Machine Learning 1

Saumya Ranjan (A15483401) 10/21/2021

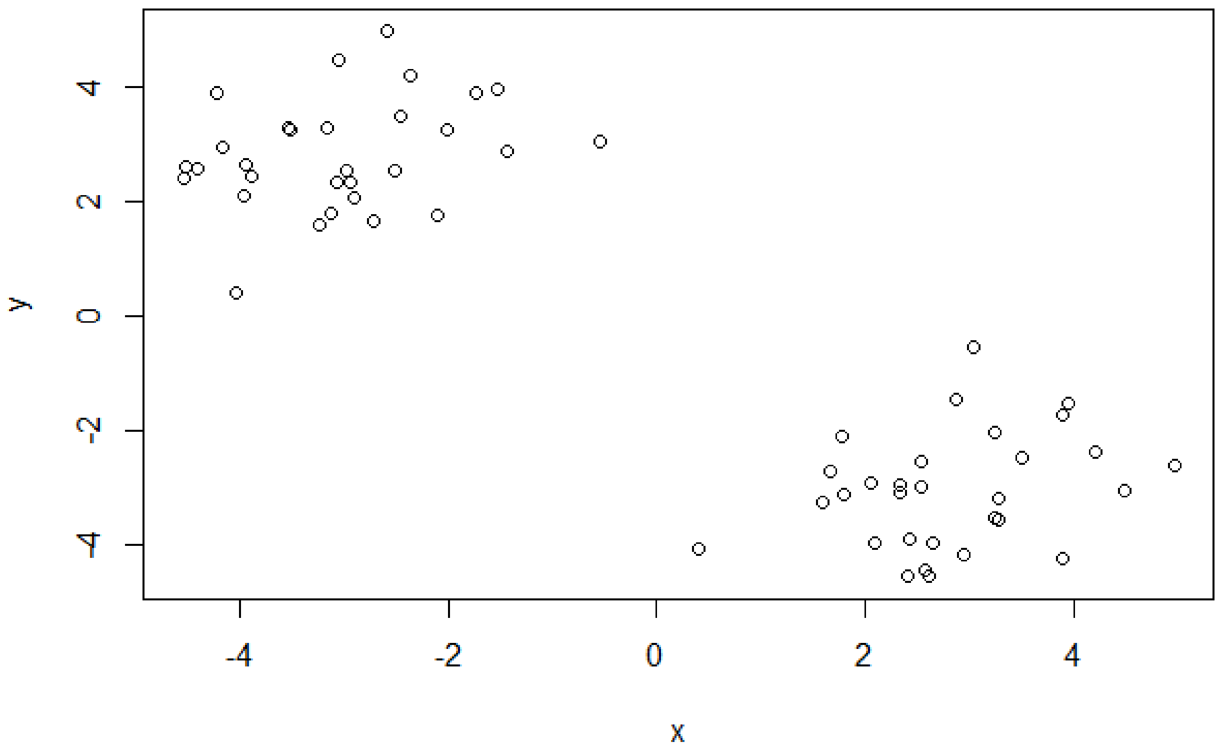
First is clustering methods

K means clustering

The Function in base R to do K means clustering is called “kmeans()”

First 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))
plot(x)
```



Question: 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 -3.040848  2.821928
## 2  2.821928 -3.040848
## 
## 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] 56.18605 56.18605
##  (between_SS / total_SS =  90.2 %)
## 
## Available components:
## 
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Question: How many point are in each cluster?

```
km$size
```

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

30 points in each cluster

Question: What ‘component’ of your result 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 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
```

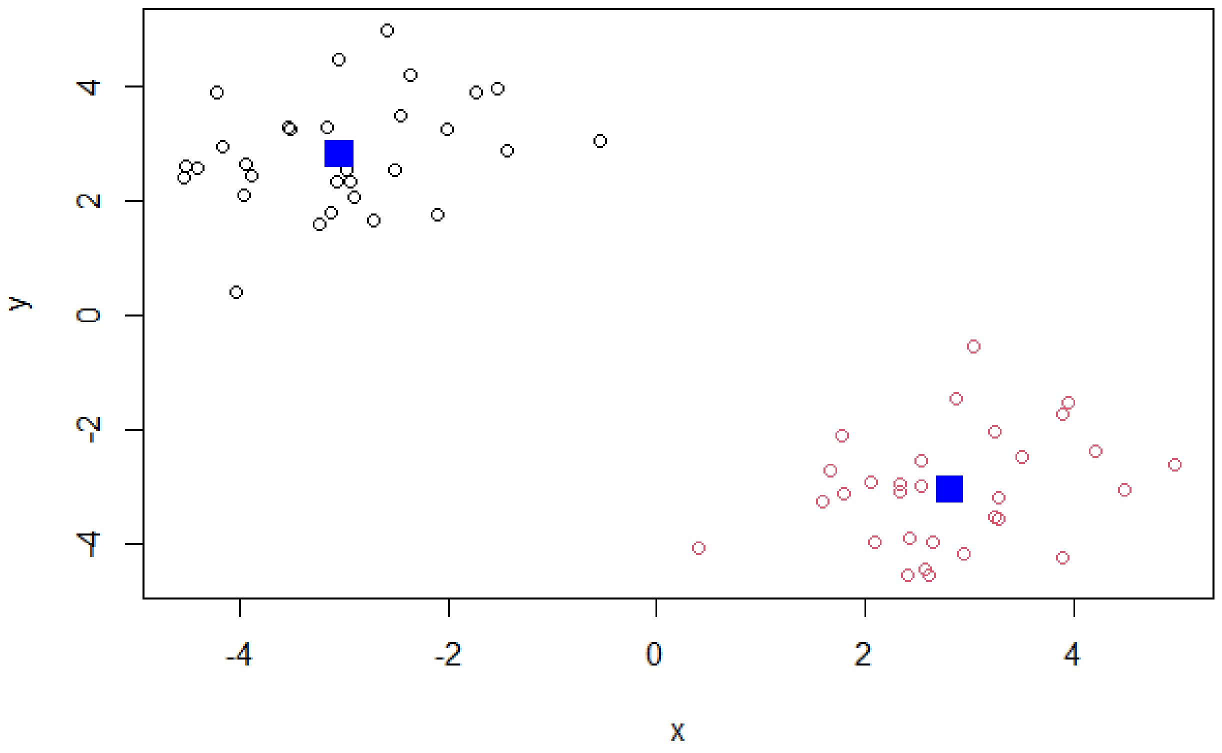
Question: What component of your result object details cluster center?

```
km$centers
```

```
##           x         y
## 1 -3.040848  2.821928
## 2  2.821928 -3.040848
```

