Machine Learning 1

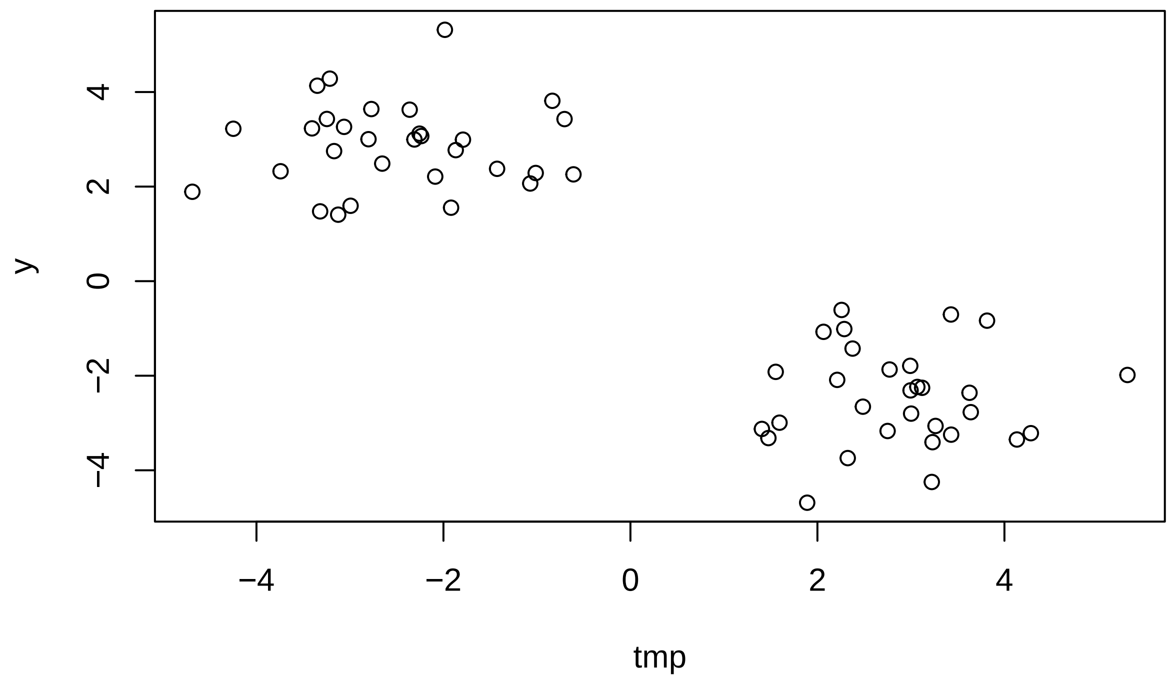
Delaney (PID: A15567985)

2/8/2022

First up kmeans()

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

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



Now we have some made up data in 'x' lets see how kmeans works with this data

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

```
## K-means clustering with 2 clusters of sizes 30, 30
##
## Cluster means:
##         tmp         y
## 1 -2.475488  2.867306
## 2  2.867306 -2.475488
##
## 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 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] 54.78856 54.78856
##  (between_SS / total_SS =  88.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Q. How many points are in each cluster?

```
k$size
```

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

Q. How do we go 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 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
```

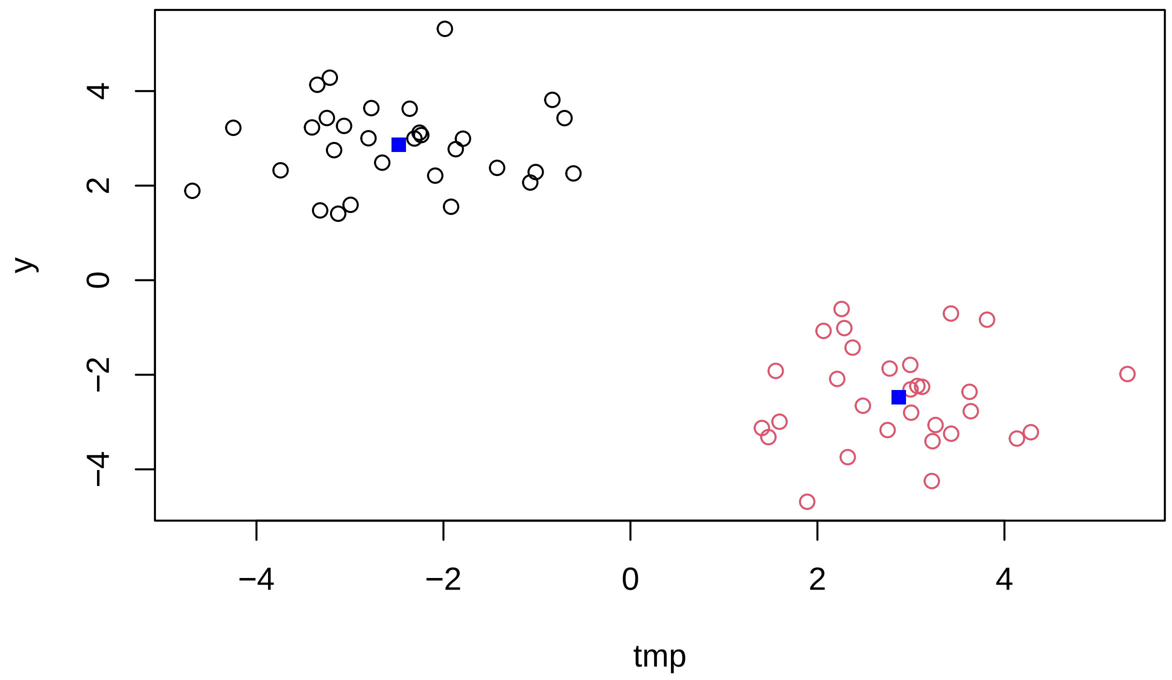
Q. What about cluster centers?

```
k$centers
```

```
##         tmp         y
## 1 -2.475488  2.867306
## 2  2.867306 -2.475488
```

Now we've got to the main results let's use them to plot our data with kmeans result.

