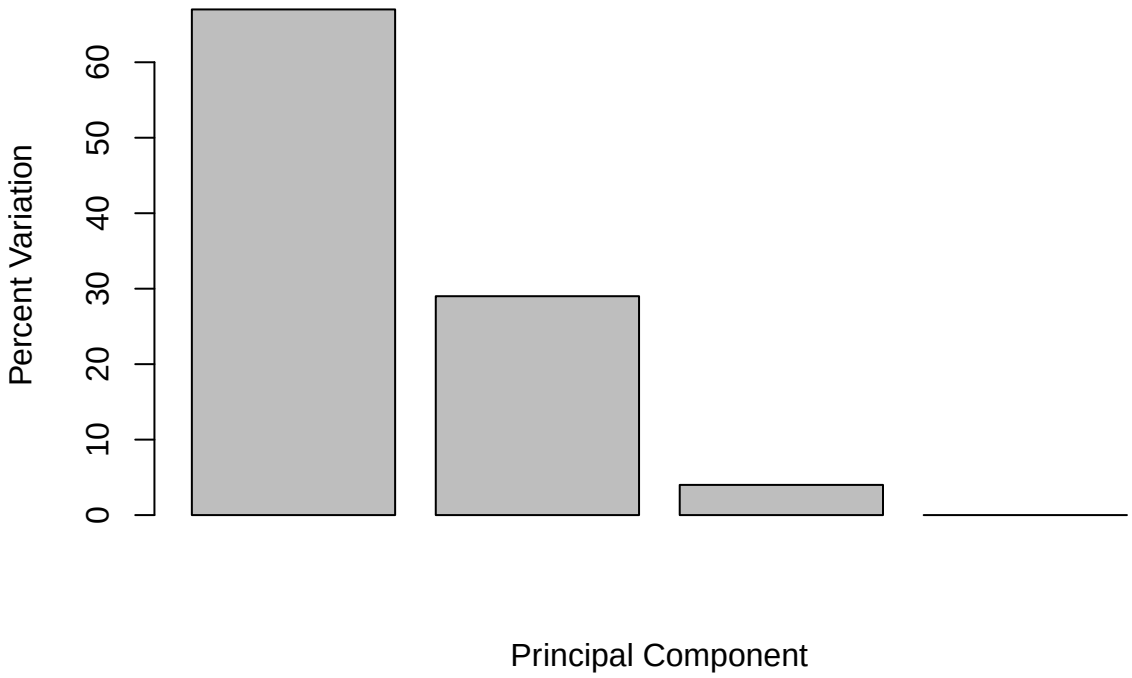

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

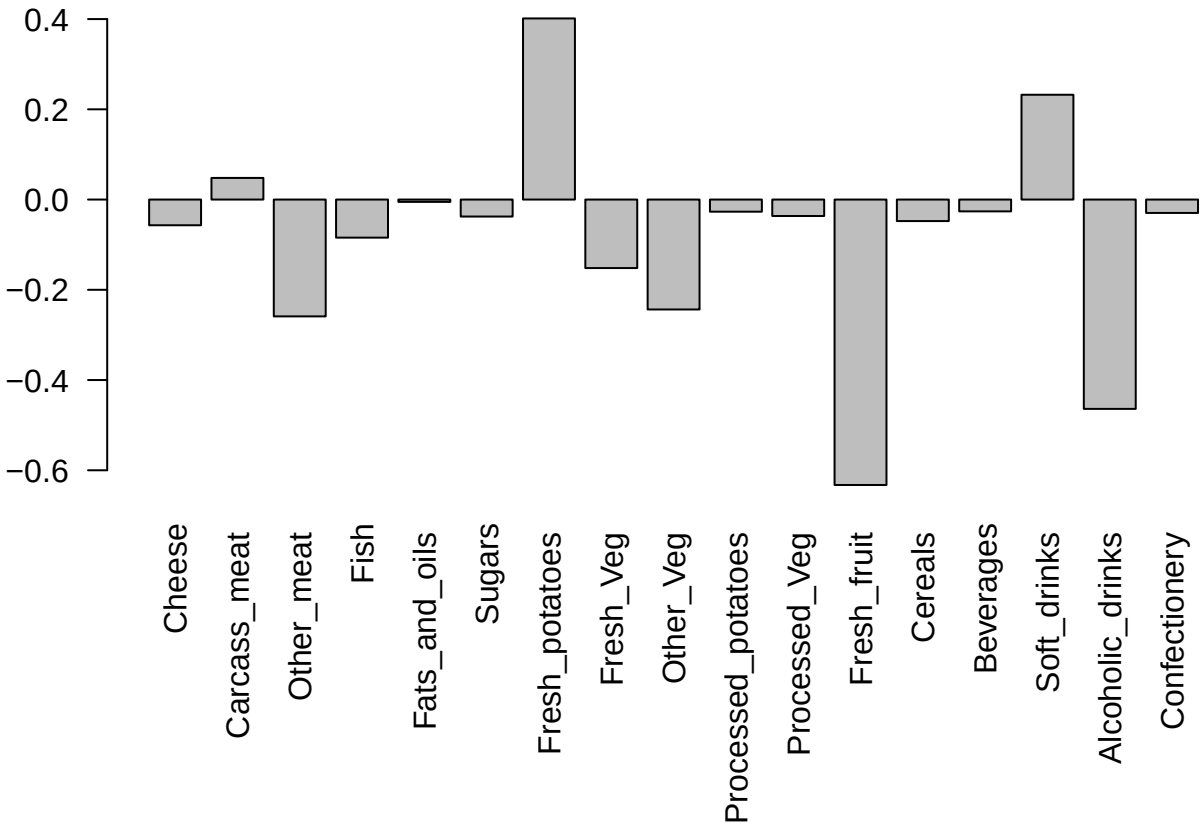
```
z <- summary(pca)
z$importance
```

```
##                             PC1       PC2      PC3          PC4
## Standard deviation     324.15019 212.74780 73.87622 4.188568e-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")
```