Queston: Plot x colored by kmeans cluster assignment and add cluster centers as blue points

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



H clust

A big limitation w kmeans is that we have to tell it K (the number of clusters we want)

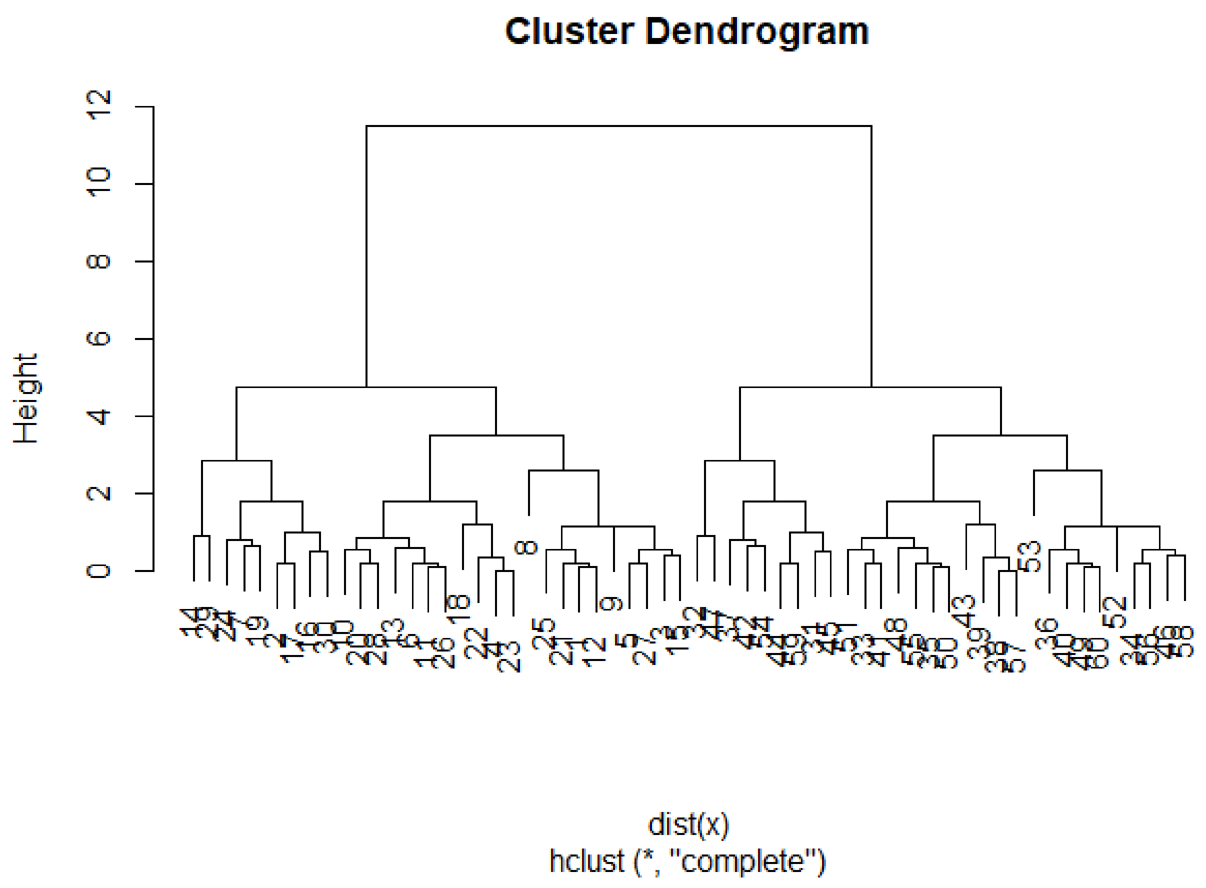
Analyze the same data with hclust

```
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 bit more work. We have to cut the tree where we think it makes more sense. For this we use ‘cutree()’ function

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

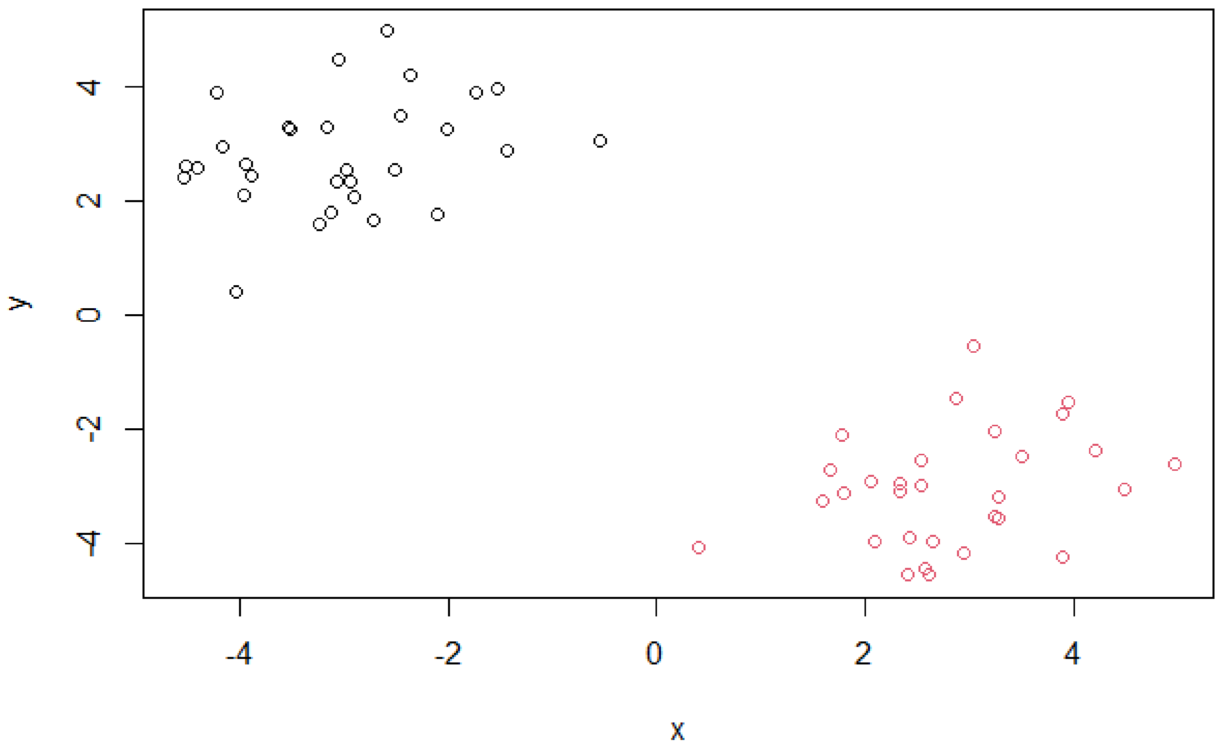
```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