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



Now for Hierachical Clustering

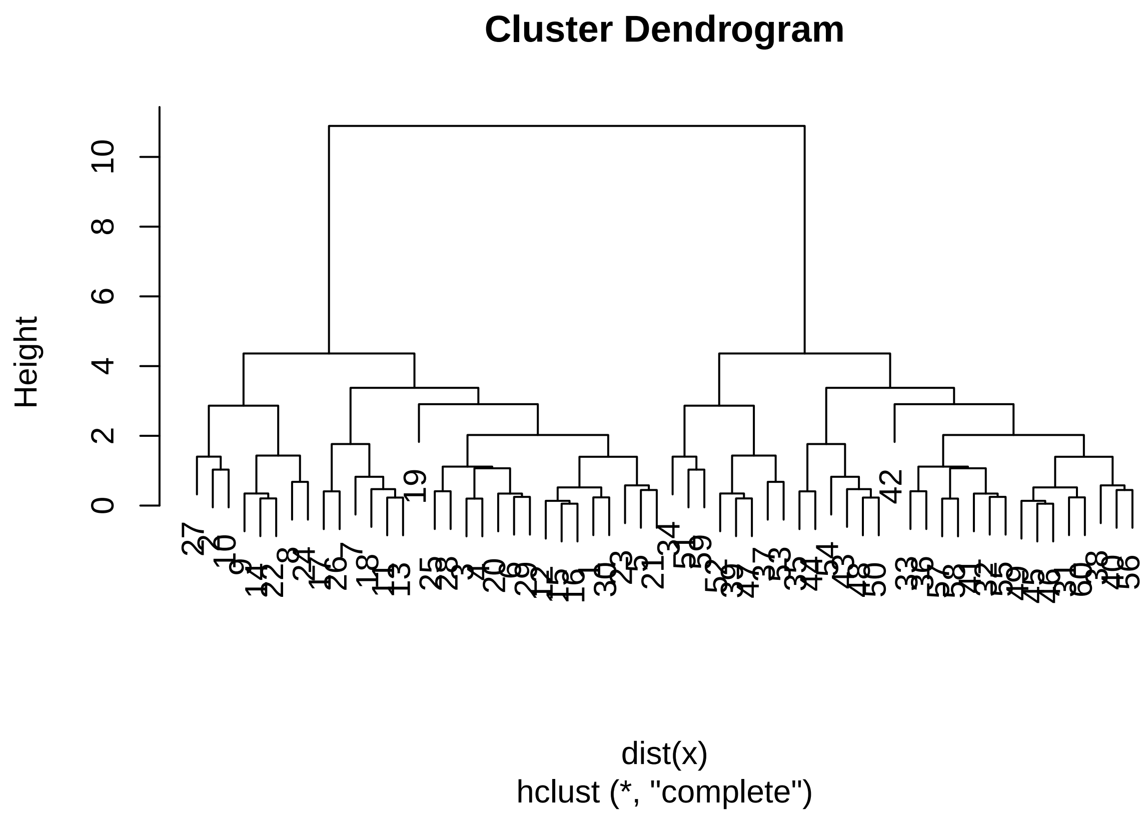
We will cluster the same data 'x' with the hclust(). In this case 'hclust()' requires a distance matrix as input.

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

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

Let's plot our hclust result

```
plot(hc)
```



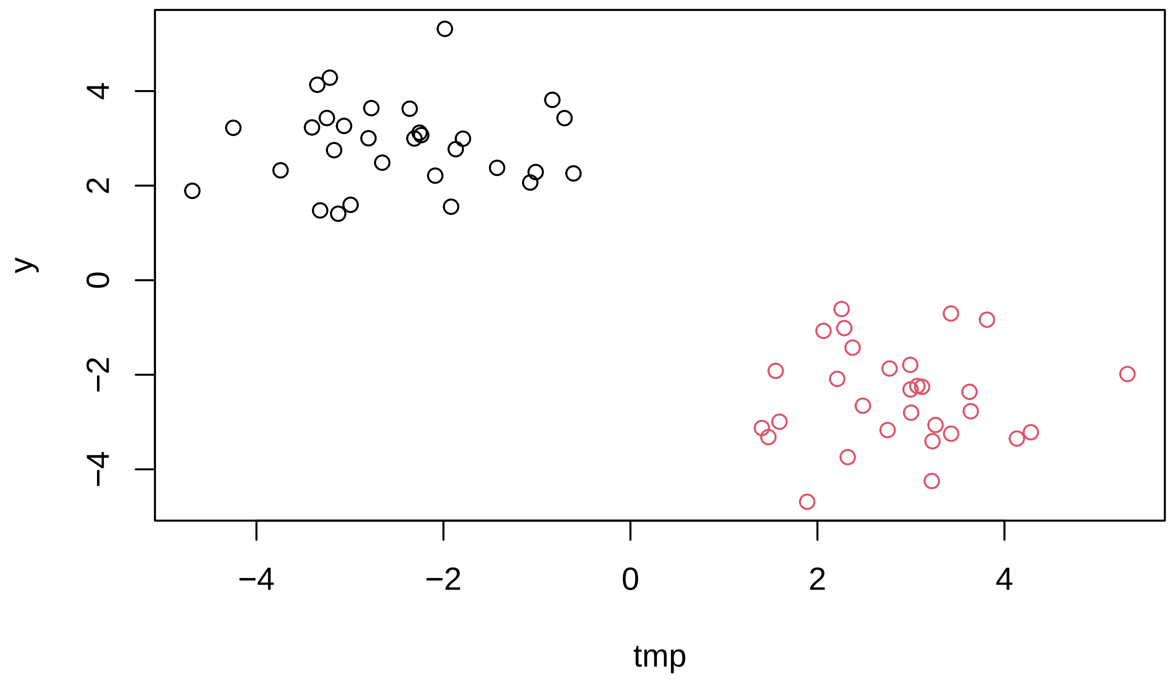
To get our cluster membership vector we need to "cut" the tree with the 'cutree()'.

```
grps <- cutree(hc, h=8)
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 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)

PCA of UK food data

Read data from website and try a few visualizations.

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

Data Import

> Q1. How many rows and columns are in your new data frame named 'x'? Whar R functions could you use to answers this question?

```
dim(x)
```

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

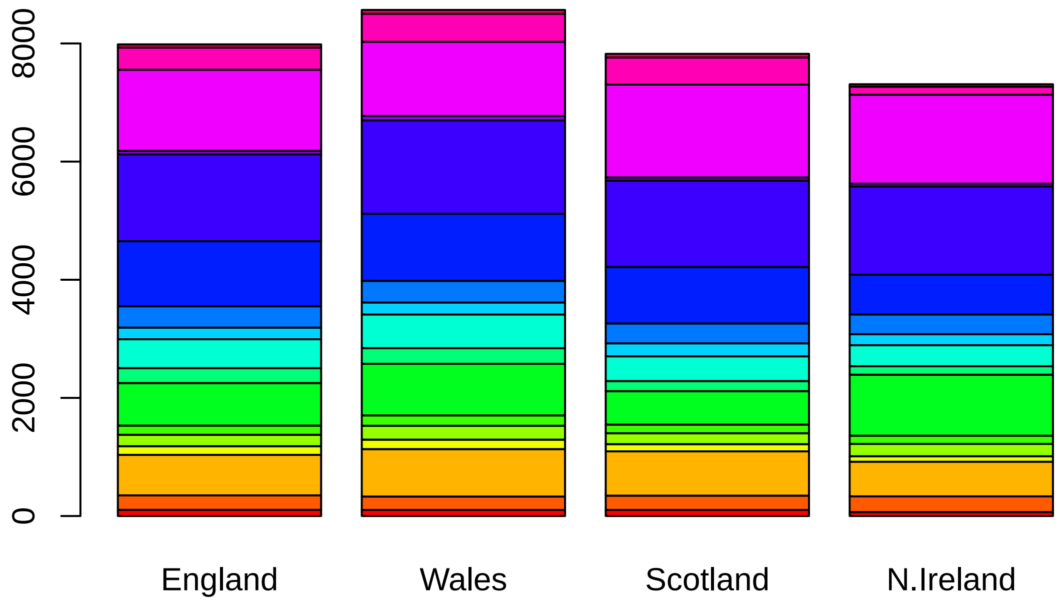
There are 17 rows and 4 columns for the new data frame named 'x'. The R functions that can be used to answer this are 'dim(x)' for both or 'ncol()' and 'nrow()' seperately.

Checking your data

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

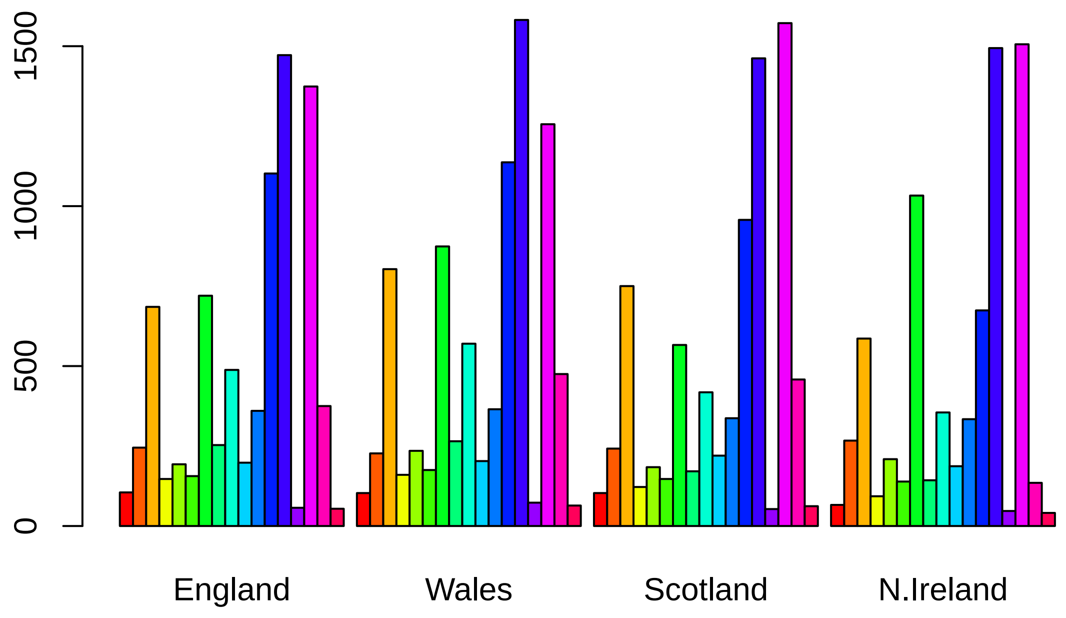
I prefer using the argument setting 'row.names()' to set the correct row-names because it only returned the data frame with the row names changed. 'Rownames()' is a non-generic function, while 'row.names()' is a generic function and is specific for data frames.

