Machine Learning 1

Katherine Wong (A16162648)

10/21/2021

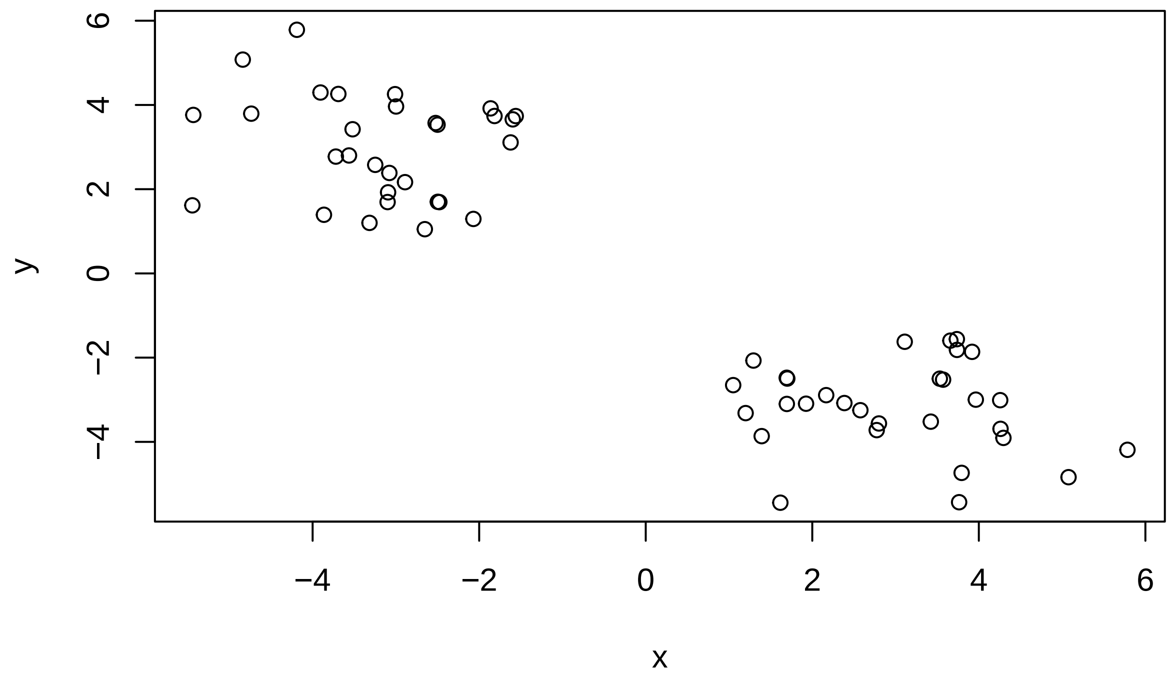
First up is clustering methods

Kmeans clustering

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

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

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



Q. Can we use kmeans() to cluster this data setting k 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  3.004976 -3.160746
## 2 -3.160746  3.004976
##
## Clustering vector:
##  [1] 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 1 1 1 1 1 1 1 1
## [39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1] 77.17282 77.17282
##  (between_SS / total_SS =  88.1 %)
##
## 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] 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 1 1 1 1 1 1 1 1
## [39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

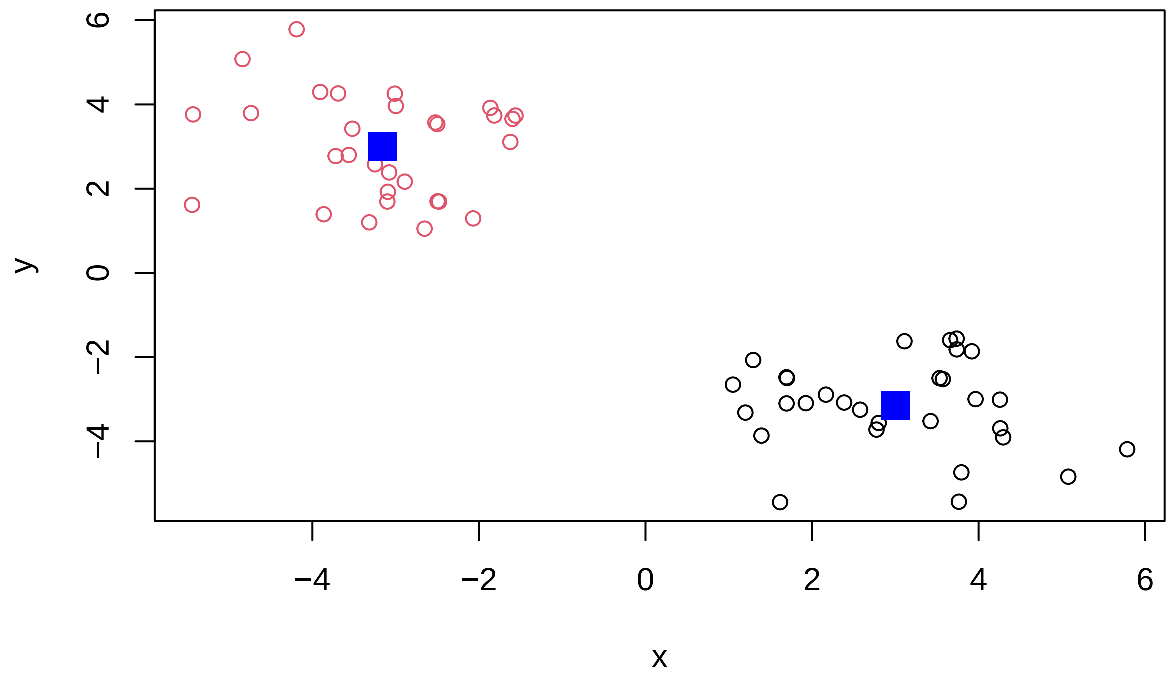
Q. What 'component of your result object details cluster center?

```
km$centers
```

```
##           x         y
## 1  3.004976 -3.160746
## 2 -3.160746  3.004976
```

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

Analyze this same data with hclust().

