---

title: “Machine learning 1” author: “Anel Valdez” date: 10/21/2021 output:

First up is clustering methods! # Kmeans clustering

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

first make up some data wher 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,] -3.306511  2.048764
##  [2,] -4.187927  2.878163
##  [3,] -3.496496  3.274347
##  [4,] -0.433485  1.729501
##  [5,] -2.631104  4.406378
##  [6,] -2.447662  4.573213
##  [7,] -3.758142  2.270311
##  [8,] -2.908051  2.700681
##  [9,] -3.656277  3.714426
## [10,] -2.754623  2.348252
## [11,] -3.707112  2.343369
## [12,] -4.326925  2.726694
## [13,] -2.086769  1.981343
## [14,] -3.786023  4.407221
## [15,] -3.140338  2.555675
## [16,] -3.712581  2.518554
## [17,] -4.031471  2.008597
## [18,] -3.444590  1.982588
## [19,] -2.529869  2.523735
## [20,] -2.409553  1.313215
## [21,] -1.660990  3.405168
## [22,] -2.202921  2.151118
## [23,] -3.245948  4.169960
## [24,] -1.397953  1.700057
## [25,] -3.610214  3.455356
## [26,] -3.476578  3.987535
## [27,] -4.589985  3.235326
## [28,] -3.815253  1.900383
## [29,] -2.689867  4.292147
## [30,] -3.043591  3.687220
## [31,]  3.687220 -3.043591
## [32,]  4.292147 -2.689867
## [33,]  1.900383 -3.815253
## [34,]  3.235326 -4.589985
## [35,]  3.987535 -3.476578
## [36,]  3.455356 -3.610214
## [37,]  1.700057 -1.397953
## [38,]  4.169960 -3.245948
## [39,]  2.151118 -2.202921
## [40,]  3.405168 -1.660990
```

```
## [41,]  1.313215 -2.409553
## [42,]  2.523735 -2.529869
## [43,]  1.982588 -3.444590
## [44,]  2.008597 -4.031471
## [45,]  2.518554 -3.712581
## [46,]  2.555675 -3.140338
## [47,]  4.407221 -3.786023
## [48,]  1.981343 -2.086769
## [49,]  2.726694 -4.326925
## [50,]  2.343369 -3.707112
## [51,]  2.348252 -2.754623
## [52,]  3.714426 -3.656277
## [53,]  2.700681 -2.908051
## [54,]  2.270311 -3.758142
## [55,]  4.573213 -2.447662
## [56,]  4.406378 -2.631104
## [57,]  1.729501 -0.433485
## [58,]  3.274347 -3.496496
## [59,]  2.878163 -4.187927
## [60,]  2.048764 -3.306511
```

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

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

```
## K-means clustering with 2 clusters of sizes 30, 30
##
## Cluster means:
##          x        y
## 1 -3.08296  2.87631
## 2  2.87631 -3.08296
##
## 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 2 2
##
## Within cluster sum of squares by cluster:
## [1] 50.40012 50.40012
##  (between_SS / total_SS =  91.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Q. How many points 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 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 2 2 2 2 2 2 2 2 2 2 2
```

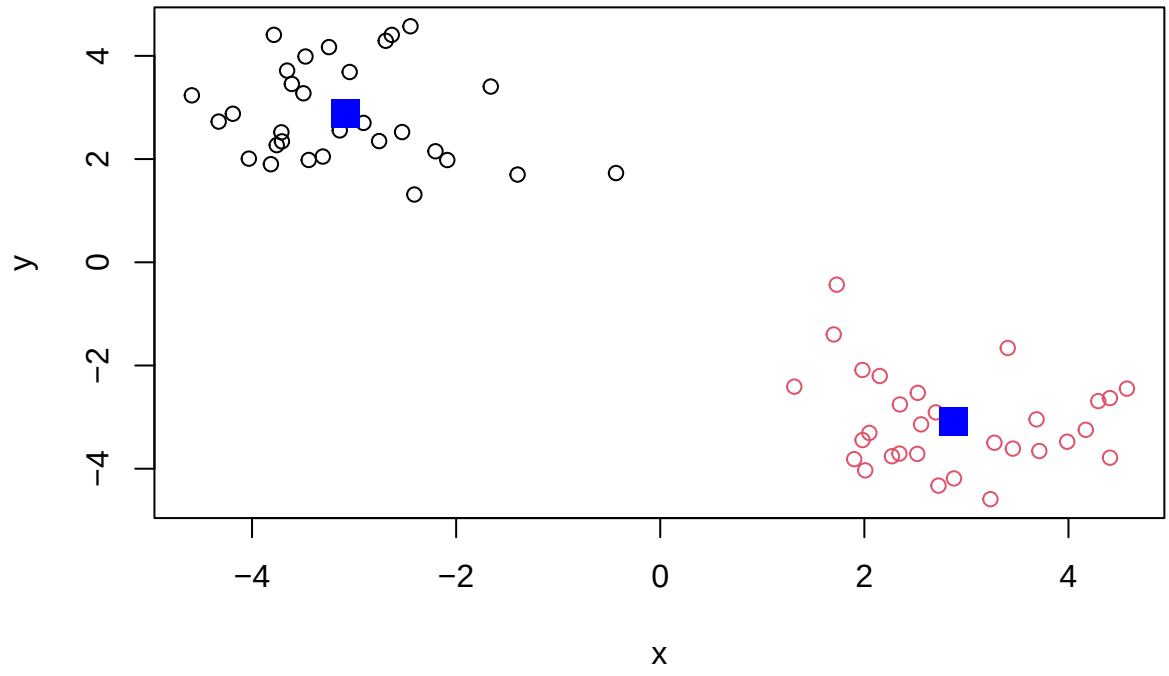
Q. What 'component' of your result object details cluster center?

```
km$centers
```

```
##          x        y
## 1 -3.08296  2.87631
## 2  2.87631 -3.08296
```

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)
```



#hclust

A big limitation with k-means is that we have to tell it K (the number of clsuters we want).

Analyze the same data with hclust()