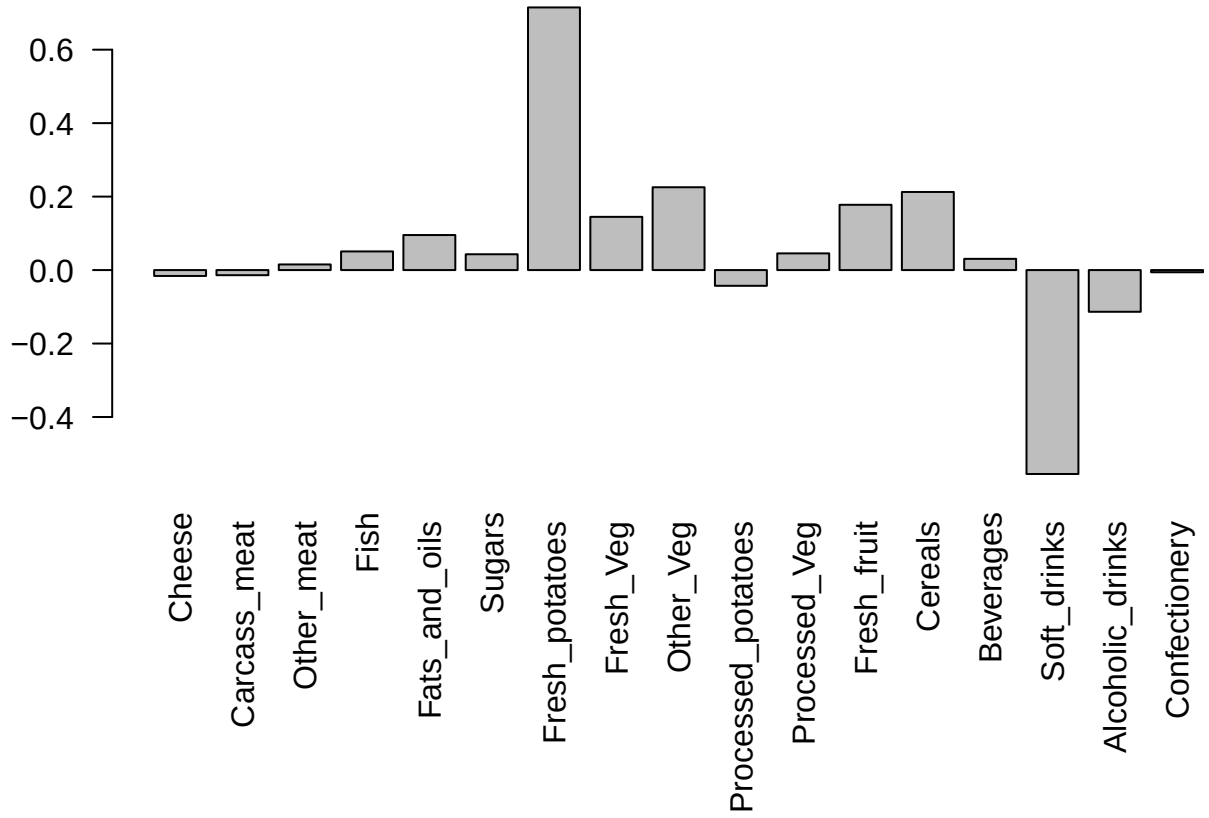



```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



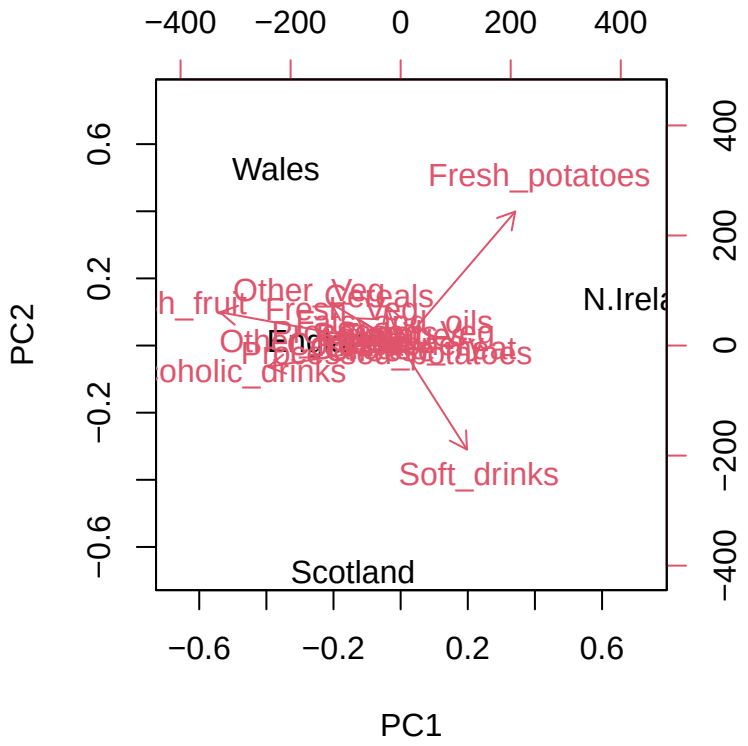
Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about? The two food groups are Potatoes and Soft Drinks. PC2 mainly tells us about the second highest amount of variation between the countries.

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



#Another way to do this is to do a BC plot

```
## The inbuilt biplot() can be useful for small datasets
biplot(pca)
```



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?