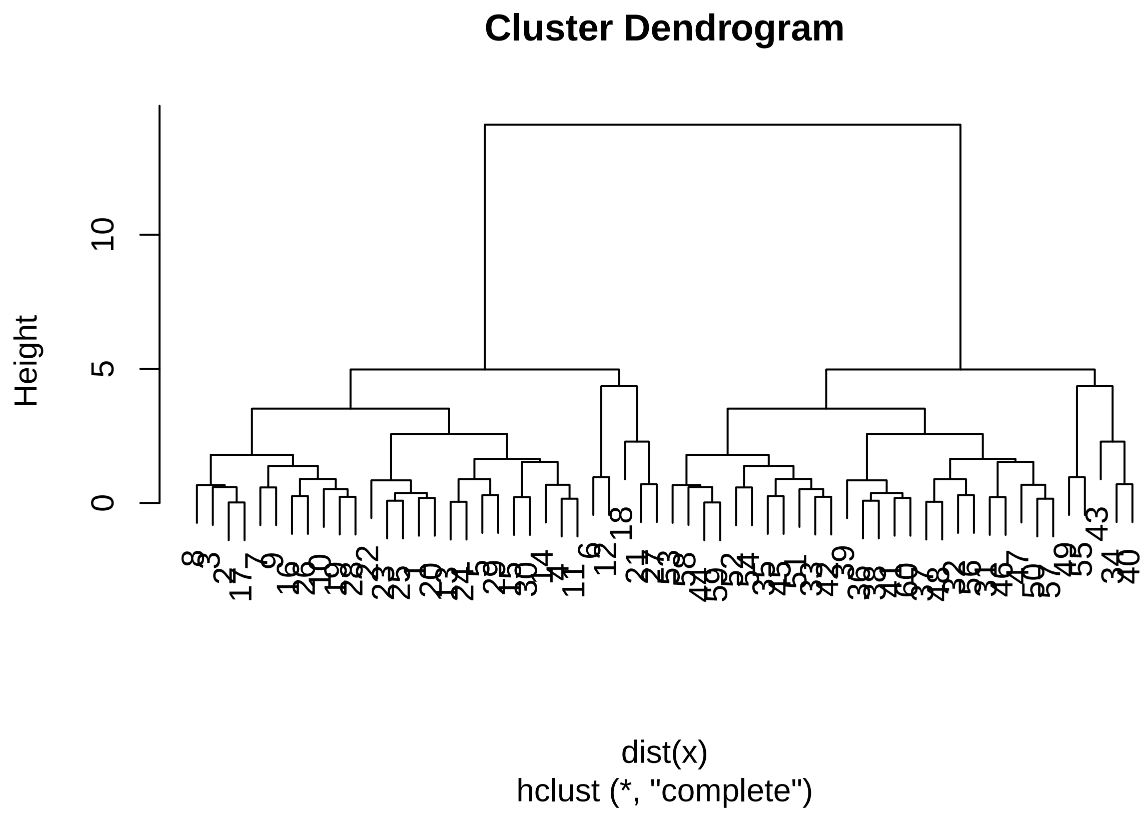
Demonstrate the use of dist(), hclust(), plot() and cutree() functions to do clustering, Generate dendrograms and return cluster assignment membership vector…

```
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.

```r
plot(hc)
```



To get our cluster membership vector we have t do a wee bit more work. We have to "cut" the tree where we think it makes sense. For this we use the 'cutree()' function.

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

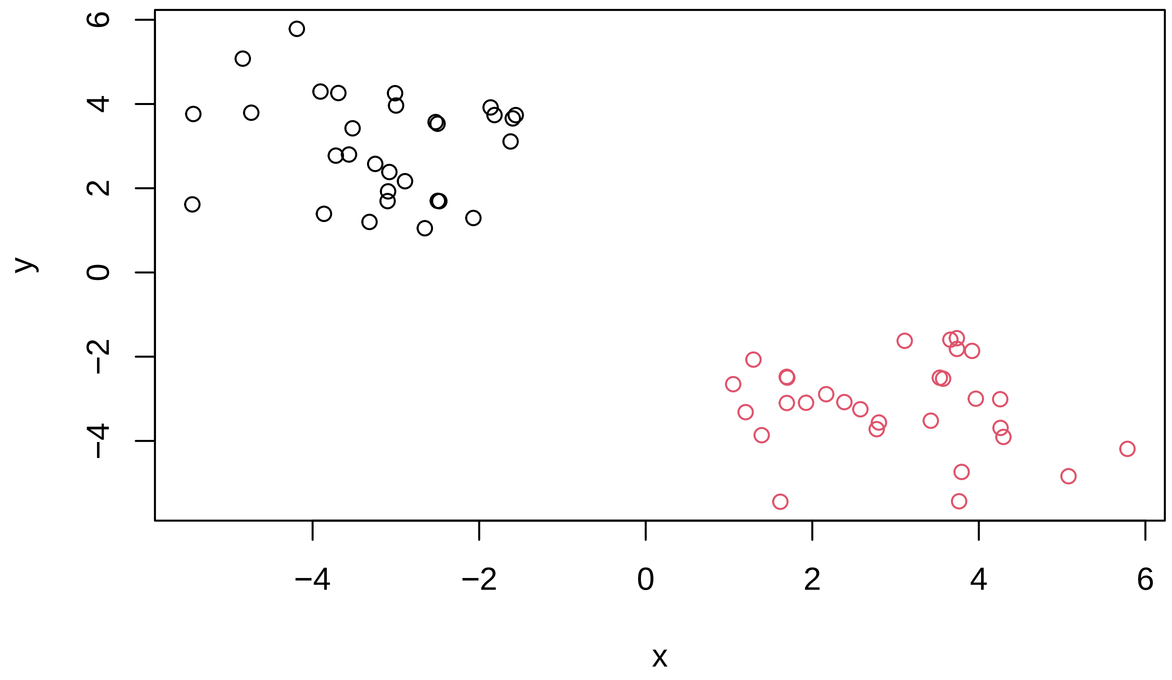
```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

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

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

Make our results plot

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



Princial Component Analysis (PCA)

Read data on food stuffs from

```
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?

```
dim(x)
```

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

There are 17 rows and 5 columns. You can use the dim() function to get the rows and columns or call nrow() to get rows and ncol() to get columns separately.

Checking data

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
```

There is an extra col as reported by the dim() function.

```
# Note how the minus indexing works
#rownames(x) <- x[,1]
x <- x[,-1]
head(x)
```

```
##   England Wales Scotland N.Ireland
## 1     105   103      103        66
## 2     245   227      242       267
## 3     685   803      750       586
## 4     147   160      122        93
## 5     193   235      184       209
## 6     156   175      147       139
```

```
# Or do this when loading data
# x <- read.csv(url, row.names=1)
# head(x)
```

Check dimensions again

```
dim(x)
```

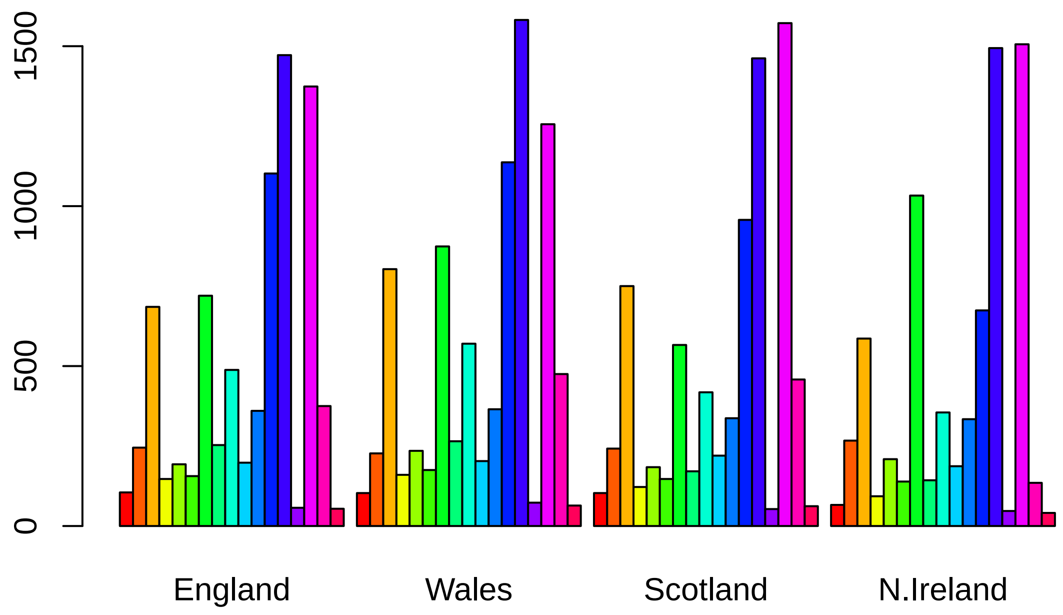
```
## [1] 17  4
```

> 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?