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

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

Make our results plot

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



PCA of UK food data

```
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?

```
nrow(x)
```

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

```
ncol(x)
```

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

Checking your data

```
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
```

```
tail(x)
```

```
##                    X England Wales Scotland N.Ireland
## 12       Fresh_fruit    1102  1137      957       674
## 13           Cereals    1472  1582     1462      1494
## 14         Beverages      57    73       53        47
## 15       Soft_drinks    1374  1256     1572      1506
## 16 Alcoholic_drinks      375   475      458       135
## 17    Confectionery       54    64       62        41
```

Want to have 4 columns. Right now the row names are set to be one column of its own

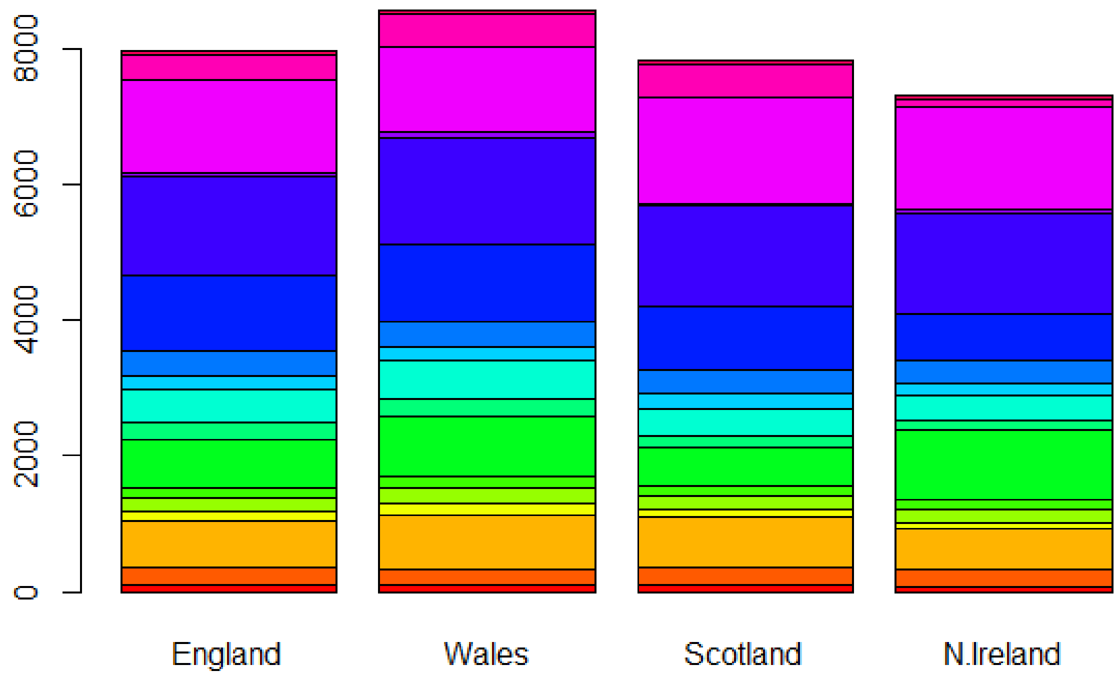
```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names=1)
head(x)
```