```
#The number of genes is the number of rows and the number of samples are columns
ncol(rna.data)
```

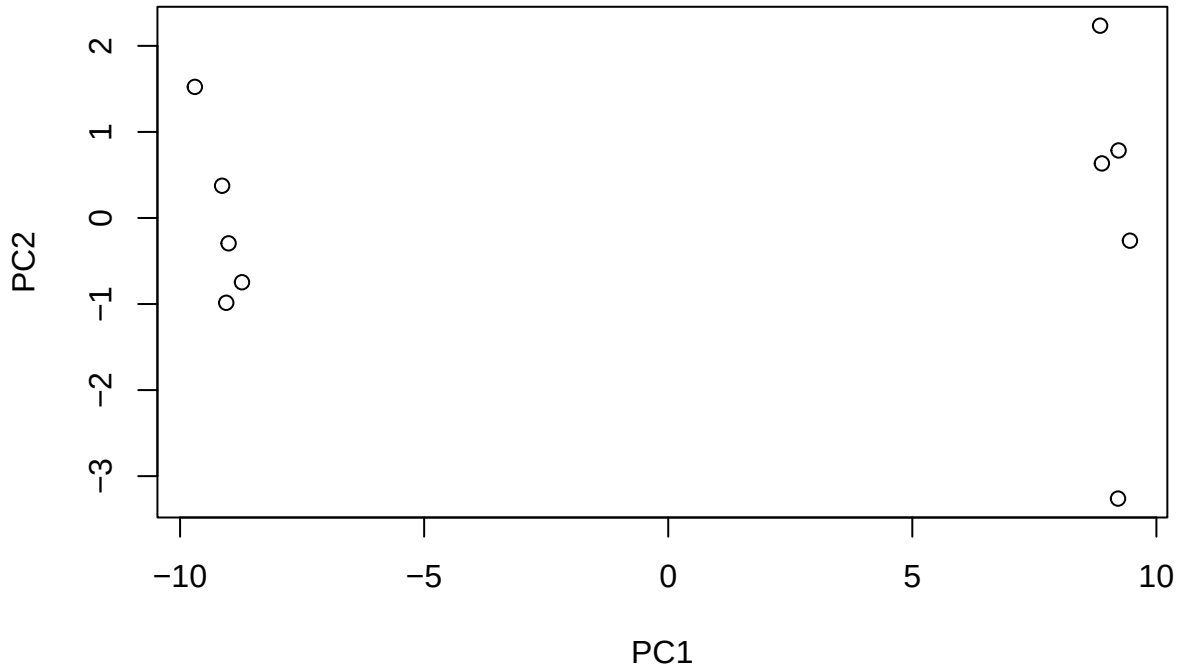
```
## [1] 10
```

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

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

There are 100 genes and 10 samples.

```
## Again we have to take the transpose of our data
pca <- prcomp(t(rna.data), scale=TRUE)

## Simple un polished plot of pc1 and pc2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```

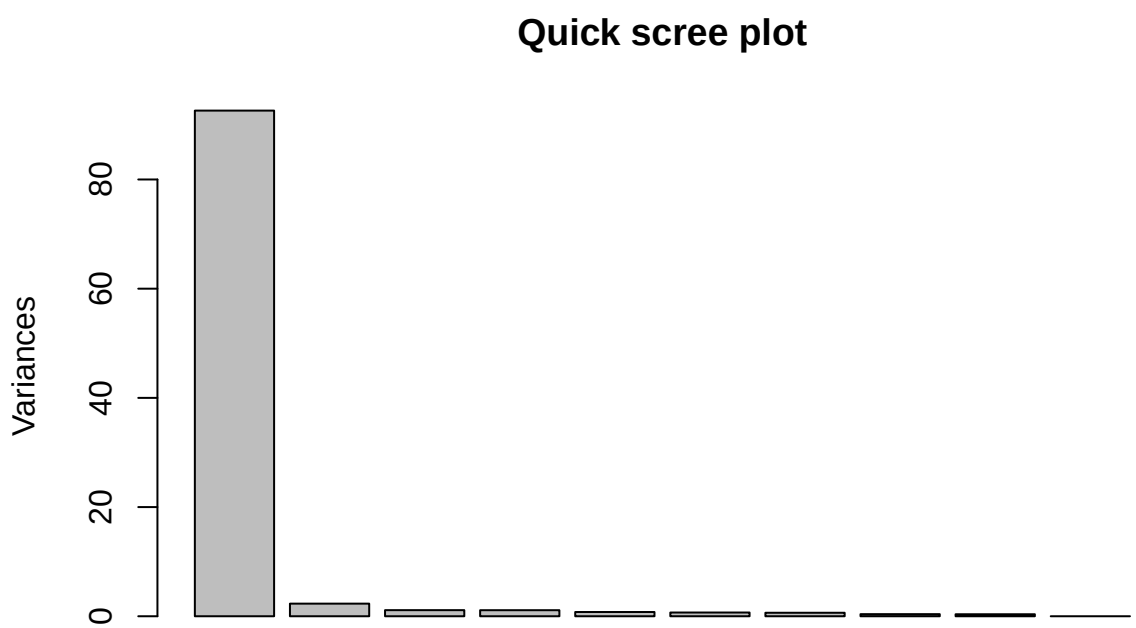



```
summary(pca)
```

```
## 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
```