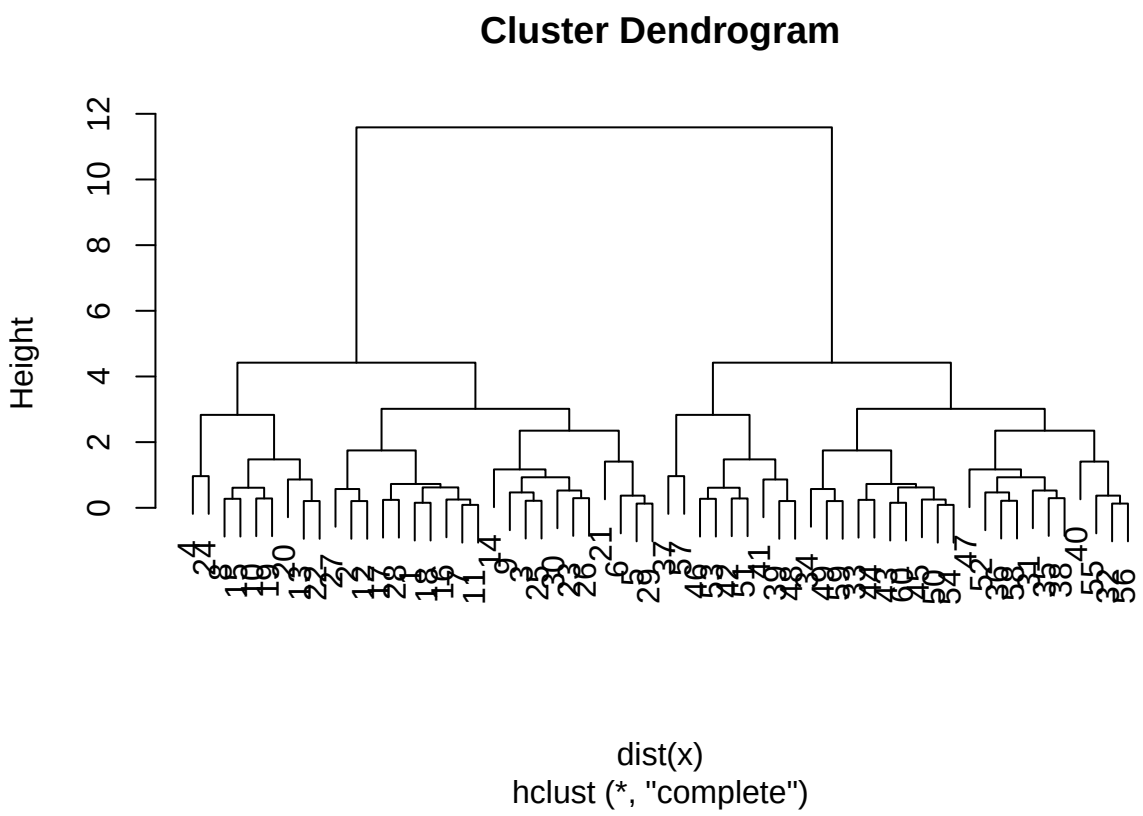
Demonstarte the use of dist(), hclust(), plot() and cutree() functions to do clustering, Generate dedrograms and return clsuter assignment/ membership vector...

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

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

There is a plot method for hclust result objects. Let's see it.

```
plot(hc)
```



To get our cluster membership vector we have to do a little more work. We have to "cut" the tree where we think it makes sense. For this we use the 'cutree()'function.

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

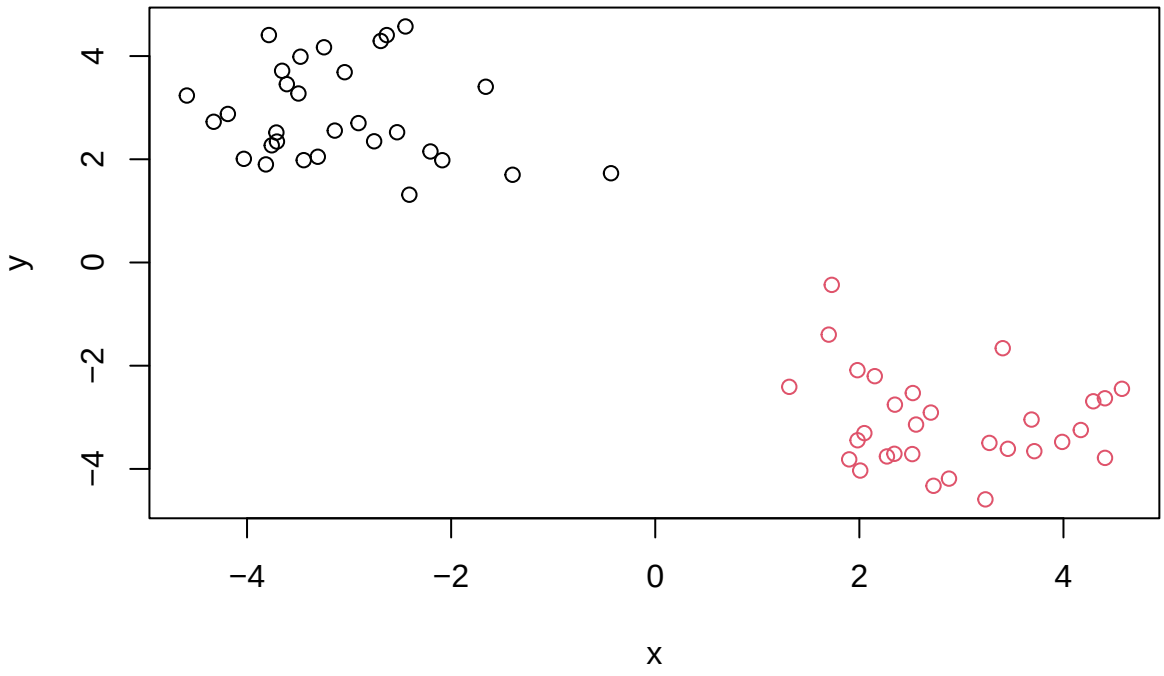
```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

You can also call 'cutree()' setting k=the number of grps/clusters you want,

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

Make our results plot

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



Principal

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

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

```
nrow(x)
```

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

```
ncol(x)
```

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

```
dim(x)
```

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

```
View(x)
```

Note how the minus indexing works

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

```
dim(x)
```

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

```
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
```

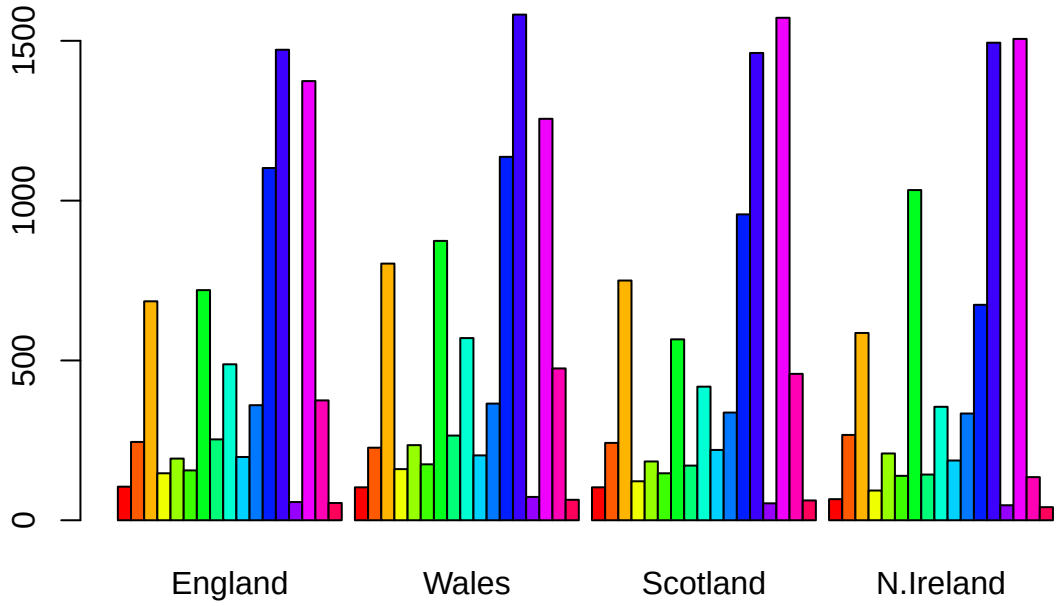
```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names = 1)
head(x)
```