```
##                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
```

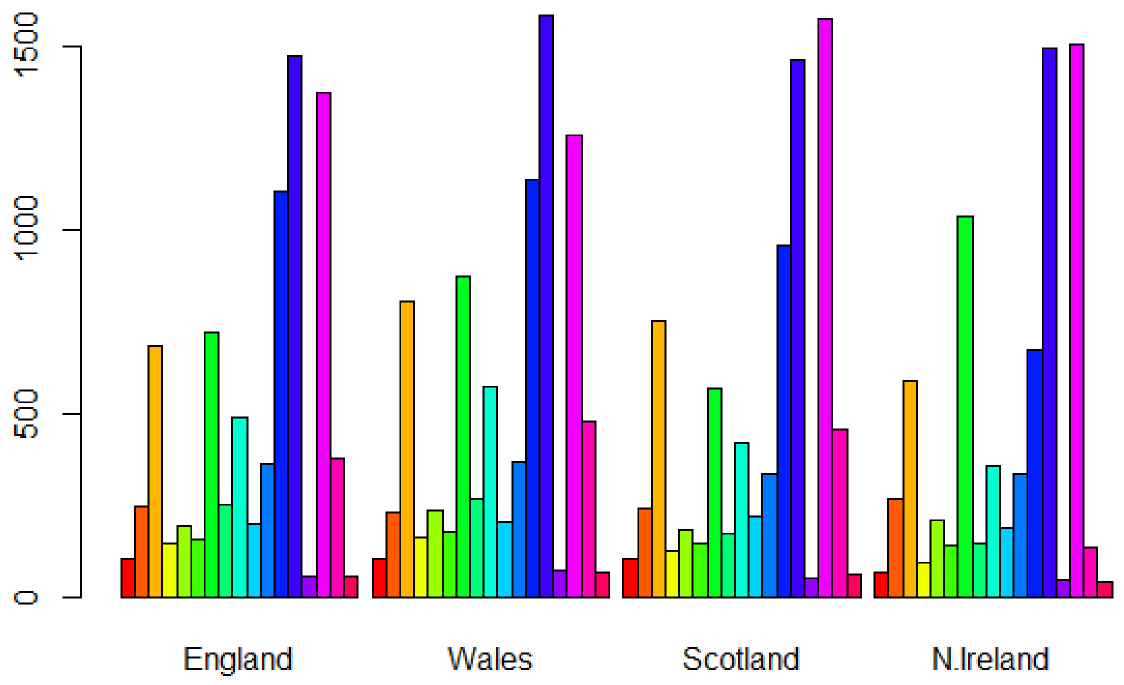
Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances? – the one where we change the row name in the beginning. If we use "x <- x[,-1]" then every time we run the code, a column gets deleted.

Now we have the data looking good we want to explore it. We will use some conventional plots

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



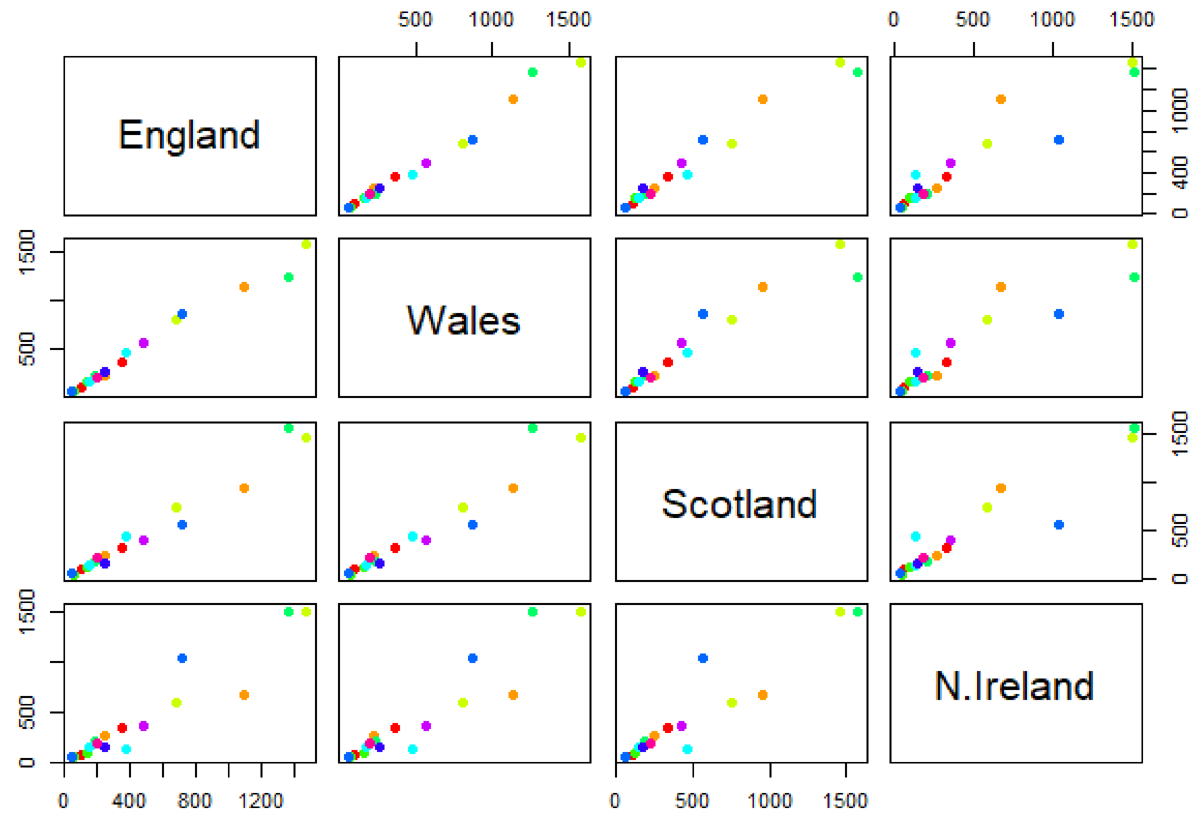
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



Q3: Changing what optional argument in the above barplot() function results in the following plot? –changing the “beside” argument. If beside= true, the bars will be beside each other.

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)
```



The function creates plots that pair 2 countries and plots the results in a scatter plot. If the line lies on the diagonal it means that the value for the 2 countries are the same; there is no variation between the 2 for that catagory.

Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set? – we can compare N Ireland to the rest of the UK by looking at how a category differs from the other countries. Like I can see in the bar graph that N Ireland eats less of what the “’light blue” group compared to the rest of the UK. But making comparisons are hard.