Loading it and setting row.names to the first column is the preferred method. For the first method, if you call x <- x[,-1] multiple times, the columns start disappearing.

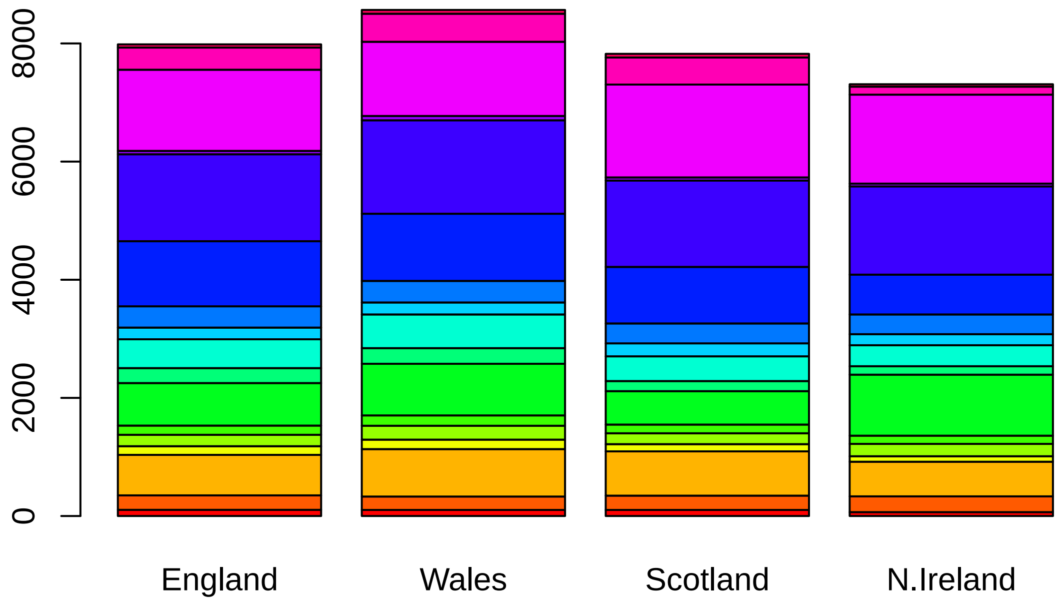
Spotting major differences and trends

```
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), beside=F, col=rainbow(nrow(x)))
```



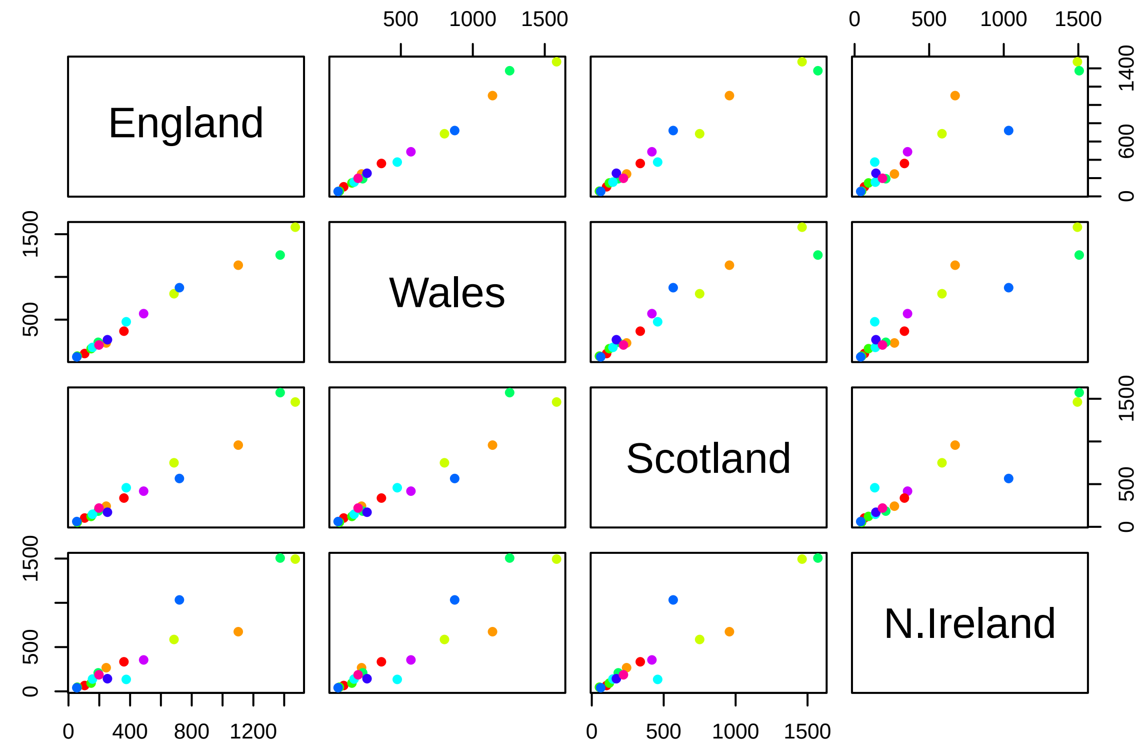
A: If you change beside=T to beside=F, you can achieve this barplot.

The stacked barplot is not helpful.

> 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?

Each row is the y axis (for example, first row is england) and column is the x axis (for example, first column is england). If it lies on the diagonal then the two countries have similar food consumption for all food types. If it deviates from the diagonal, then the countries don't have similar food consumption numbers.

```
pairs(x, 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?

N.Ireland has blue dots higher on the y axis than any other country. The orange dot also seems to be lower on the y axis compared to the other UK countries.

PCA to the rescue!

The main function in base R for PCA is 'prcomp()' This wants the transpose of our data

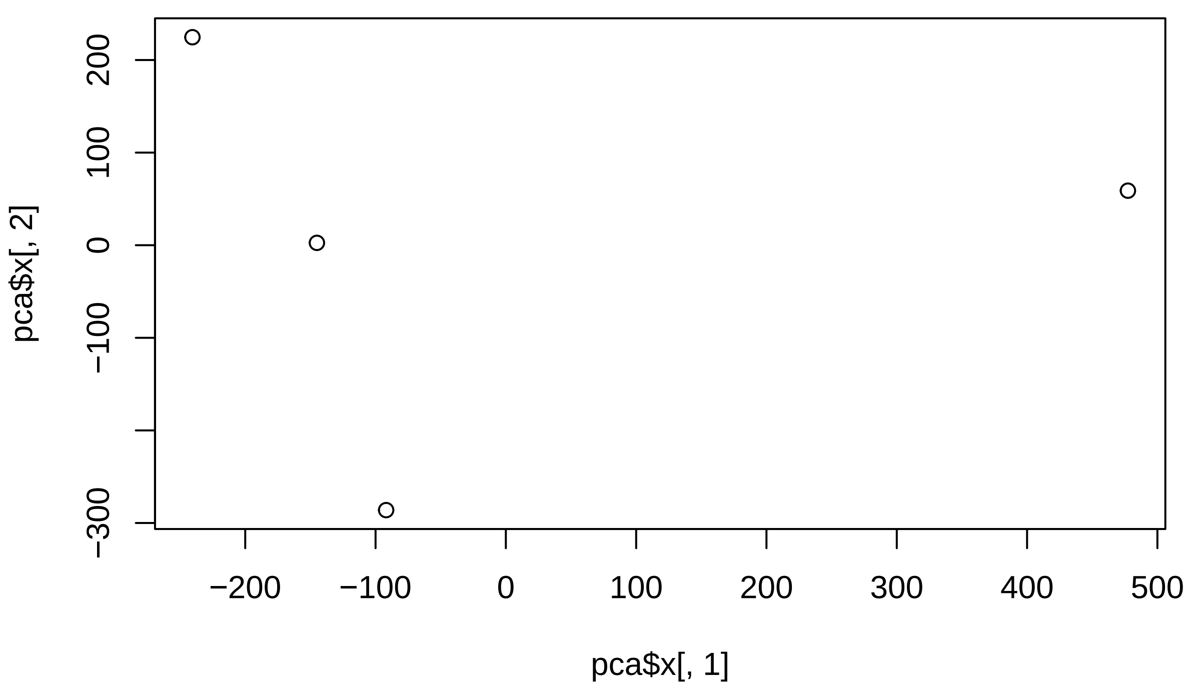
```r
pca <- prcomp(t(x))
summary(pca)
```