```
##                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
```

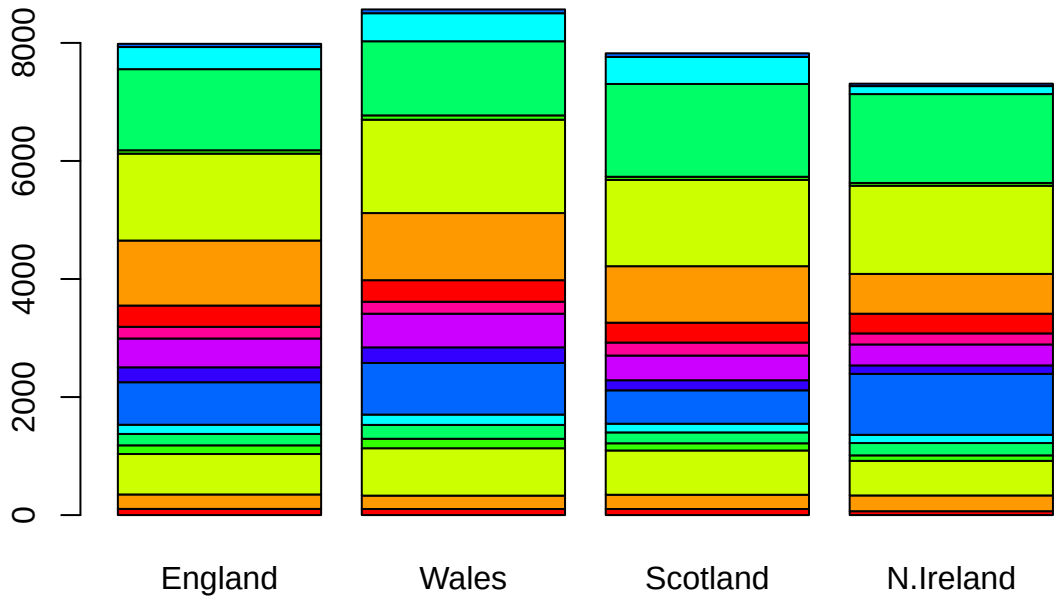
Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

The second approach is better because it doesn't delete rows.

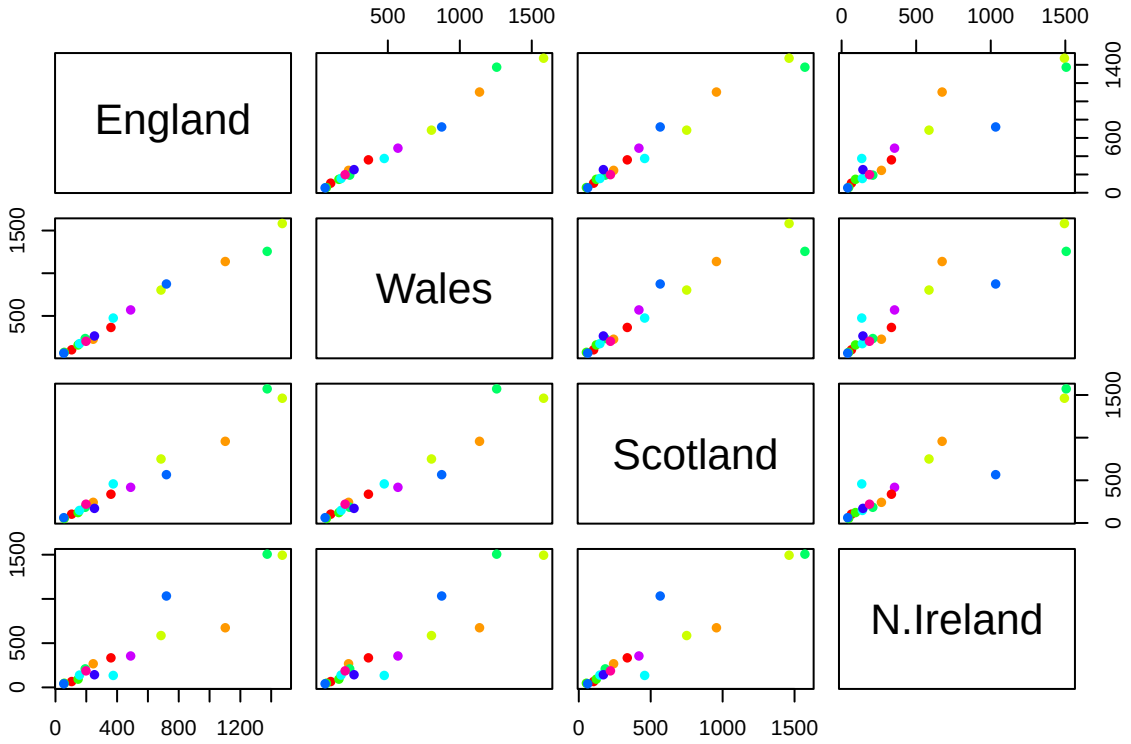
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```




```
barplot(as.matrix(x), col=rainbow(10))
```




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



> 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?

The countries are being plotted against each other. If the point are not on the diagonal it means that the values are dissimilar in terms of drinking.

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

The main difference is that it has the most different data compared to the UK.

The main function in base R is prcomp() This want's the transpose

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

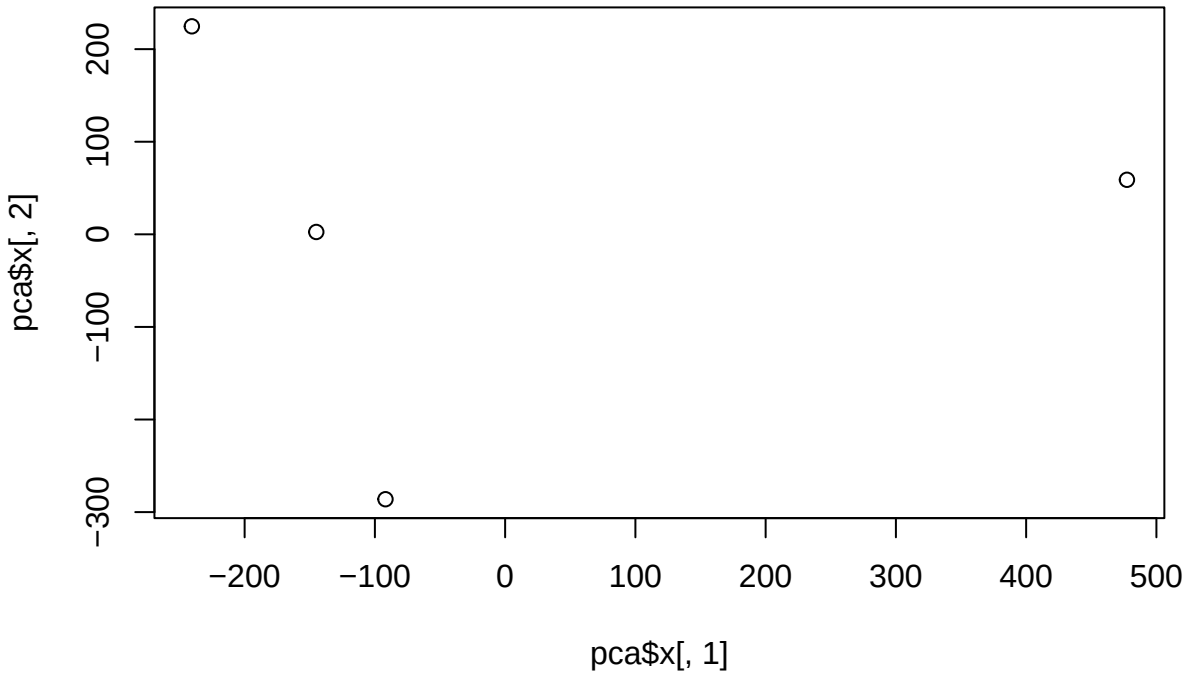
```
## Importance of components:
##                             PC1      PC2      PC3       PC4
## Standard deviation     324.1502 212.7478 73.87622 4.189e-14
## Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
## Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

```
attributes(pca)
```