A barplot of this data:

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

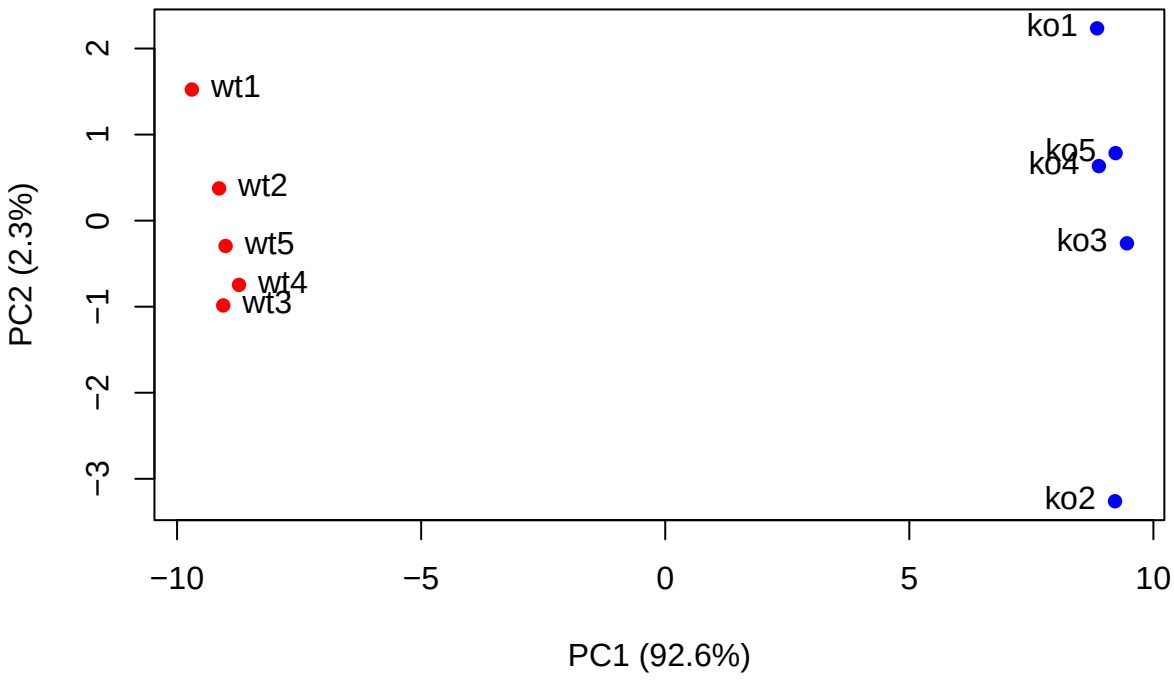



```
## Variance captured per PC
pca.var <- pca$sdev^2

## Percent variance is often more informative to look at
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)
pca.var.per
```