```
cols <- rainbow(nrow(x))
barplot( as.matrix(x), col=cols )
```



Spotting major differences and trends

```
barplot( as.matrix(x), col=cols, beside=TRUE )
```



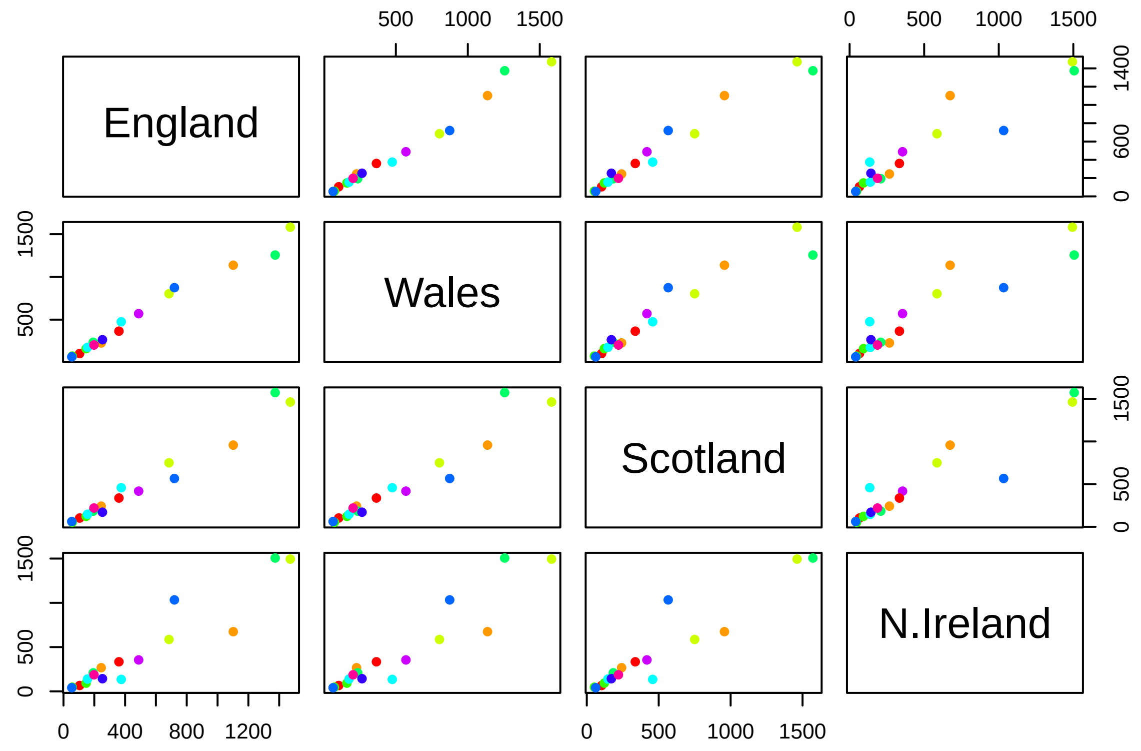
Q3. Changing what optional argument in the above barplot() function results in the following plot?

Changing 'beside=TRUE' changes each column to be next to each other rather than 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?

The following creates a matrix of scatter plots to help understand the pairwise relationship between the different variables in the data set. If a given point lies on the diagonal for a given plot, it means that the same amount of people eat the same food in the different countries.

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



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

N. Ireland generally consumes less fresh fruit and eats more potatoes and soft drinks than the other countries of the UK.

PCA to the rescue

The main base R PCA function is called 'prcomp()' and we will need to give it the transpose of our input data!