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

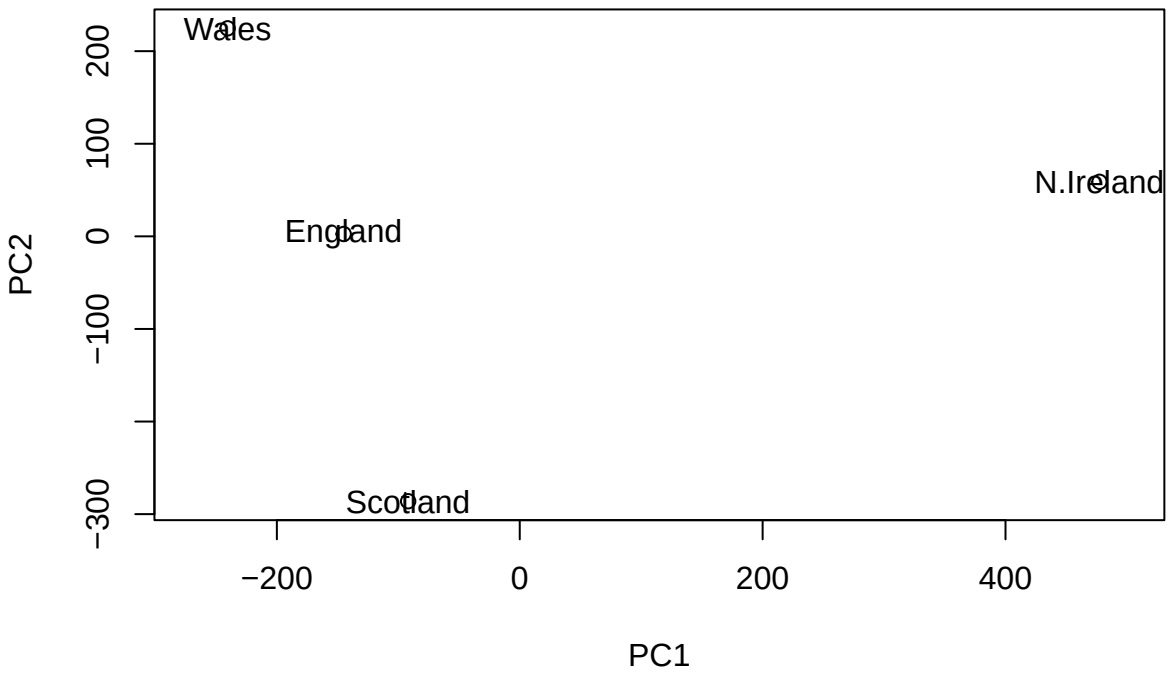
```
##
## $class
## [1] "prcomp"
```

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

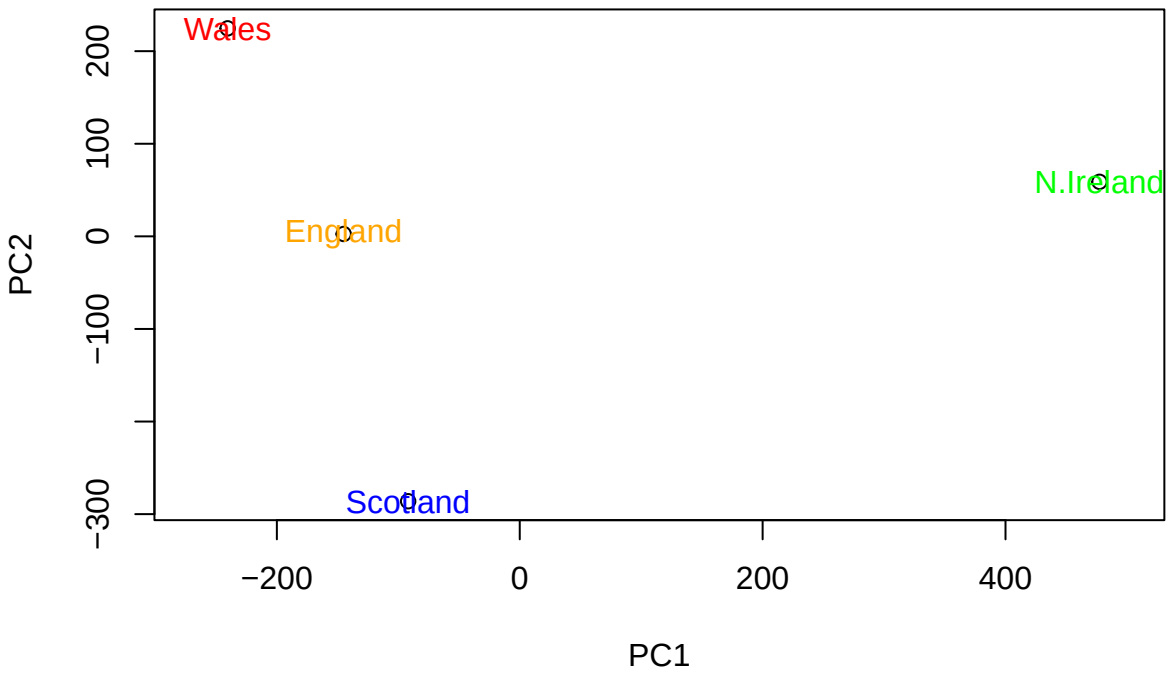


Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points

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




```
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= c("orange", "red", "blue", "green"))
```