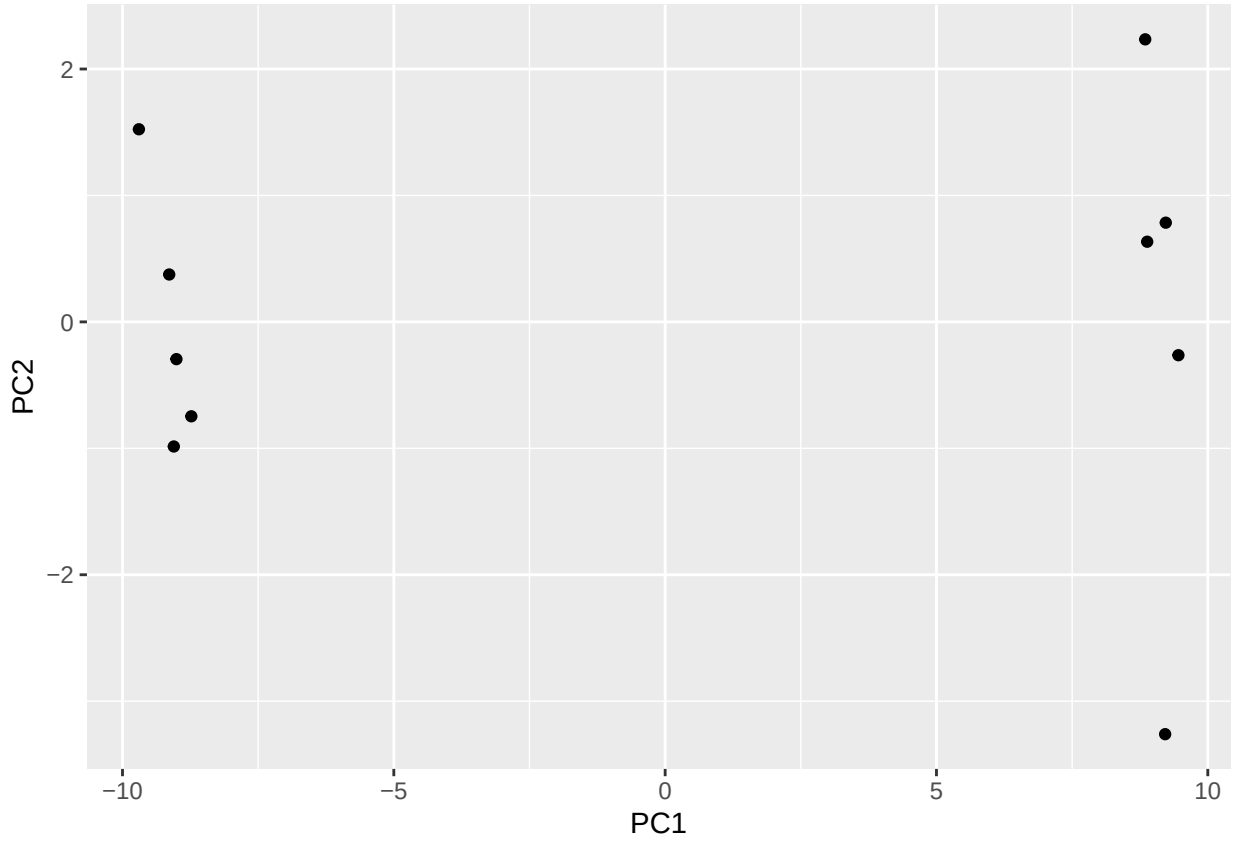
```
##  [1] 92.6  2.3  1.1  1.1  0.8  0.7  0.6  0.4  0.4  0.0
```