```
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)
```

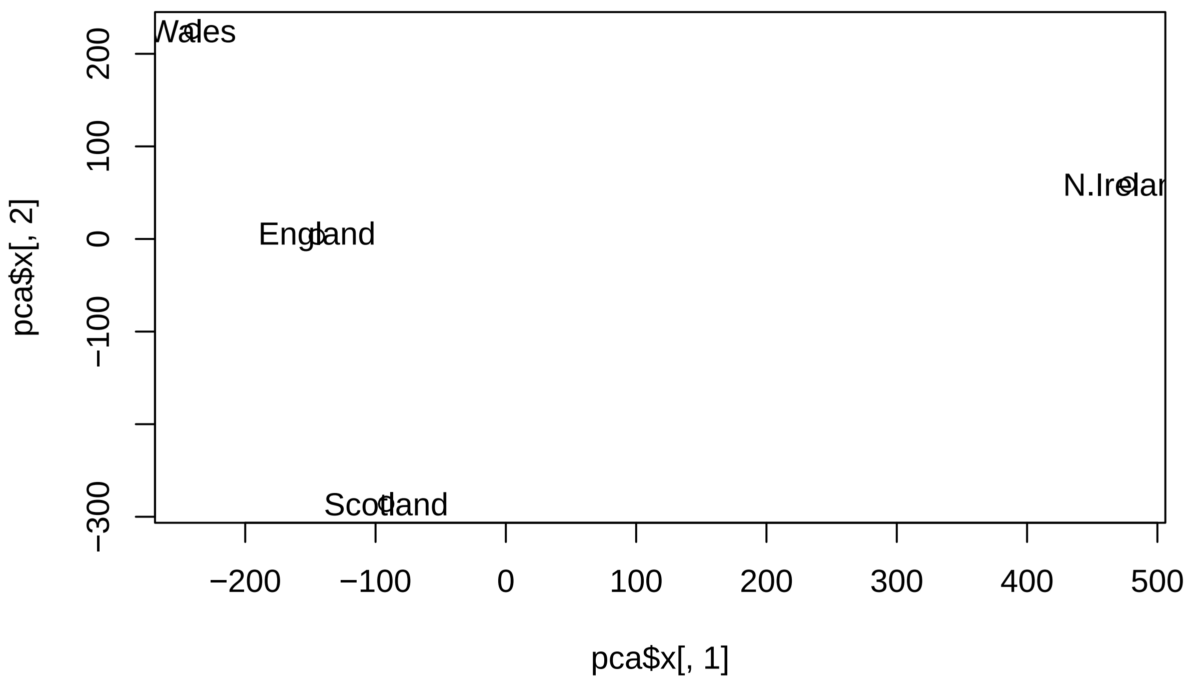
```
## $names
## [1] "sdev"     "rotation" "center"   "scale"    "x"
##
```

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

To make our new PCA plot (a.k.a. PCA score plot) we access 'pca$x'.

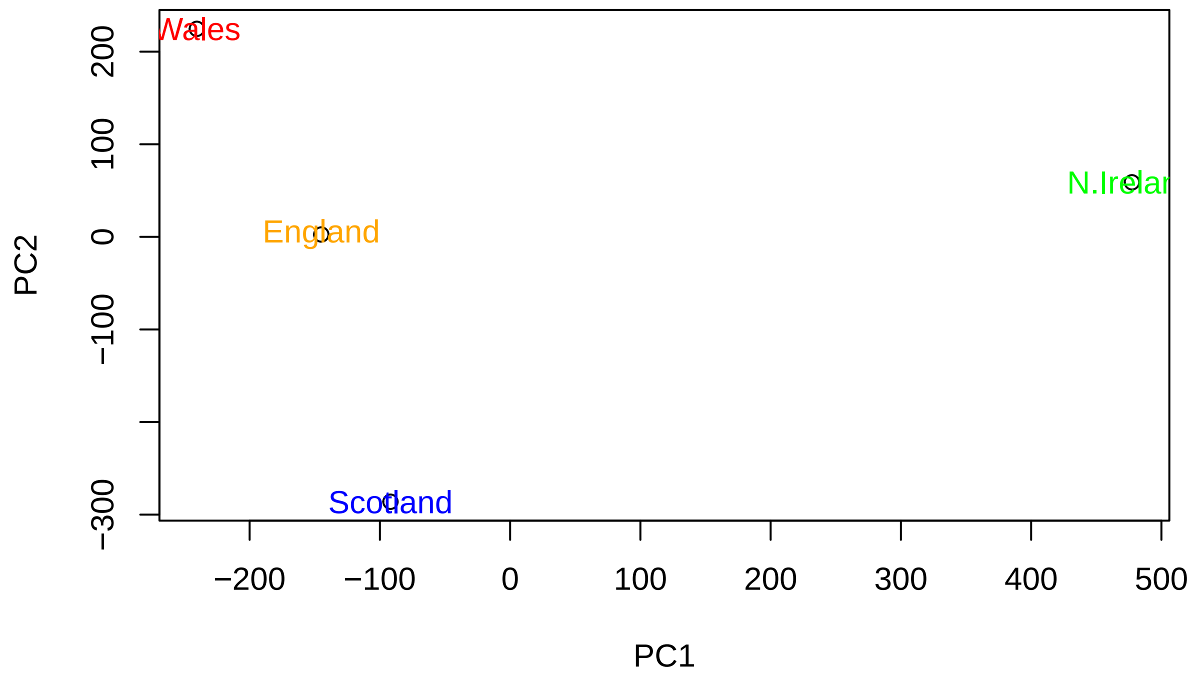
> Q7. Complete the code below to get the plot of PC1 vs PC2. The second line adds text labels over the data points.

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

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



Use square of pca$sdev to calculate how much variation in the original data each PC accounts to

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

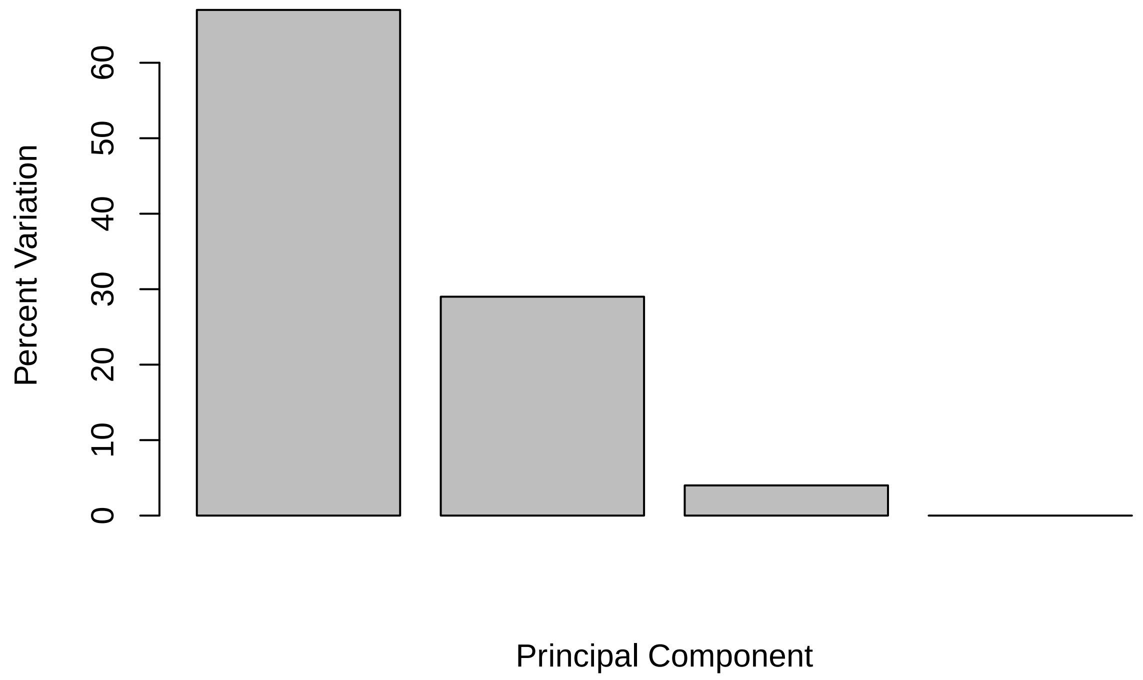
```
## [1] 67 29  4  0
```

Summarize in a plot of variances