```
## 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
```

```r
attributes(pca)
```

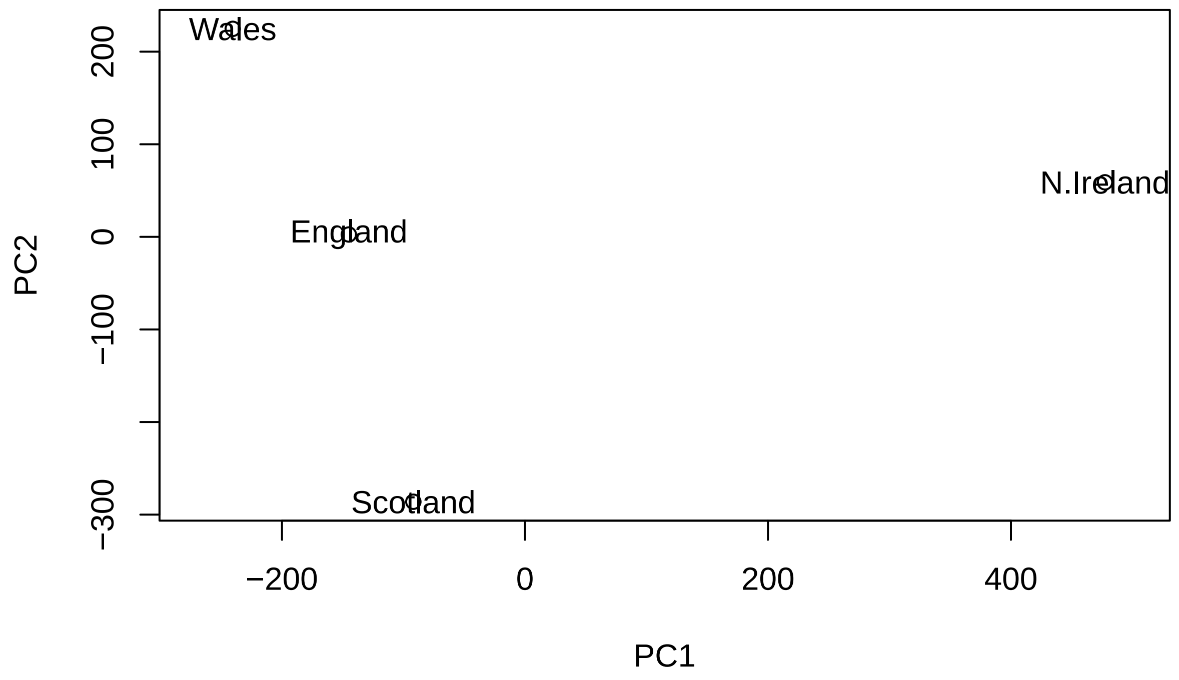
```
## $names
## [1] "sdev"     "rotation" "center"   "scale"    "x"
##
## $class
## [1] "prcomp"
```

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



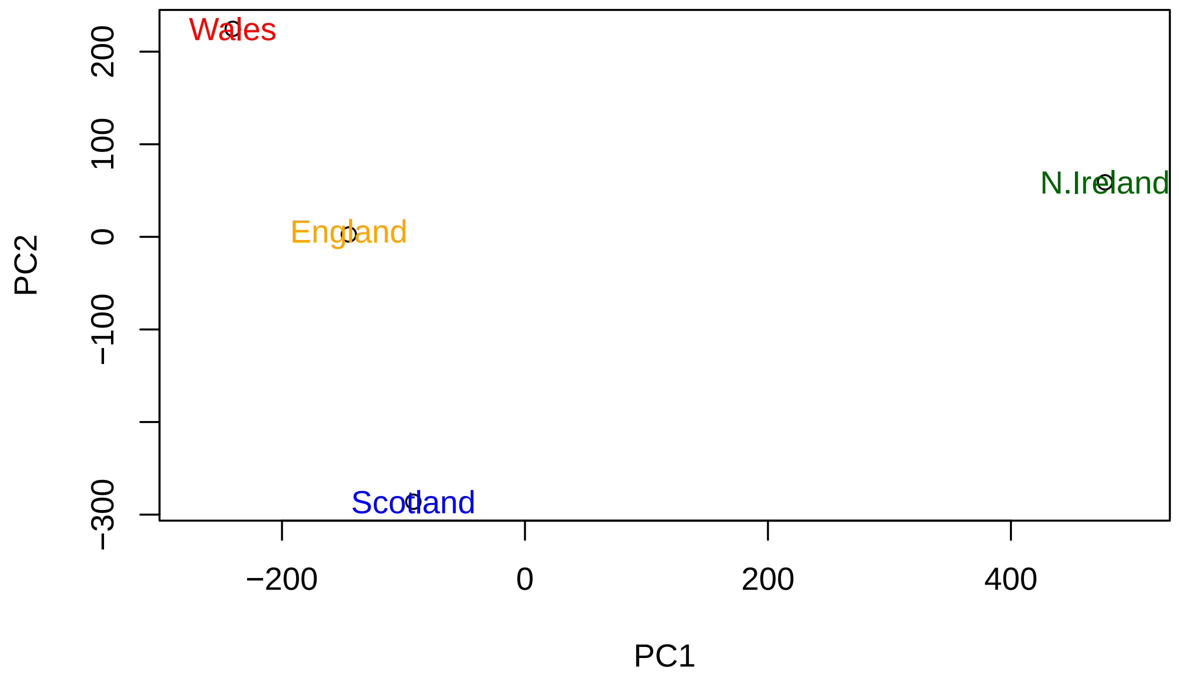
> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

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



> Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at the 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(x), col = c("orange", "red", "blue", "dark green"))
```