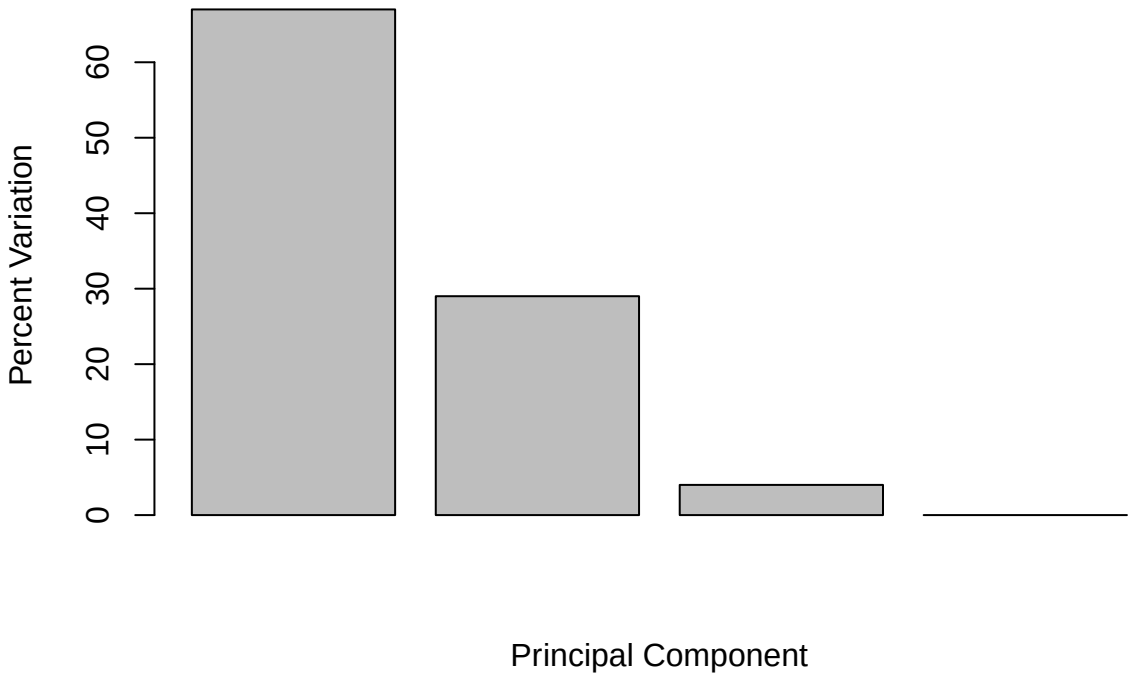

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

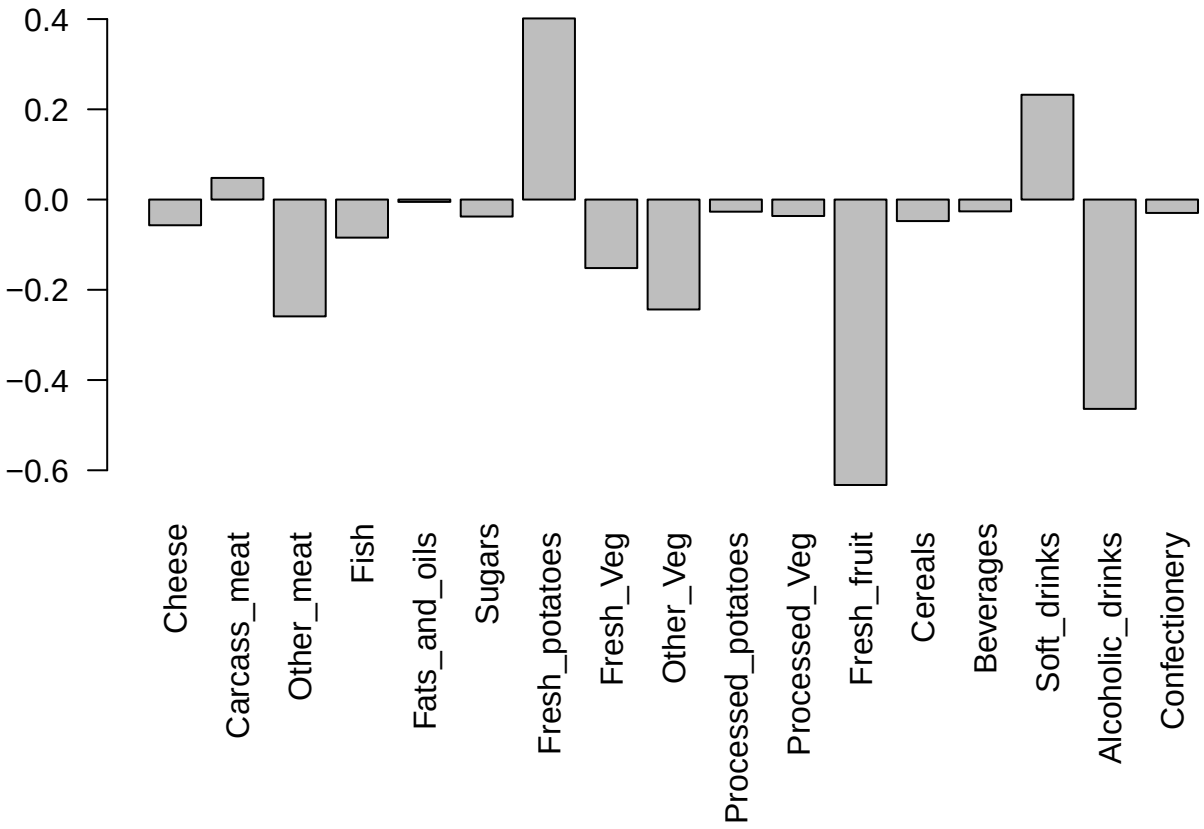
```
z <- summary(pca)
z$importance
```

```
##                             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
```

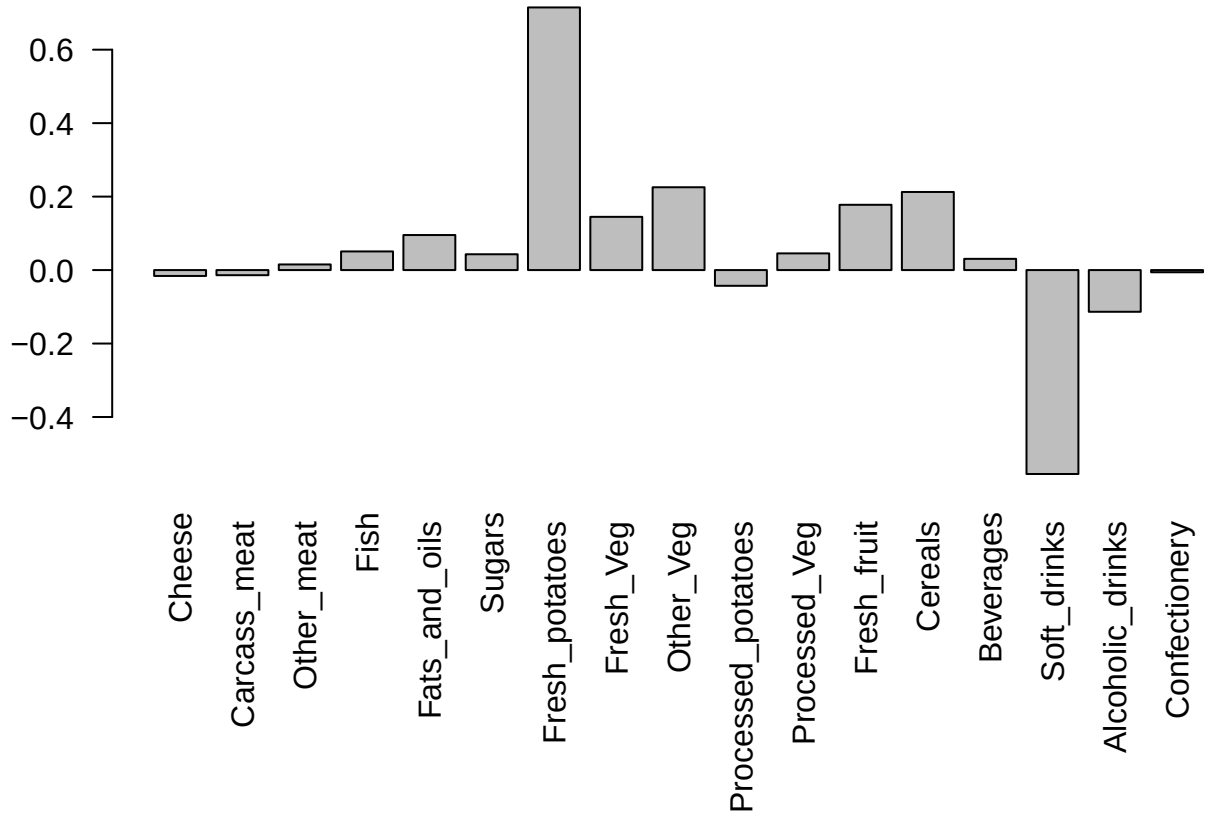
```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```




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

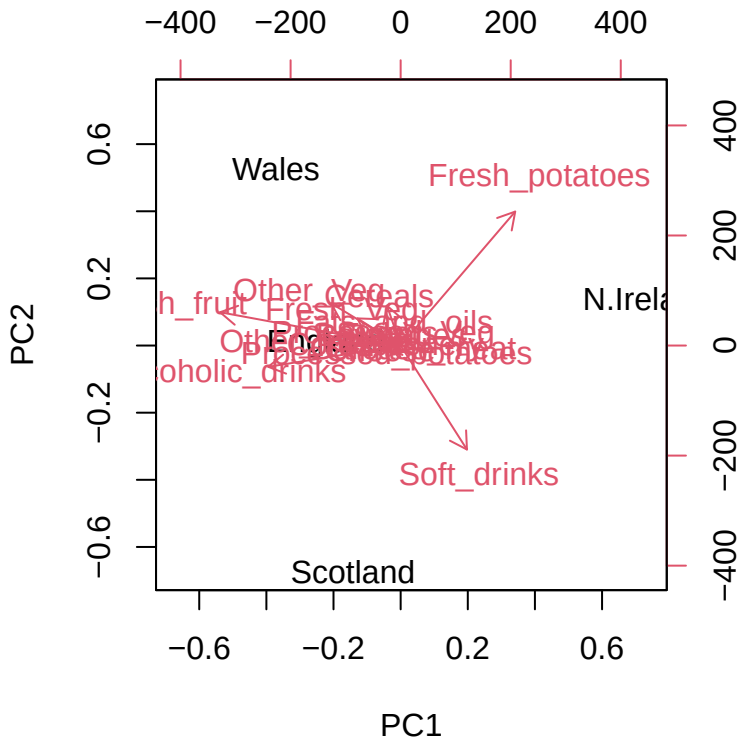