```
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
```

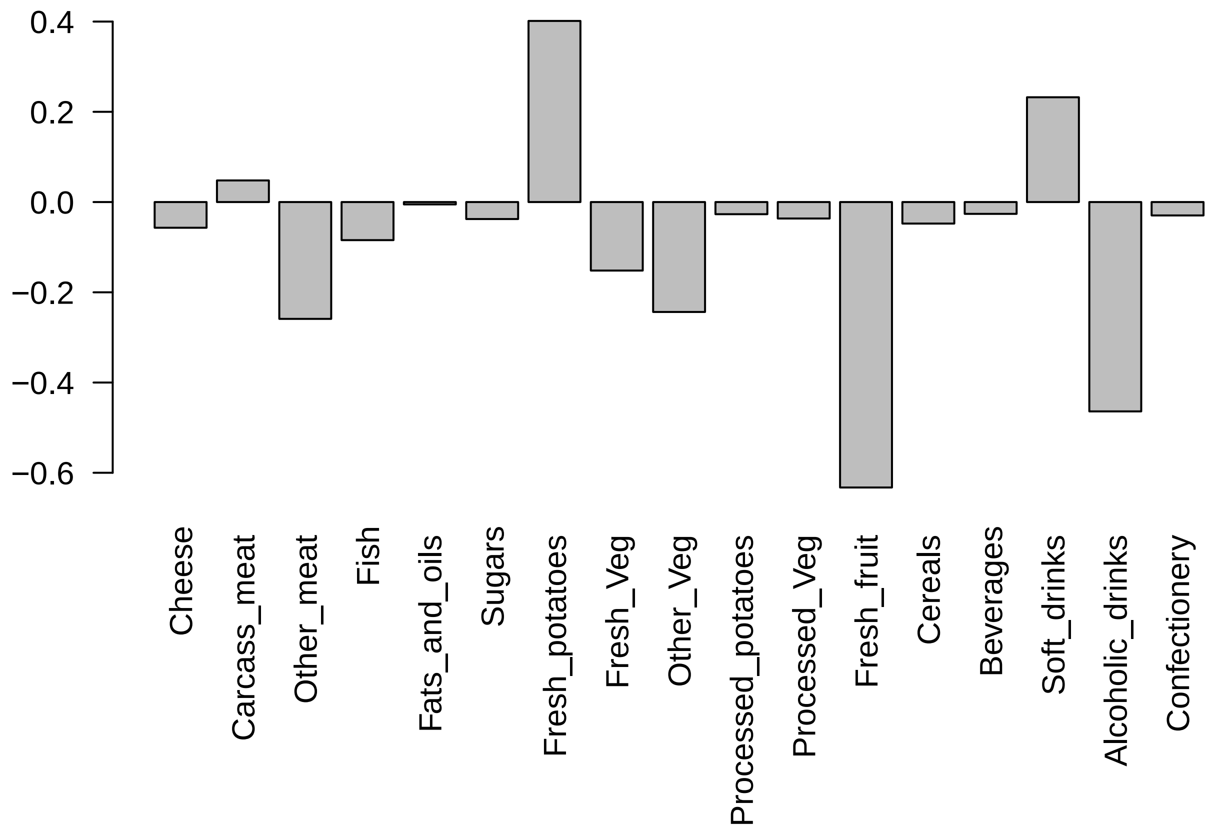
```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```



Digging deeper (variable loadings)

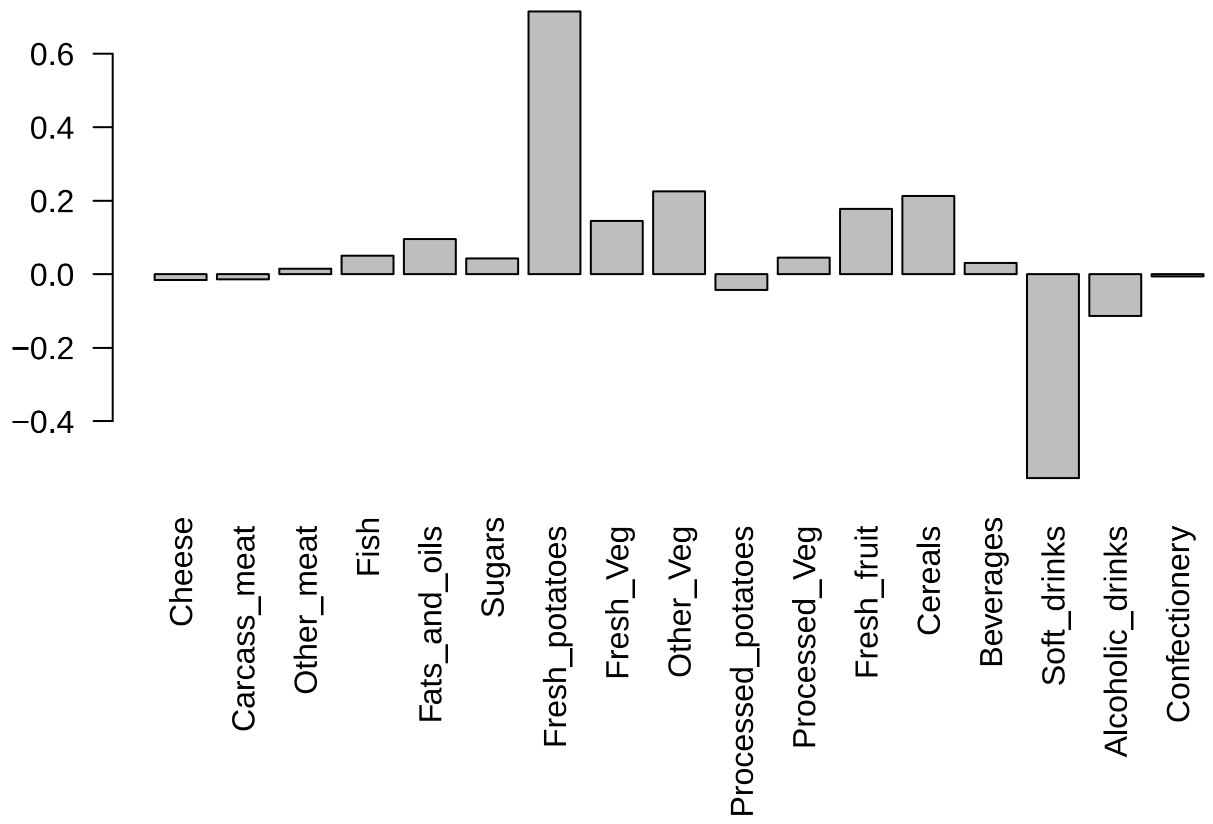
Focus on PC1 as it accounts for >90% of variance

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



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

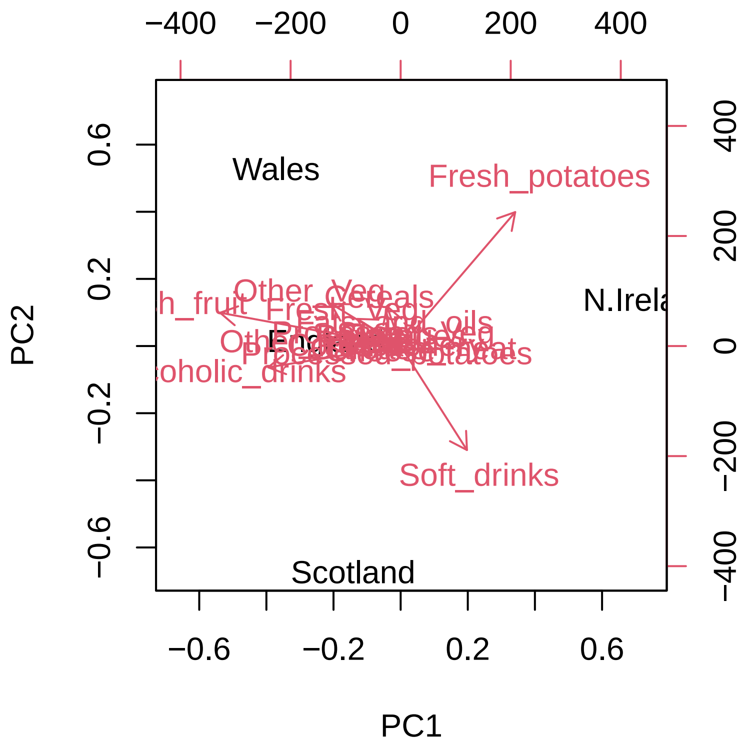
```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```



PC2 mainly shows how N. Ireland eats less fresh fruits and more potatoes and soft drinks.

Biplots

```
biplot(pca)
```



PCR of RNA-seq data

```r
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?