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

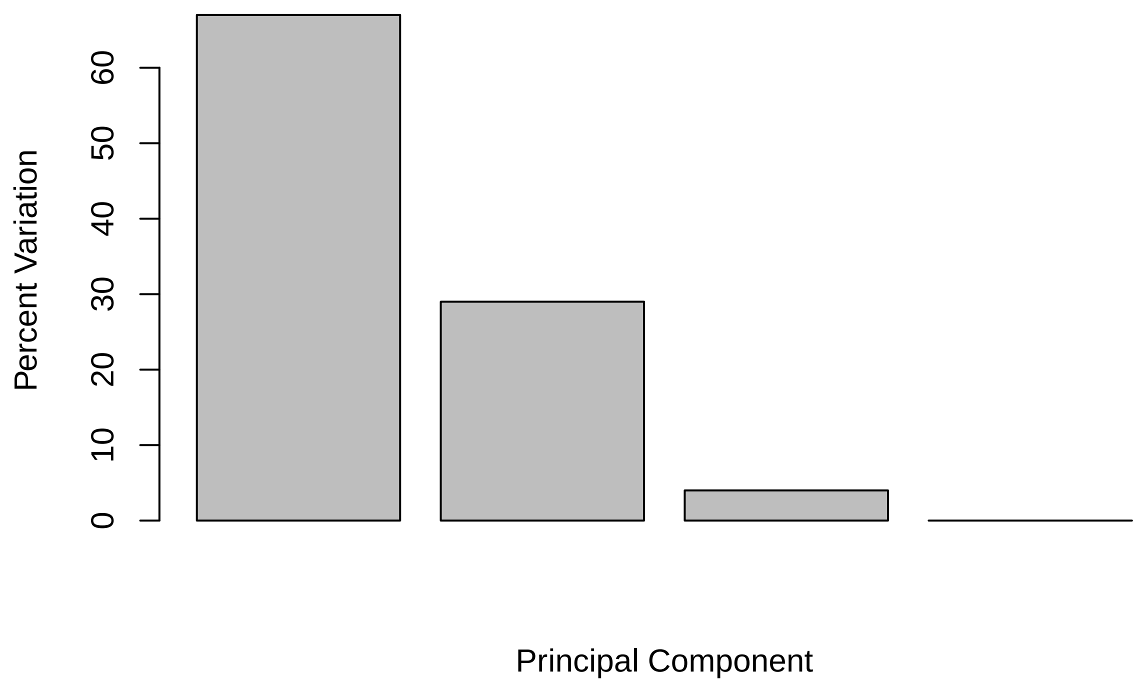
```
## [1] 67 29  4  0
```

or the second row here...

```
## or the second row here...
z <- summary(pca)
z$importance
```

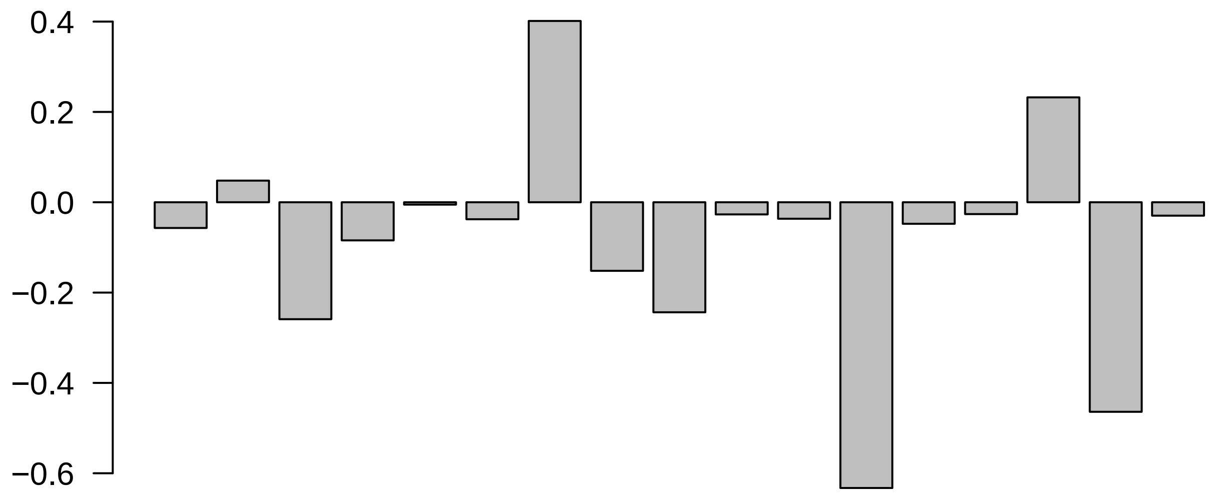
```
##                              PC1       PC2      PC3          PC4
## Standard deviation     324.15019 212.74780 73.87622 4.188568e-14
## Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
## Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

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



Digging Deeper (variable loadings)

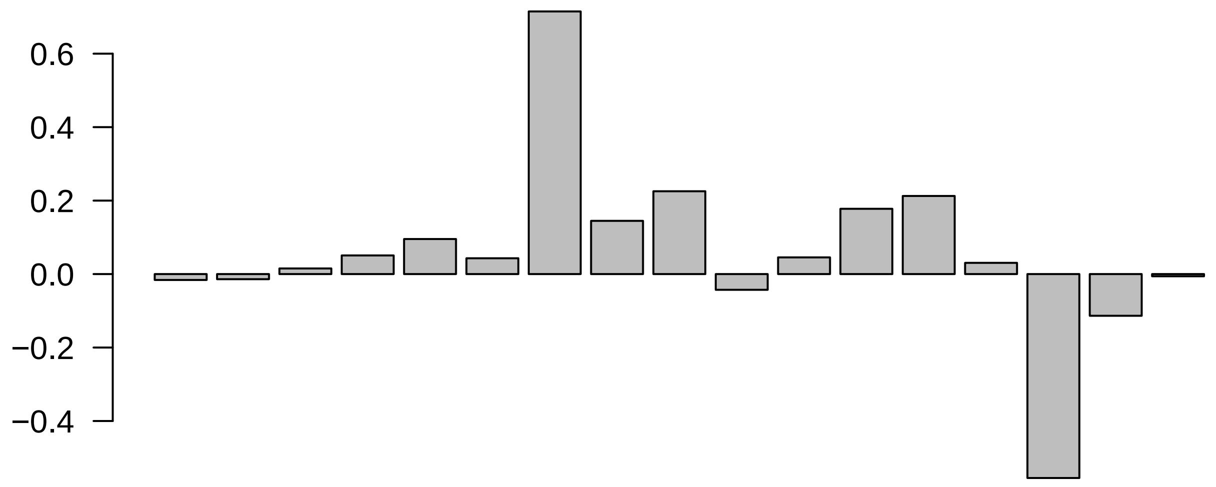
```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