```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```



The two groups that are featured prominantely are fresh potatoes and soft drinks. PC2 tells us about the second axis and its variability.

```
biplot(pca)
```




```
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
```

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

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

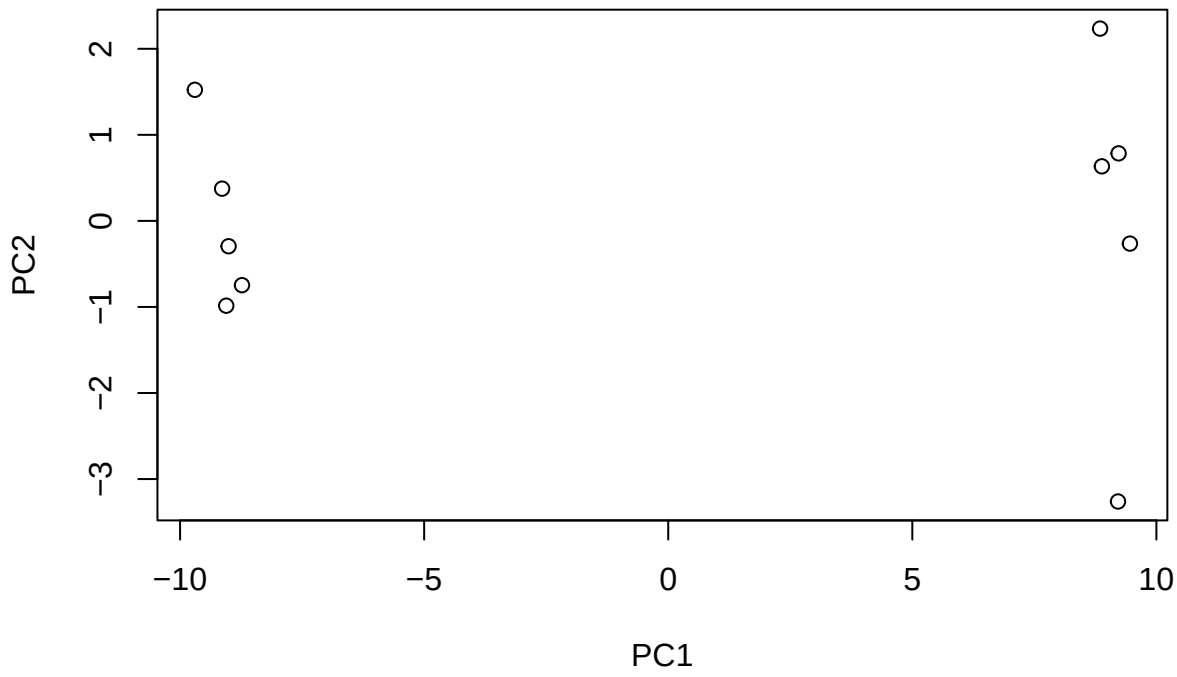
```
ncol(rna.data)
```

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

10 genes and 100 samples

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

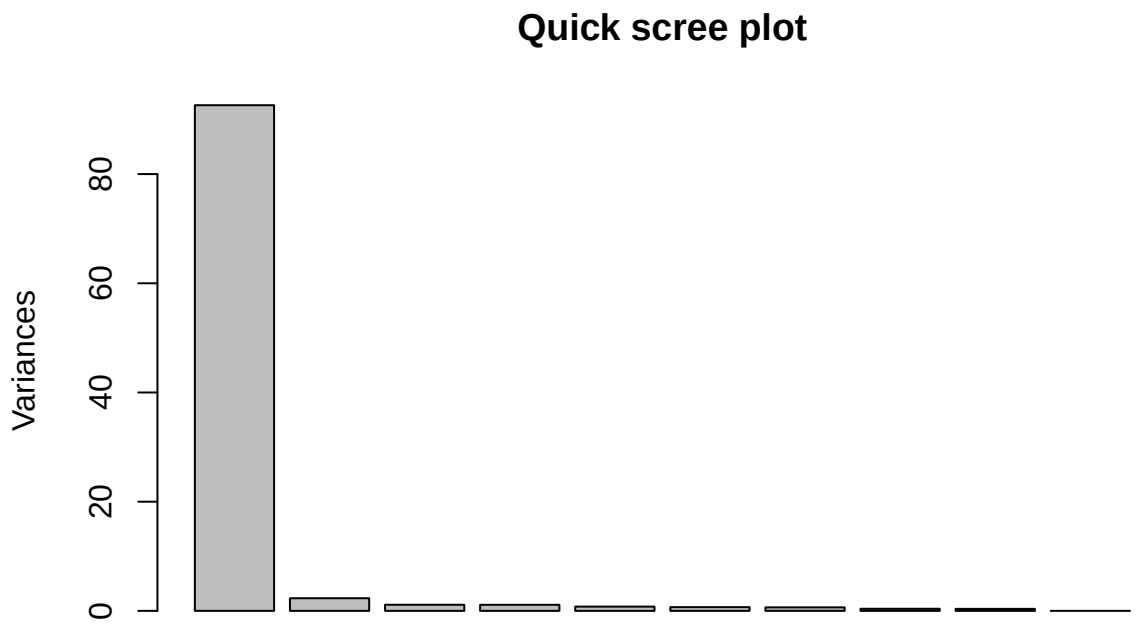
```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```