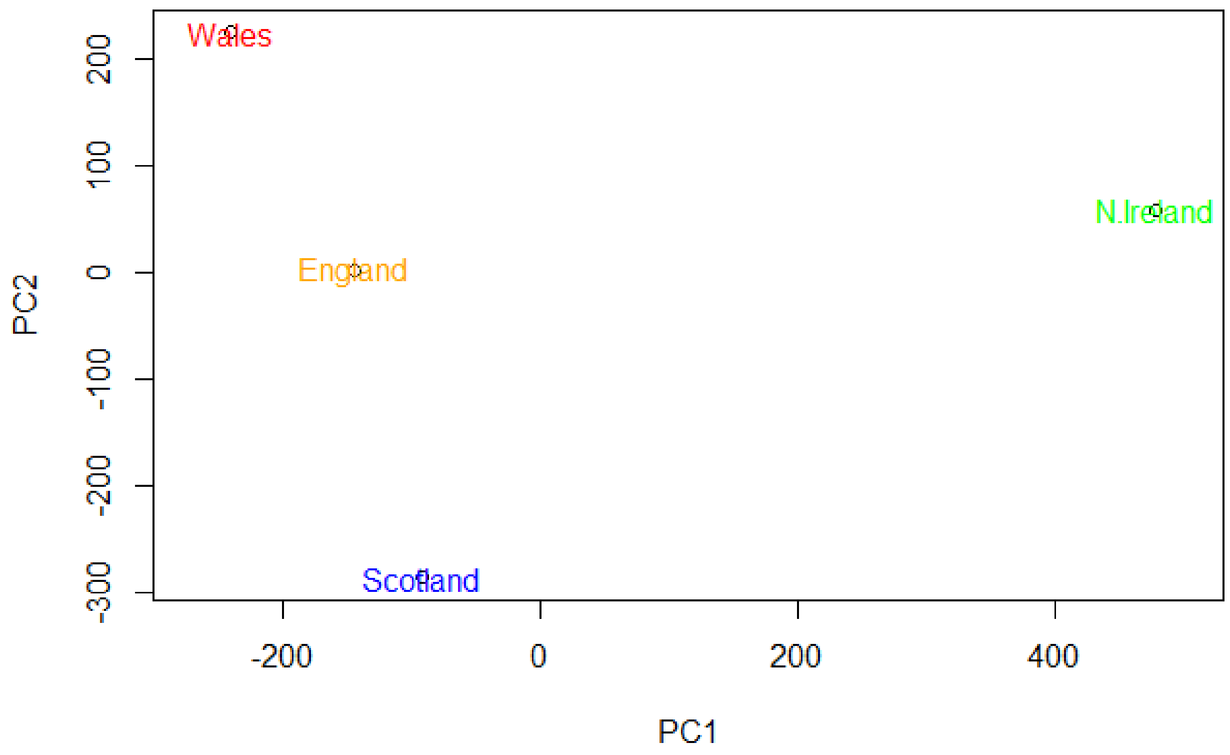
##PCA to the rescue

```
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
```