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

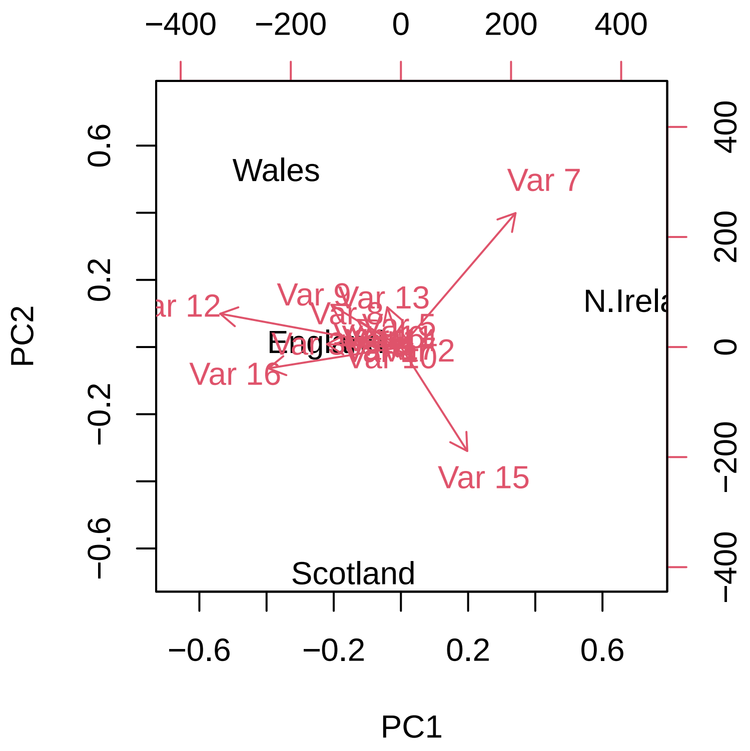
Fresh potatoes and soft drinks are positive. Negative is other meat, fresh fruit and alcoholic drinks

```
## Lets focus on PC2
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```



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?

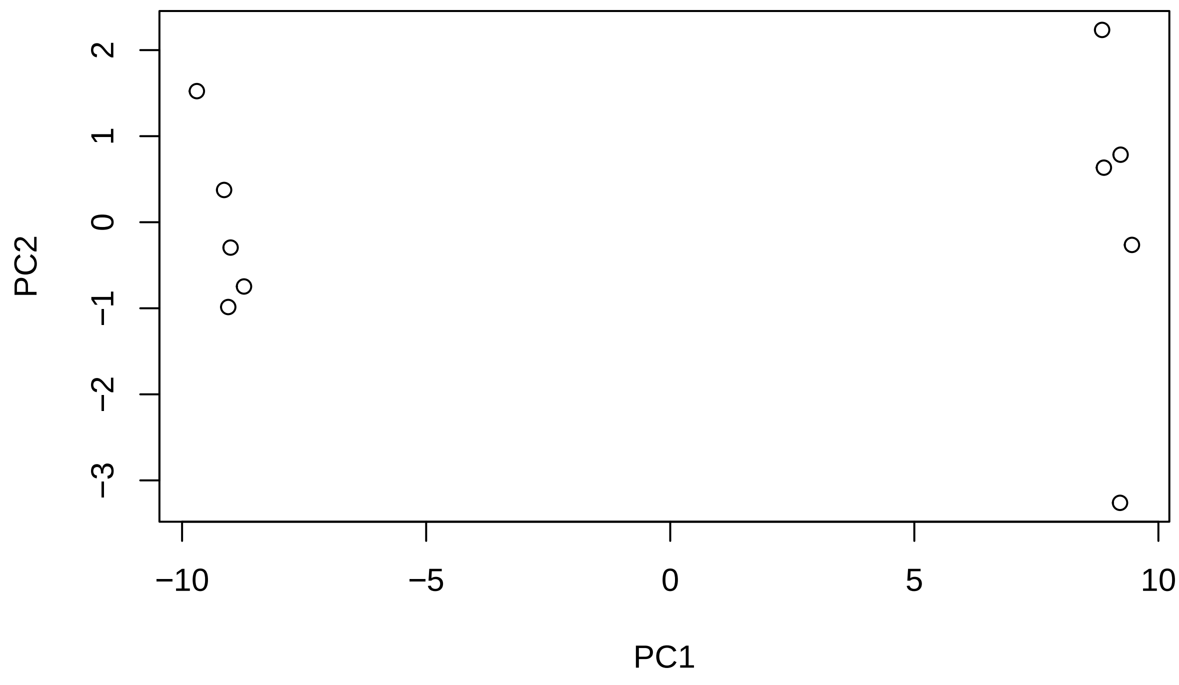
```
dim(rna.data)
```

```
## [1] 100  10
```

100 genes, and 10 samples.

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

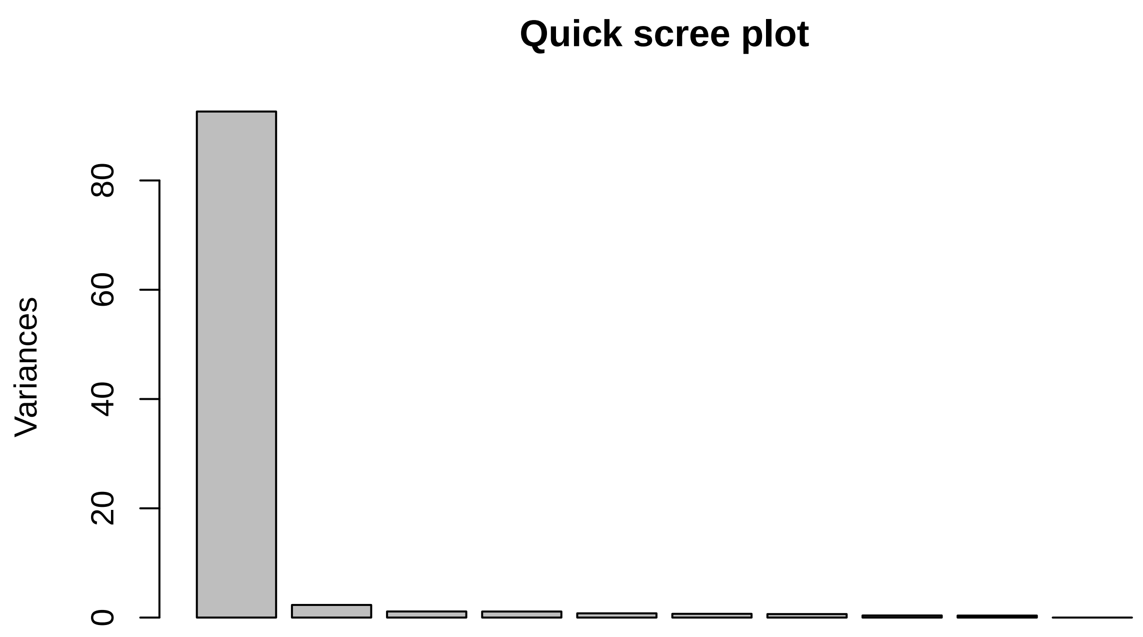
```r
## Simple un polished plot of pc1 and pc2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```



```r
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
```

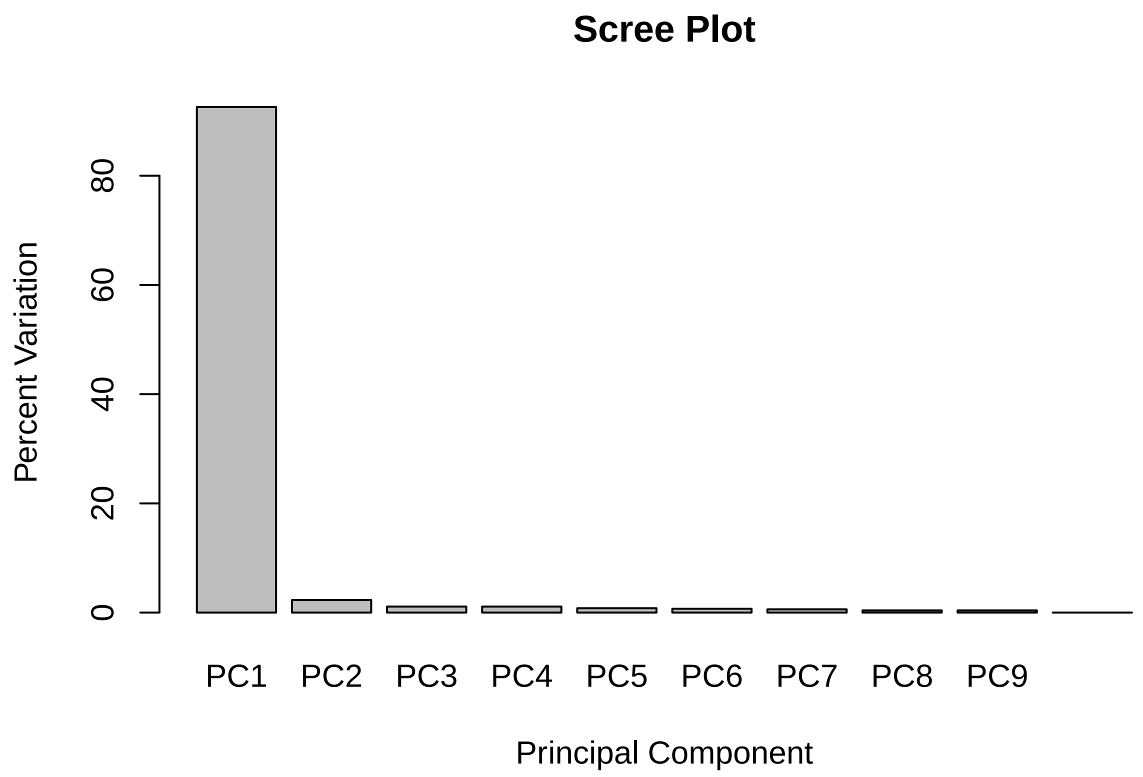
```r
plot(pca, main="Quick scree plot")
```