```
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
```

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

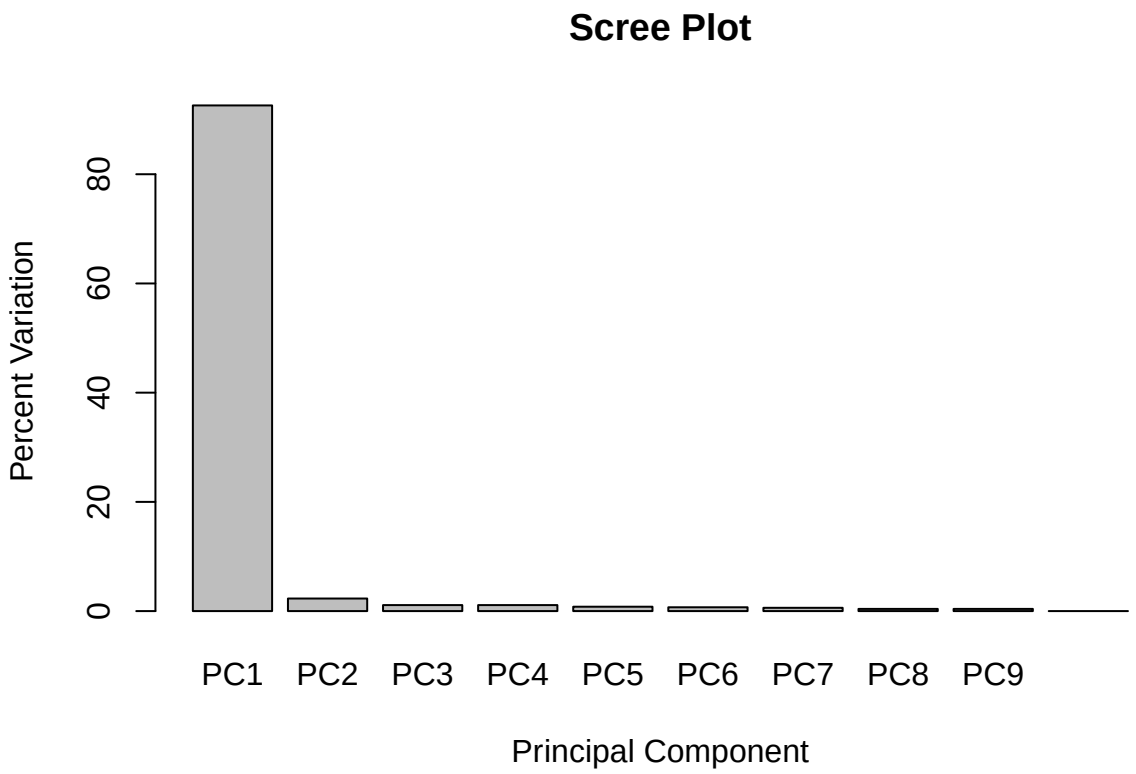



```
pca.var <- pca$sdev^2
```

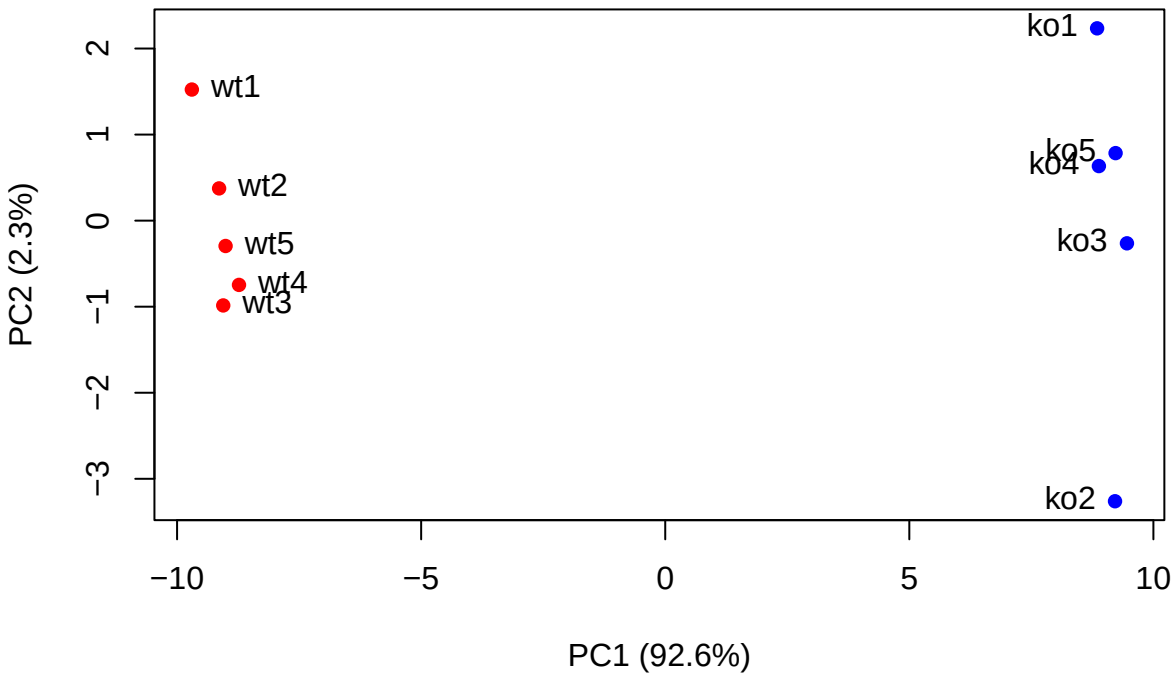
```
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)
pca.var.per
```

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

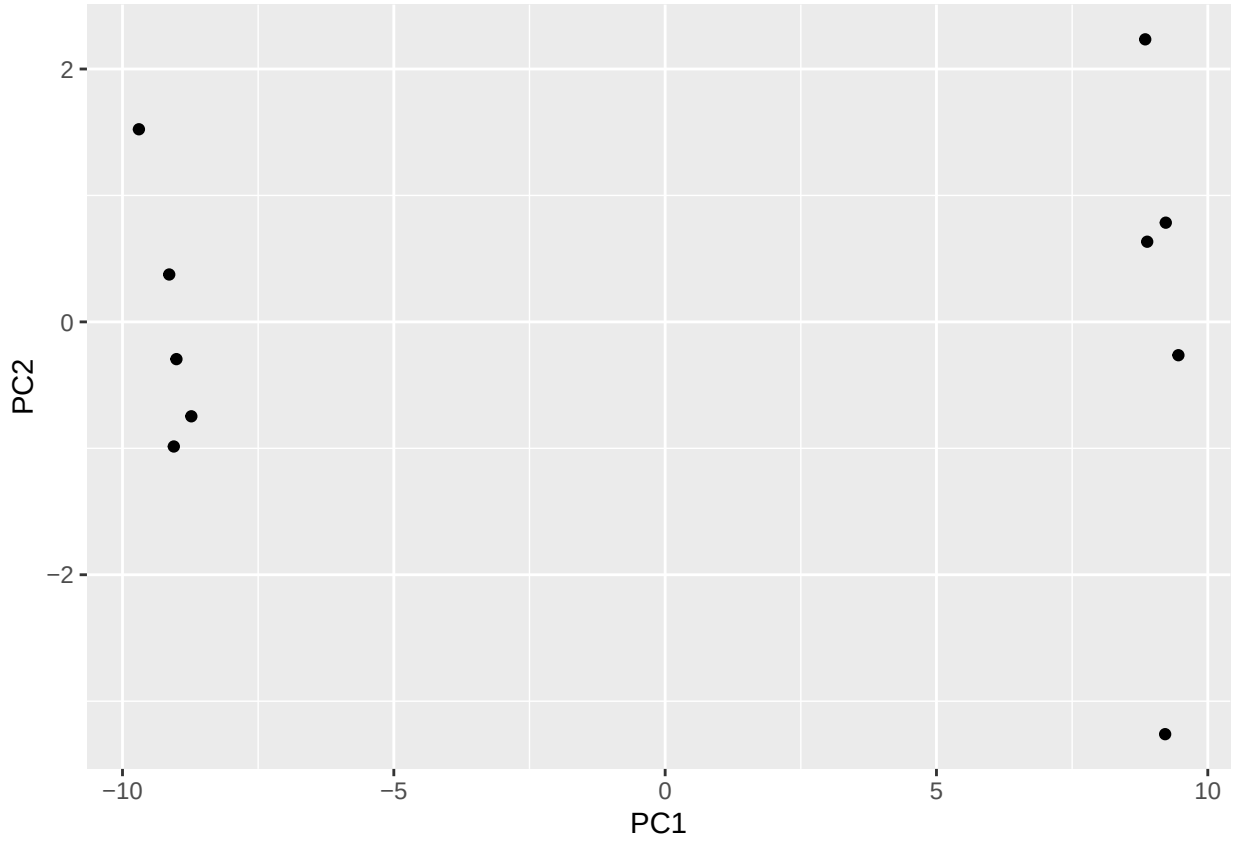
```
barplot(pca.var.per, main="Scree Plot",
        names.arg = paste0("PC", 1:10),
        xlab="Principal Component", ylab="Percent Variation")
```