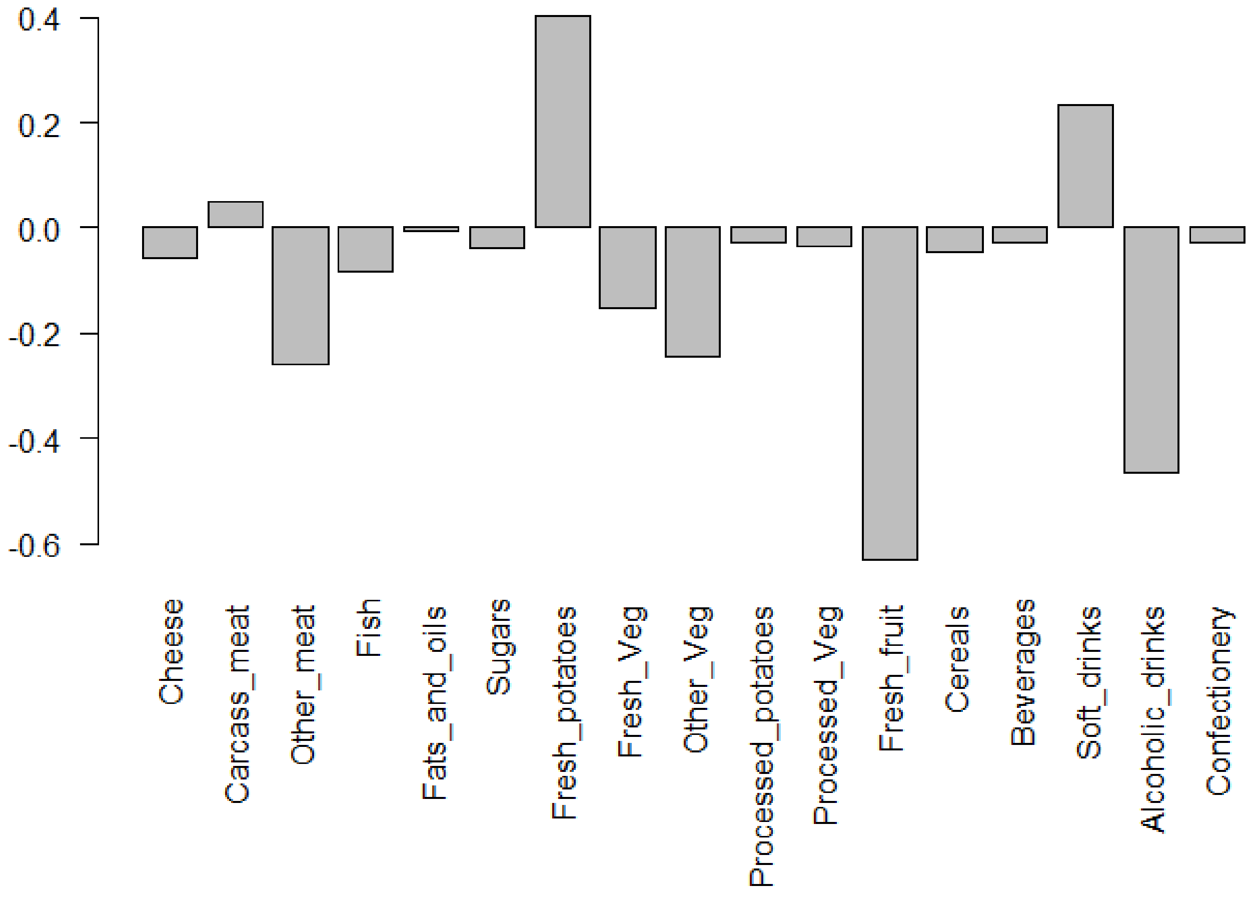
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
# 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), col=c("orange", "Red", "blue", "green"))
```



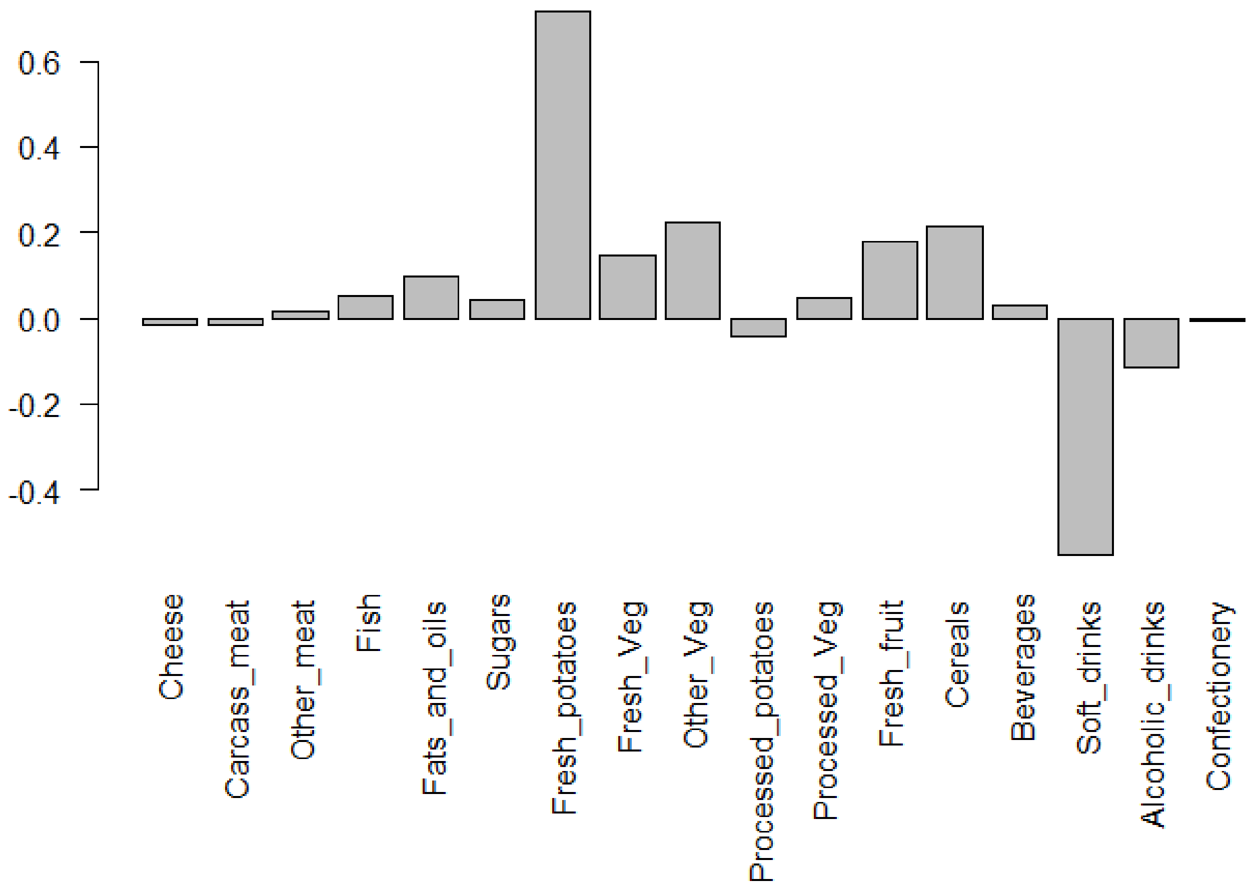
##Digging deeper (variable loadings)

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



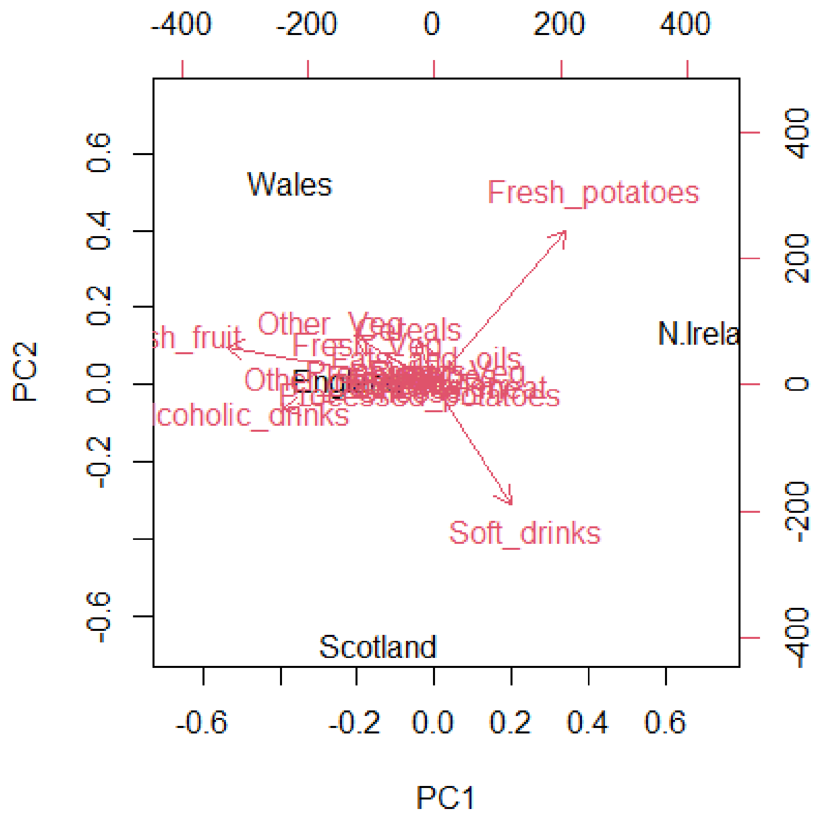
Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about?

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



– the two food groups that feature are fresh potatoes and soft drinks. It tells us that the variation for N Ireland (where most of the differences between N Ireland and the rest of the UK) lies in fresh potato and soft drink consumption

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



##2. PCA of RNA-seq data

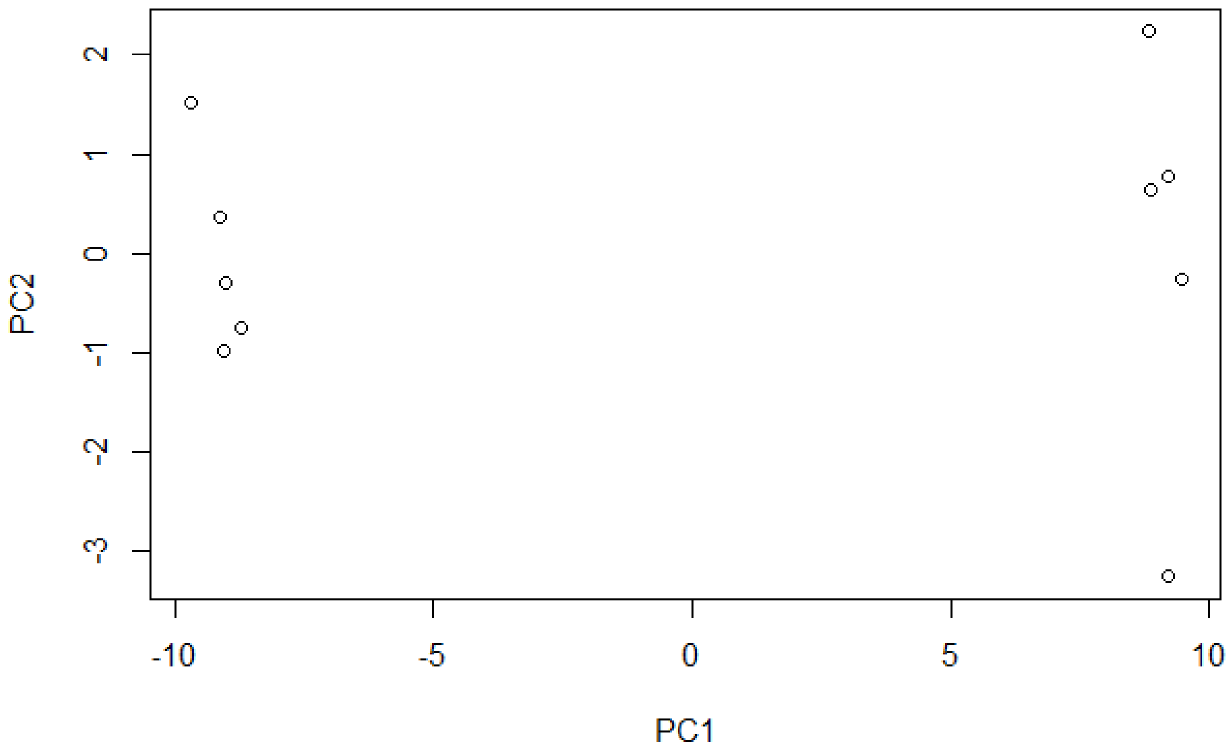
```
url2 <- "https://tinyurl.com/expression-CSV"
rna.data <- read.csv(url2, row.names=1)
head(rna.data)
```