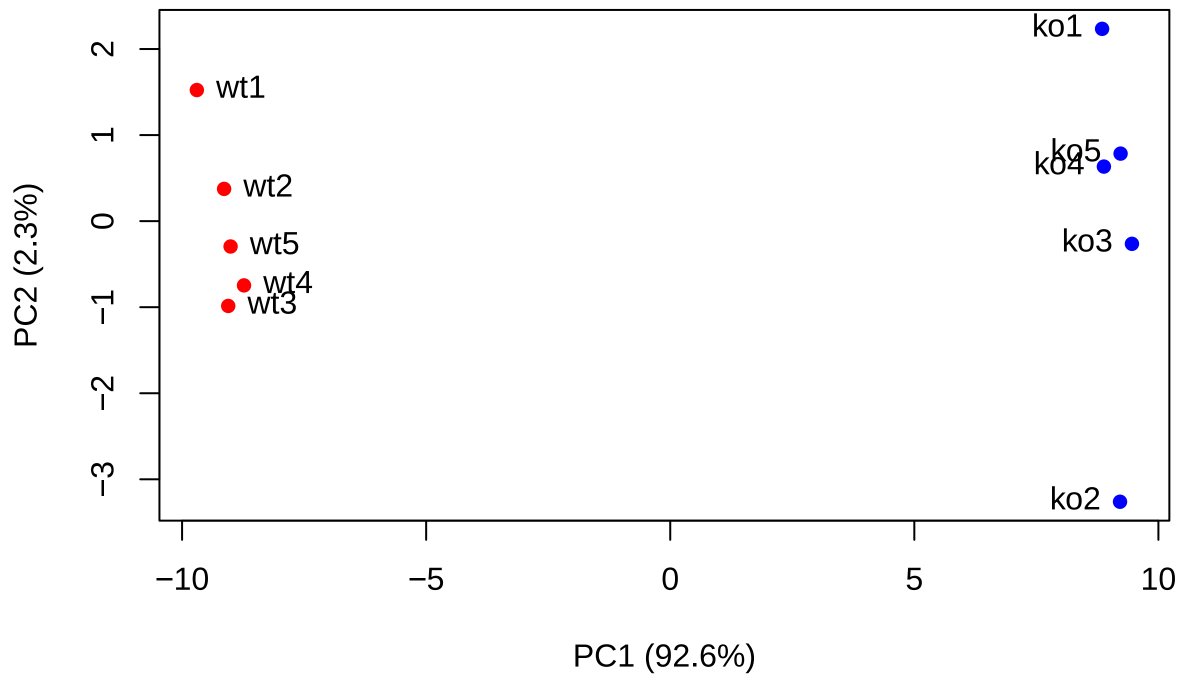
```
## Variance captured per PC
pca.var <- pca$sdev^2

## 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
```

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

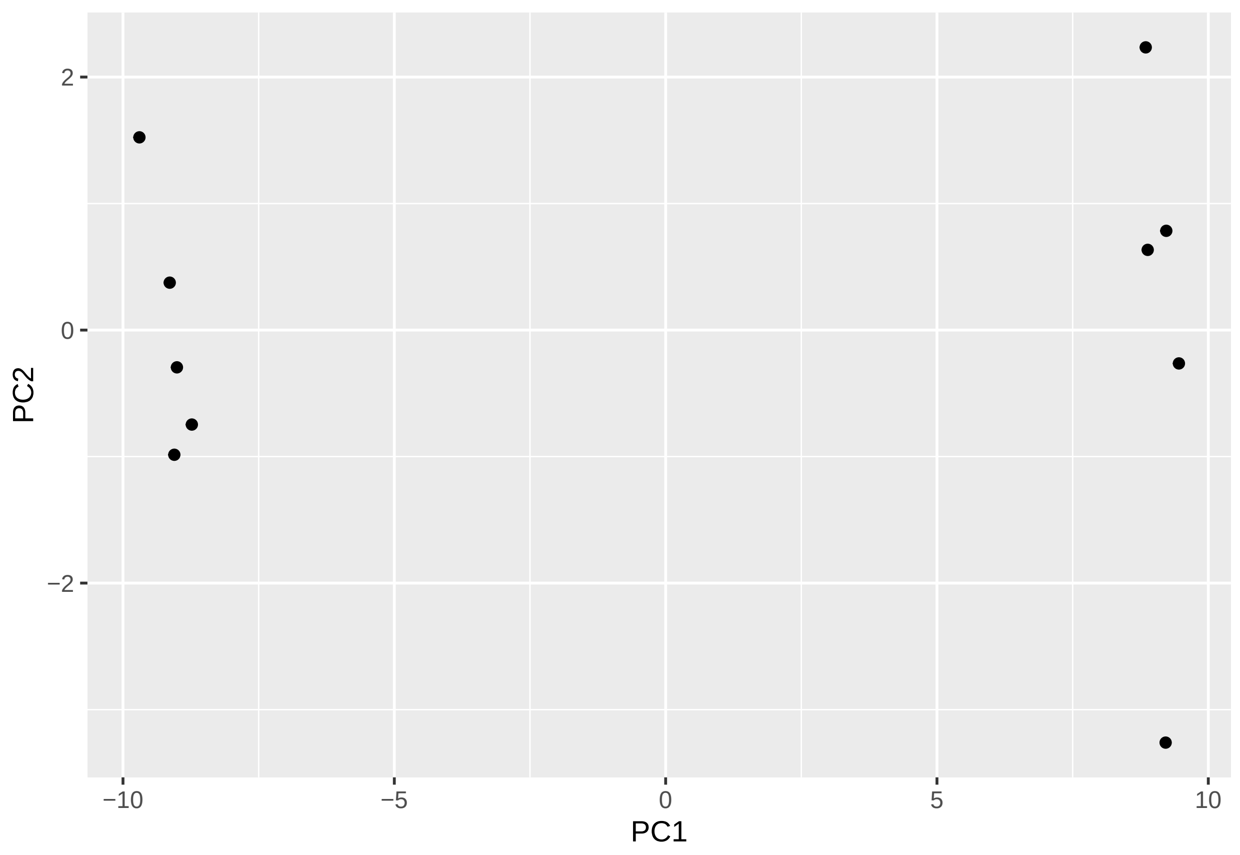


```r
## A vector of colors for wt and ko samples
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

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)))
```