```
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)))
```

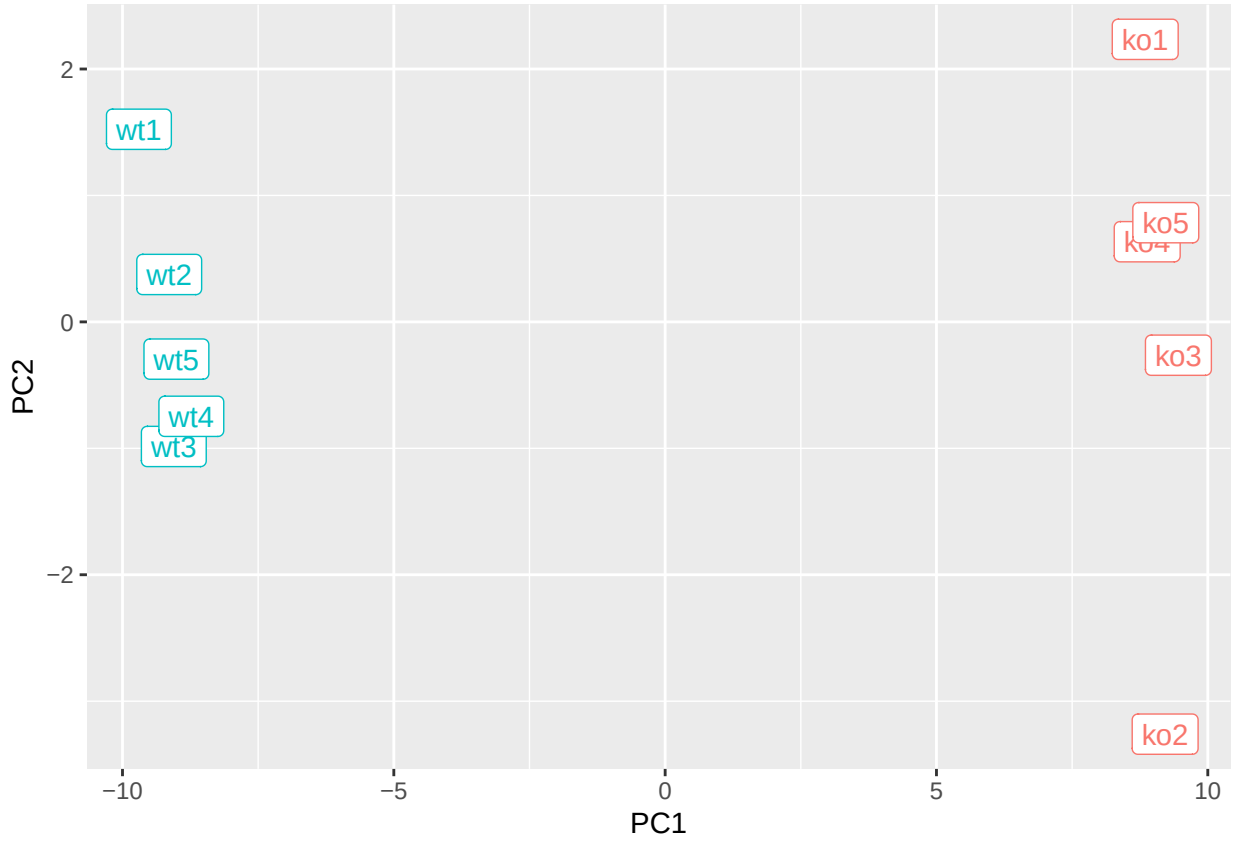



```
library(ggplot2)

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

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

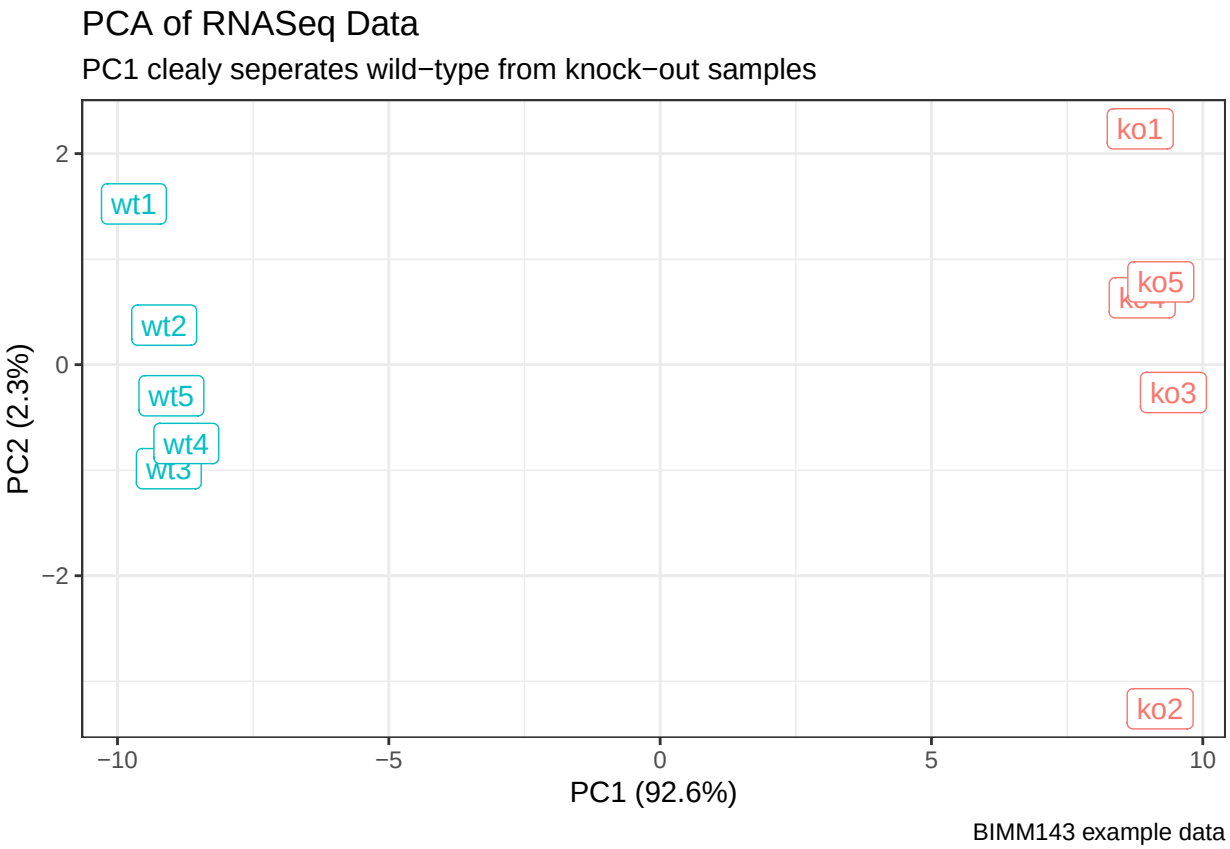


Add a ‘wt’ and ‘ko’ “condition” column

```
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
```




```
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