Beautifying our plot

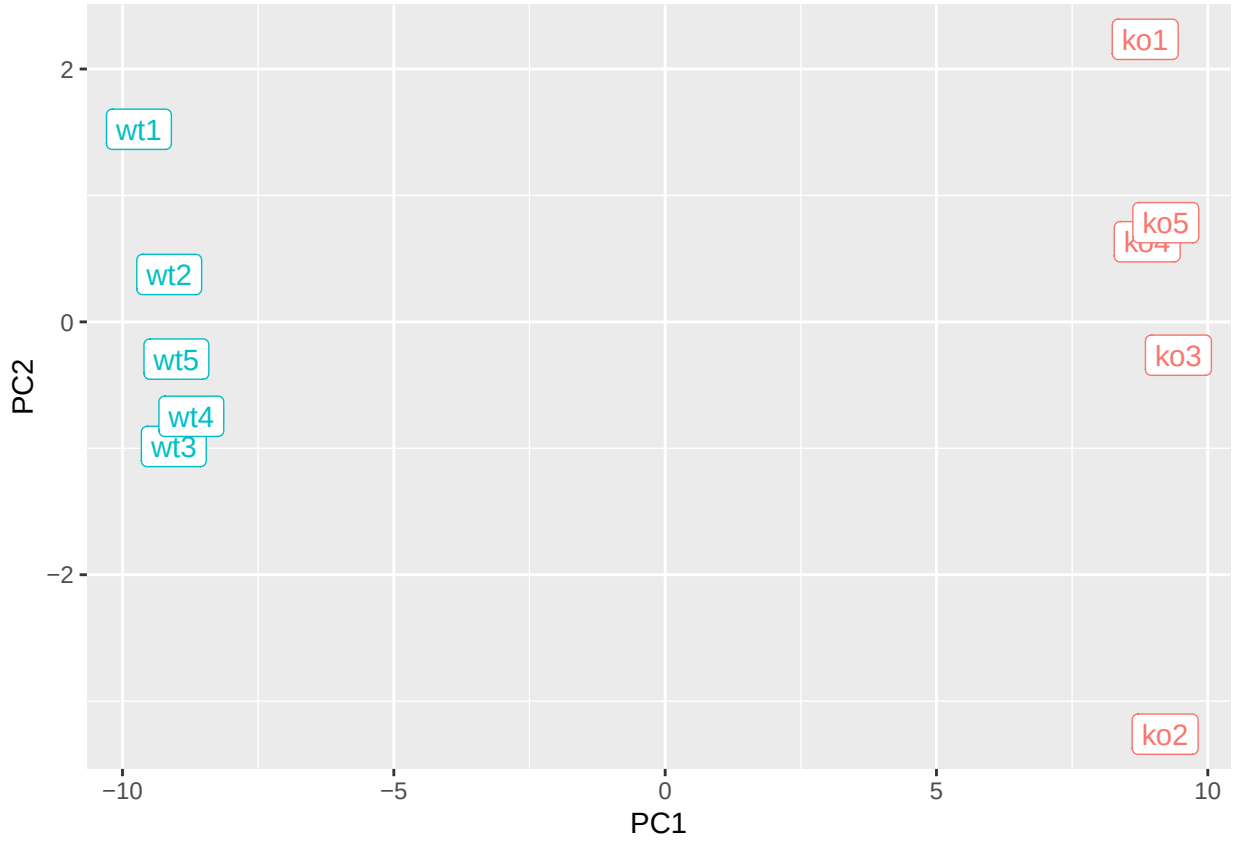
```
## A vector of colors for wt and ko samples
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

plot(pca$x[,1], pca$x[,2], col=colvec, pch=16,
     xlab=paste0("PC1 (", pca.var.per[1], "%)"),
     ylab=paste0("PC2 (", pca.var.per[2], "%)"))

text(pca$x[,1], pca$x[,2], labels = colnames(rna.data), pos=c(rep(4,5), rep(2,5)))
```




```
library(ggplot2)

df <- as.data.frame(pca$x)

# Our first basic plot
ggplot(df) +
  aes(PC1, PC2) +
  geom_point()
```