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

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

There are 100 genes and 10 samples in this data set.

Take the transpose of our data

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

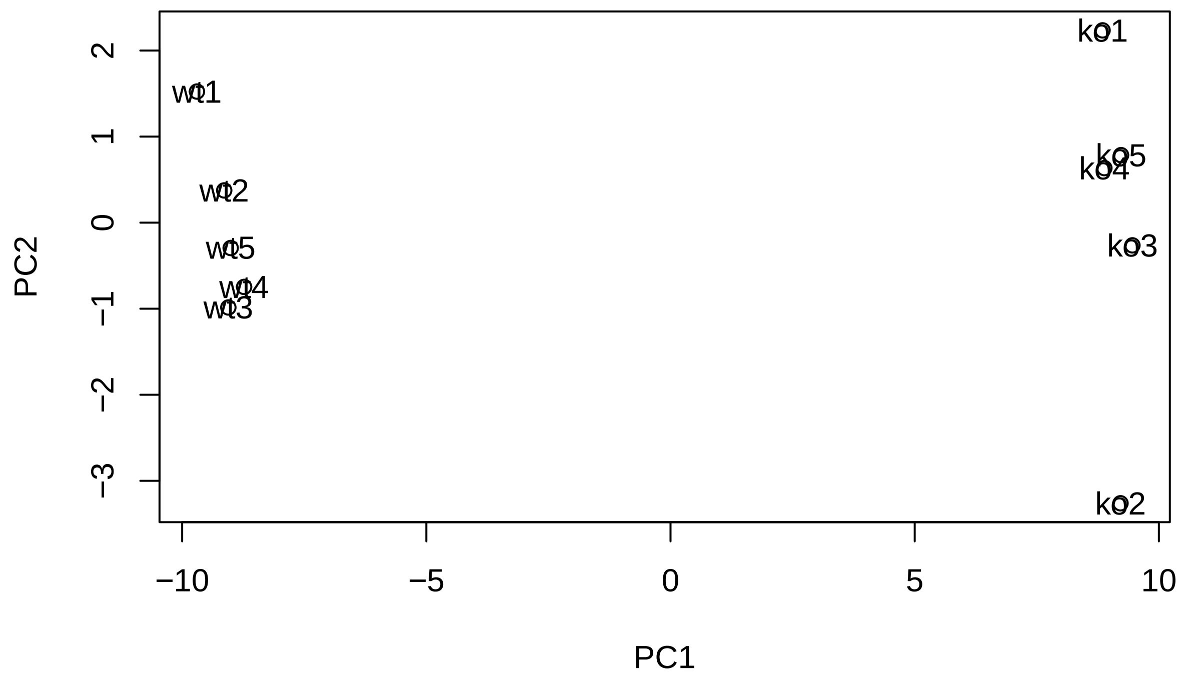
This is a summary of how well PCA is doing

```
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
```

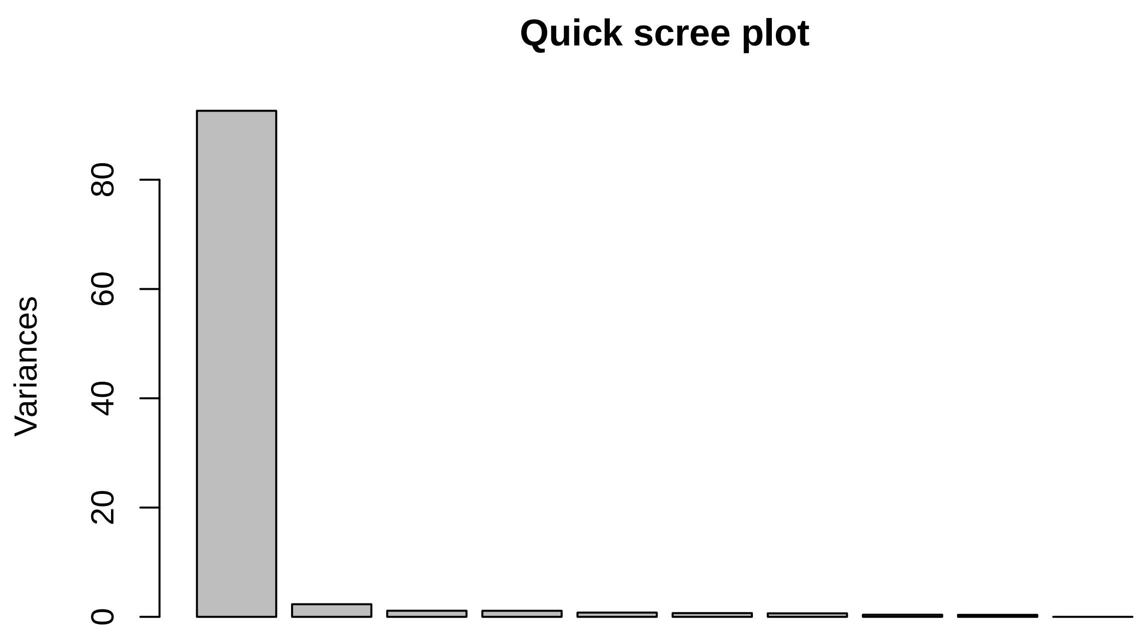
Do our PCA plot of this RNA-seq data

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
text(pca$x[,1], pca$x[,2], colnames(rna.data))
```



Quick barplot summmary of Proportion of Variance for each PC

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



Variance captured per PC

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

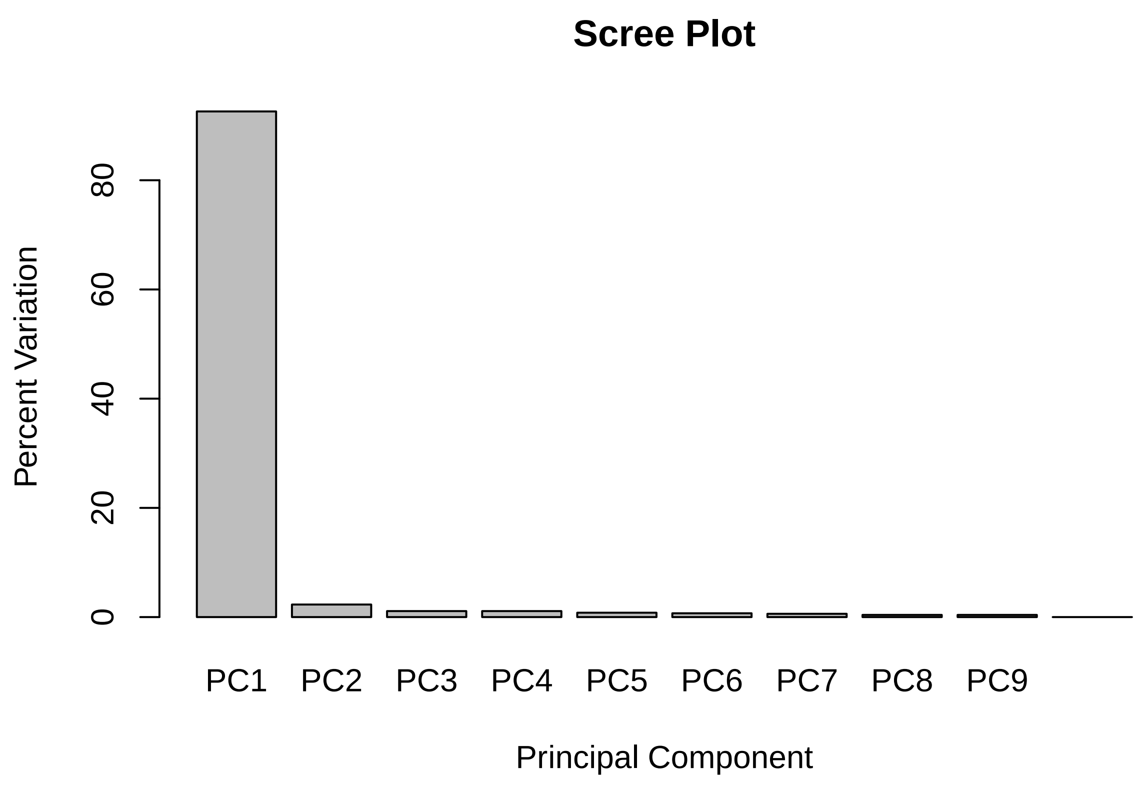
Percent variance