```
##        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? – There are 100 genes and 10 samples

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

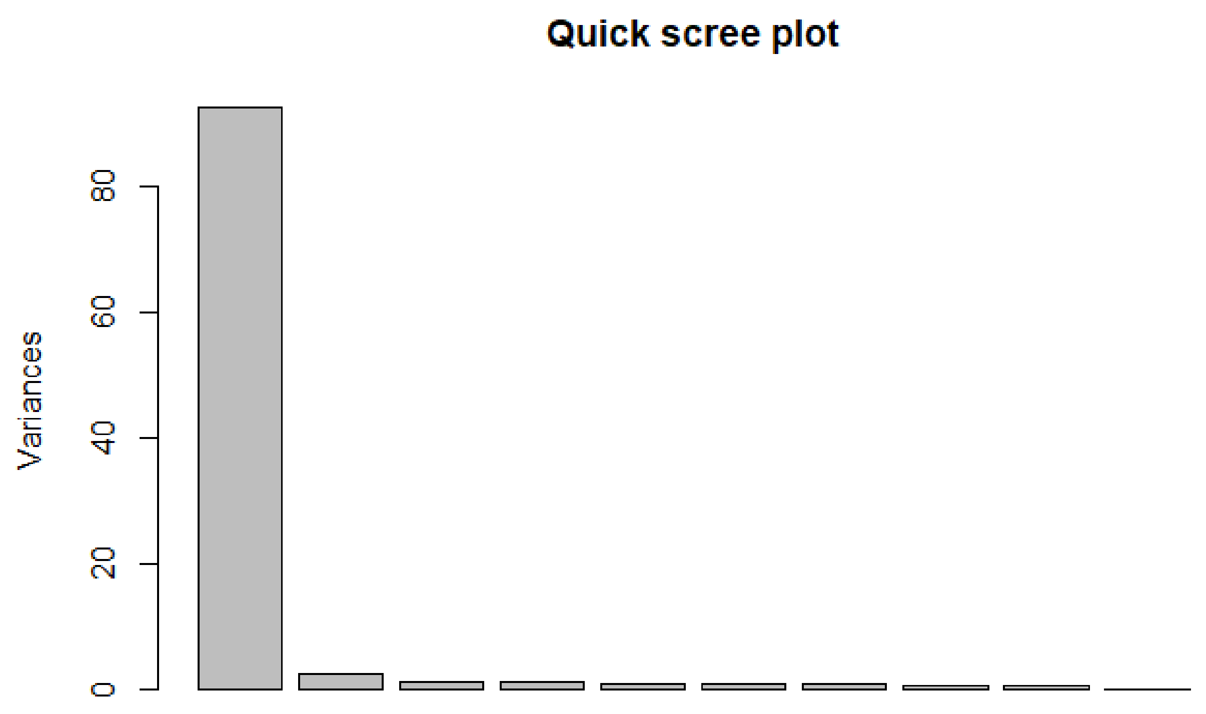
```
## Simple un polished plot of pc1 and pc2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```