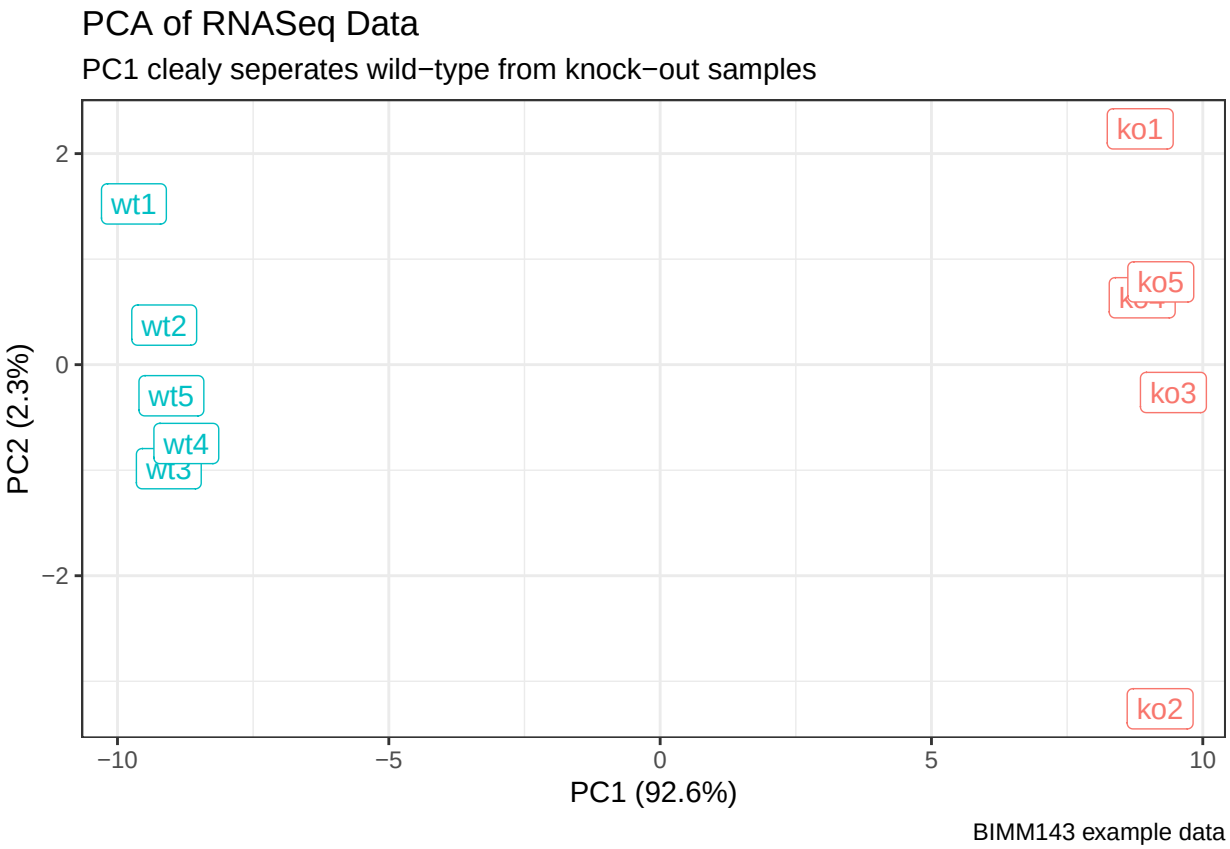



```
# Add a 'wt' and 'ko' "condition" column
df$samples <- colnames(rna.data)
df$condition <- substr(colnames(rna.data),1,2)

p <- ggplot(df) +
        aes(PC1, PC2, label=samples, col=condition) +
        geom_label(show.legend = FALSE)
p
```



#Finalized plot with new fixes

```
p + labs(title="PCA of RNASeq Data",
       subtitle = "PC1 clealy seperates wild-type from knock-out samples",
       x=paste0("PC1 (", pca.var.per[1], "%)"),
       y=paste0("PC2 (", pca.var.per[2], "%)"),
       caption="BIMM143 example data") +
     theme_bw()
```


PCA of RNASeq Data
PC1 clealy seperates wild−type from knock−out samples
ko1
wt1
ko5
wt2
ko3
wt5
wt4
wt3
ko2
2
0
−2
PC2 (2.3%)
−10
−5
0
5
10
PC1 (92.6%)
BIMM143 example data