```r
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
```

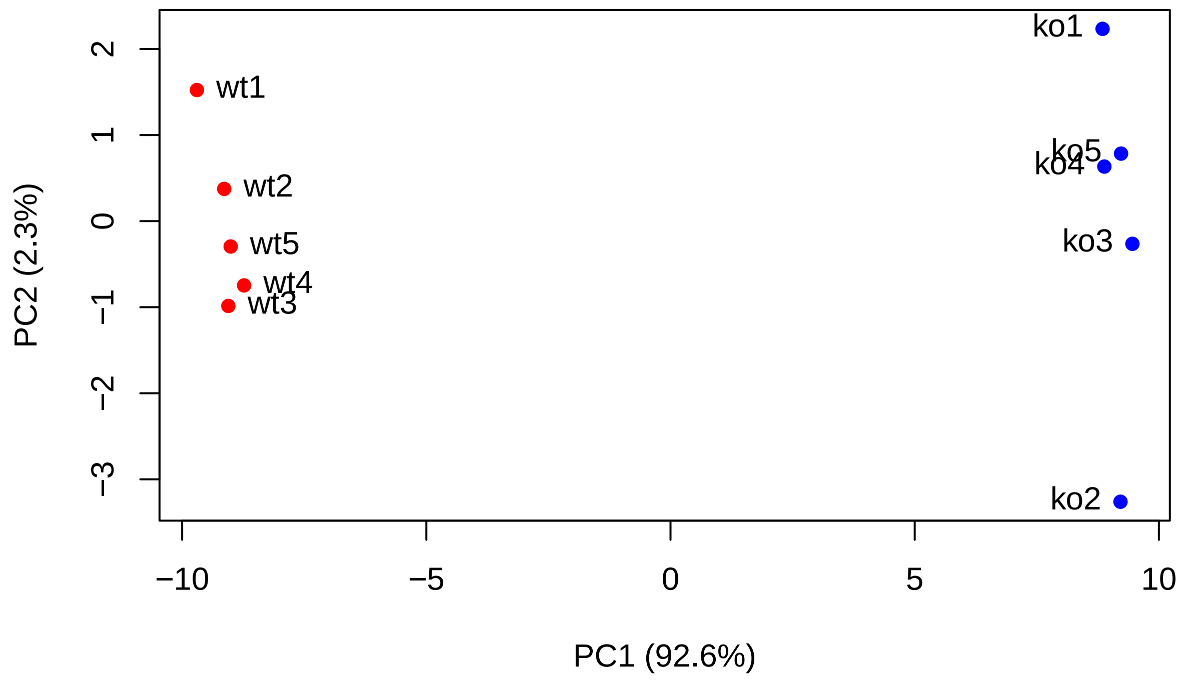
Generate scree-plot

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



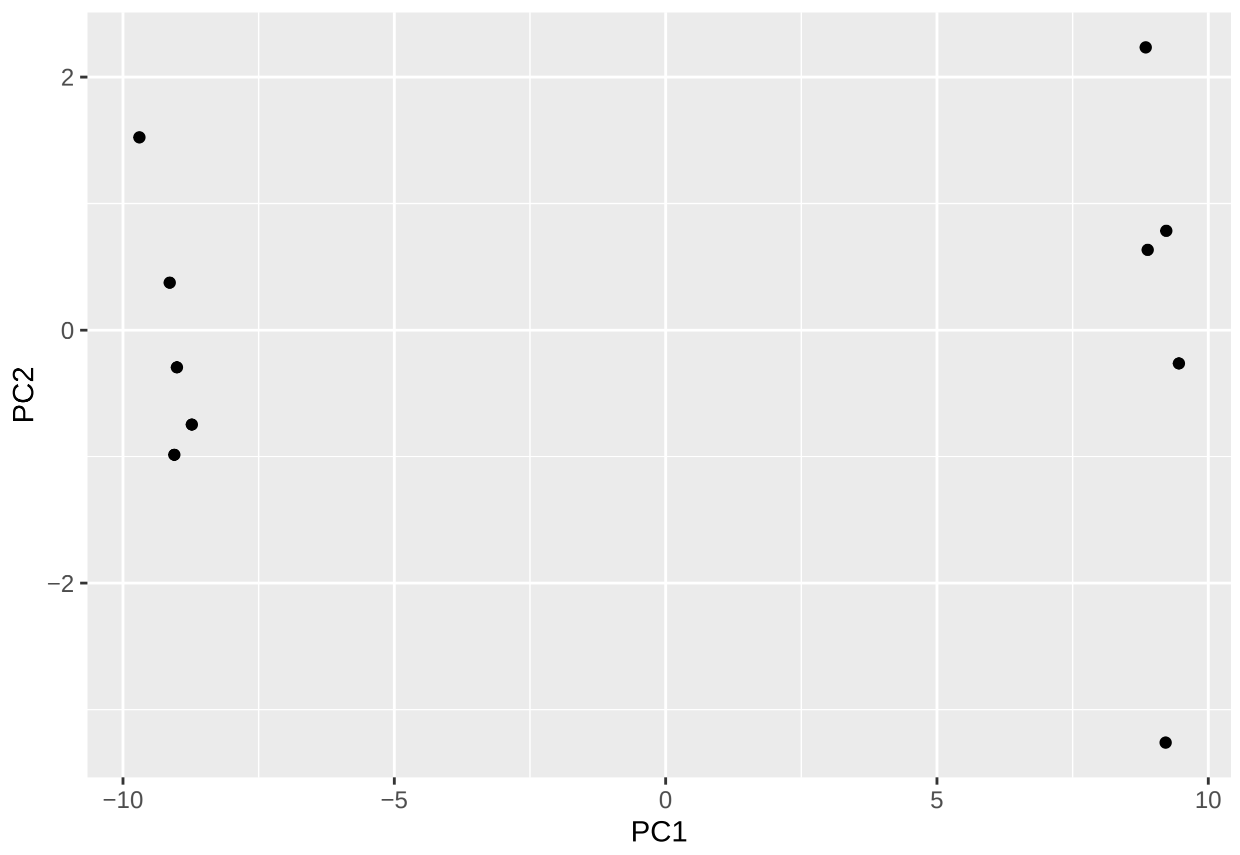
Make main PCR plot more attractive and useful