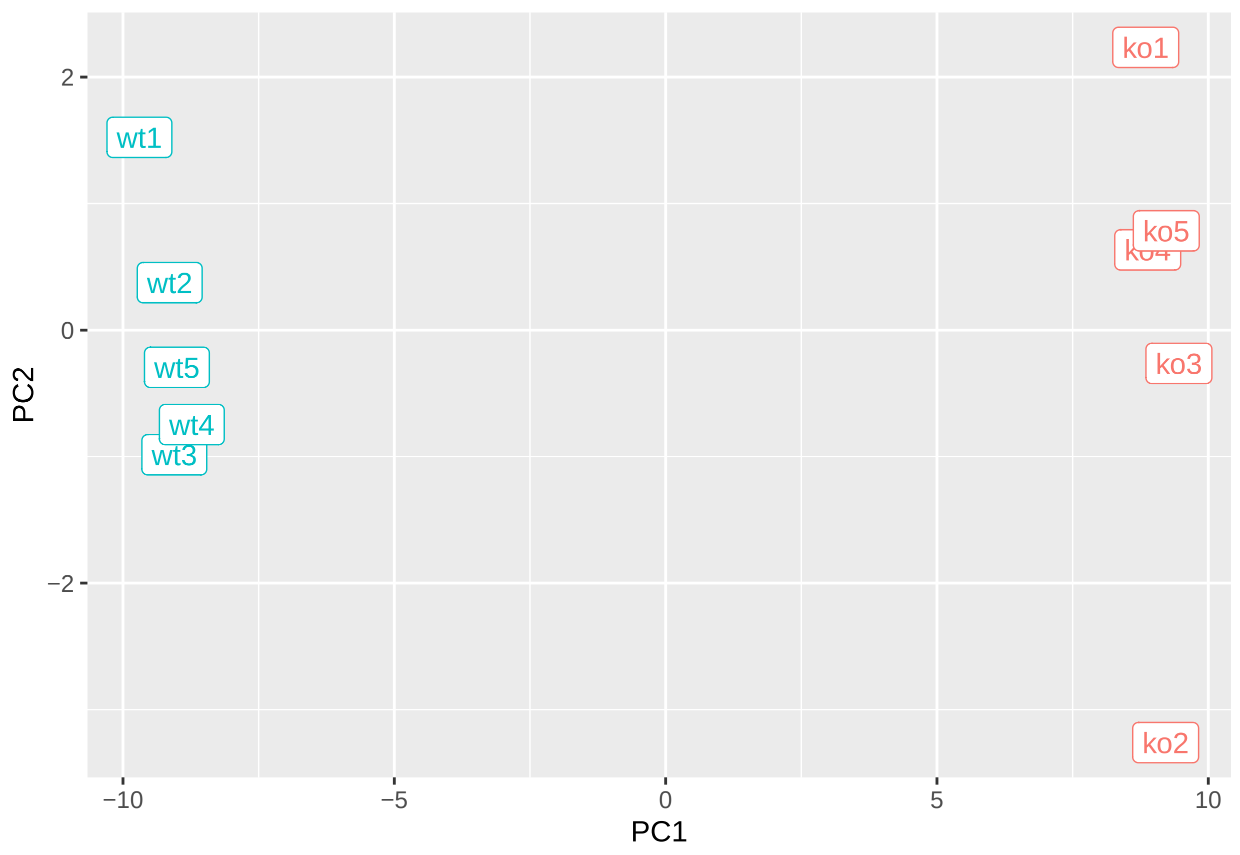


# Using ggplot

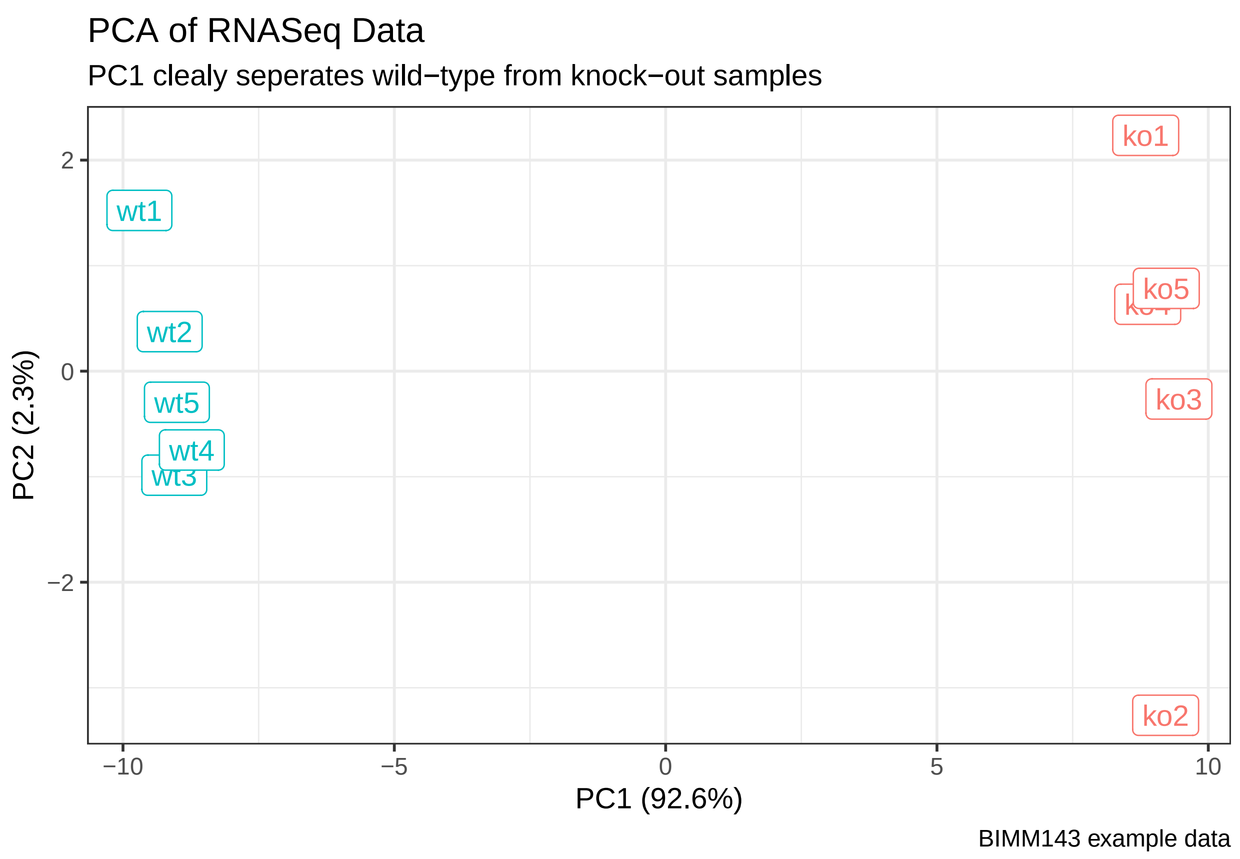
```
library(ggplot2)

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
wt1
wt2
wt5
wt4
wt3
ko1
ko5
ko4
ko3
ko2
2
0
−2
−10
−5
0
5
10
PC1 (92.6%)
PC2 (2.3%)
BIMM143 example data