```
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)))
```



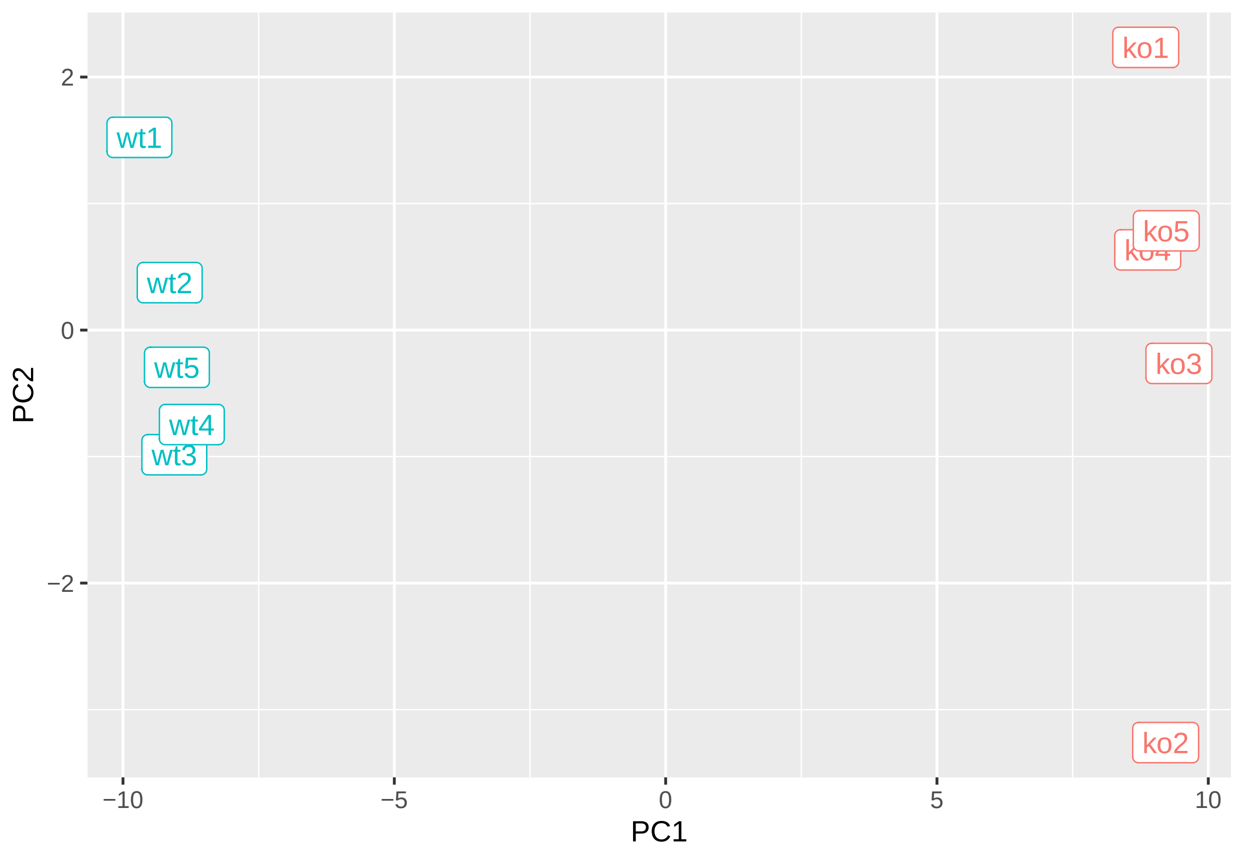
Using ggplot

```r
library(ggplot2)

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

ggplot(df) +
  aes(PC1, PC2) +
  geom_point()
```



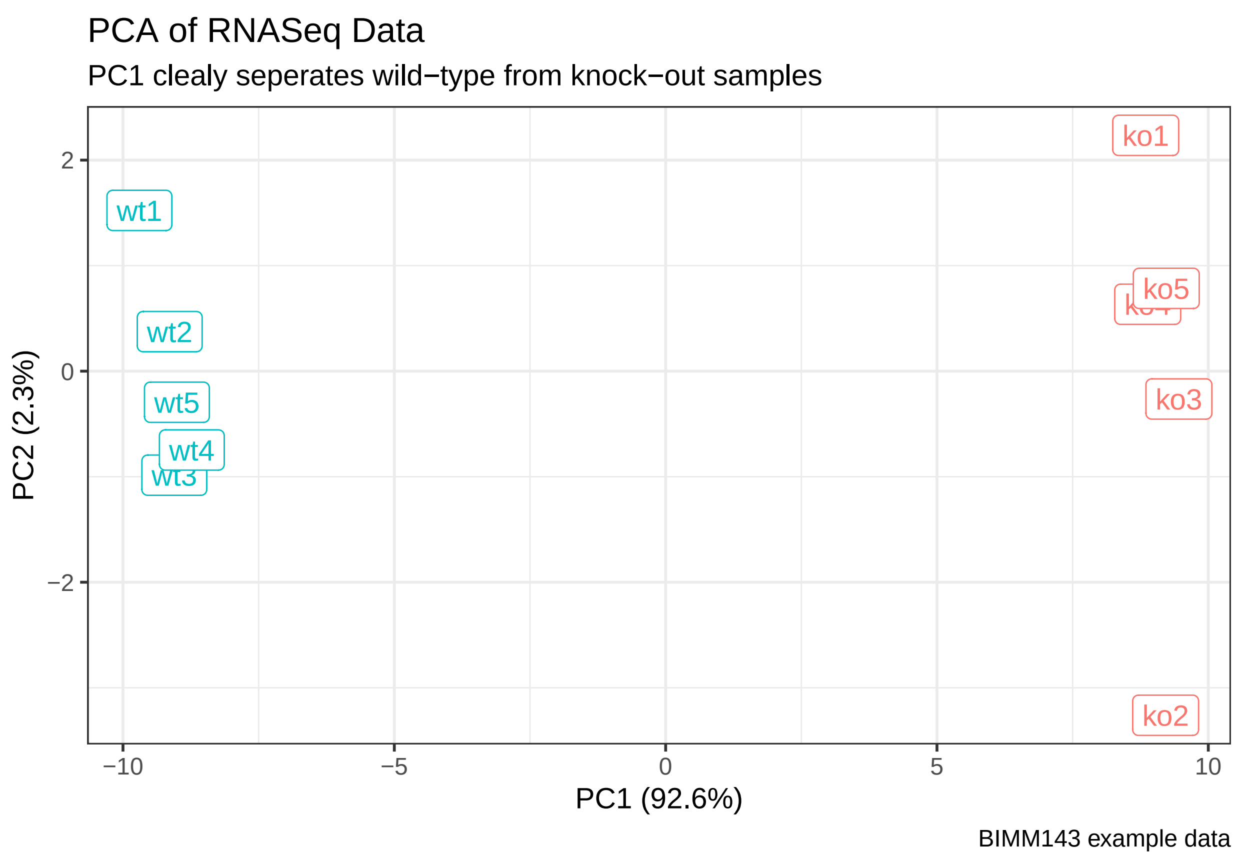
Add specific color and sample label aesthetics for wild-type and knock-out samples

```r
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
```



Add some spit and polish

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