```
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
```

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

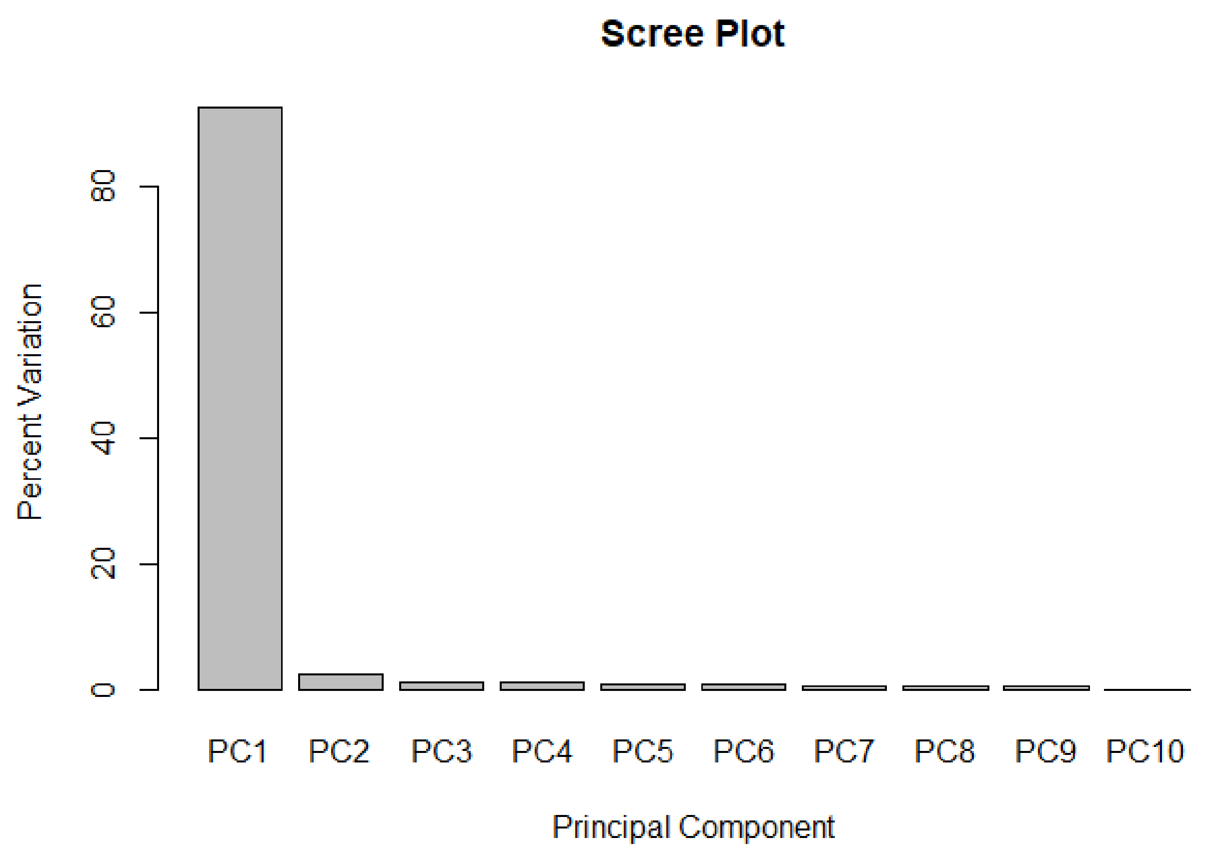


Making the scree plot ourselves

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

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

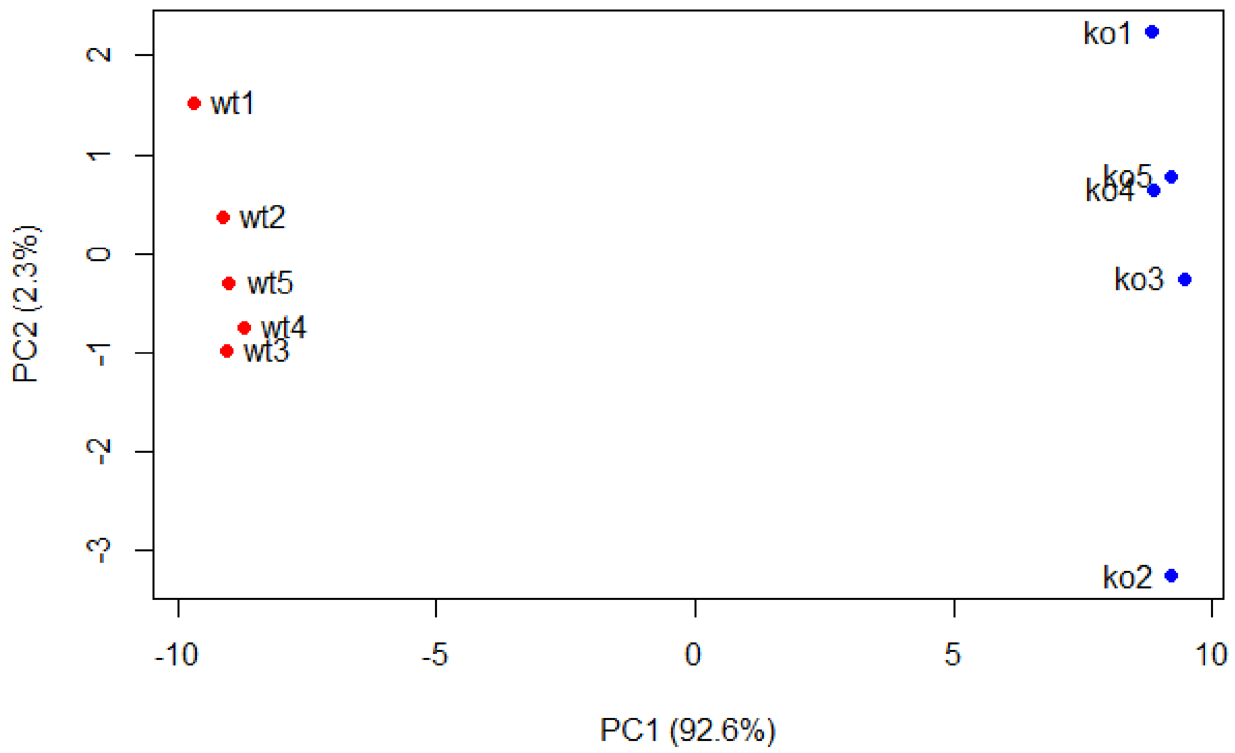
```
##  [1] 92.6  2.3  1.1  1.1  0.8  0.7  0.6  0.4  0.4  0.0
```

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



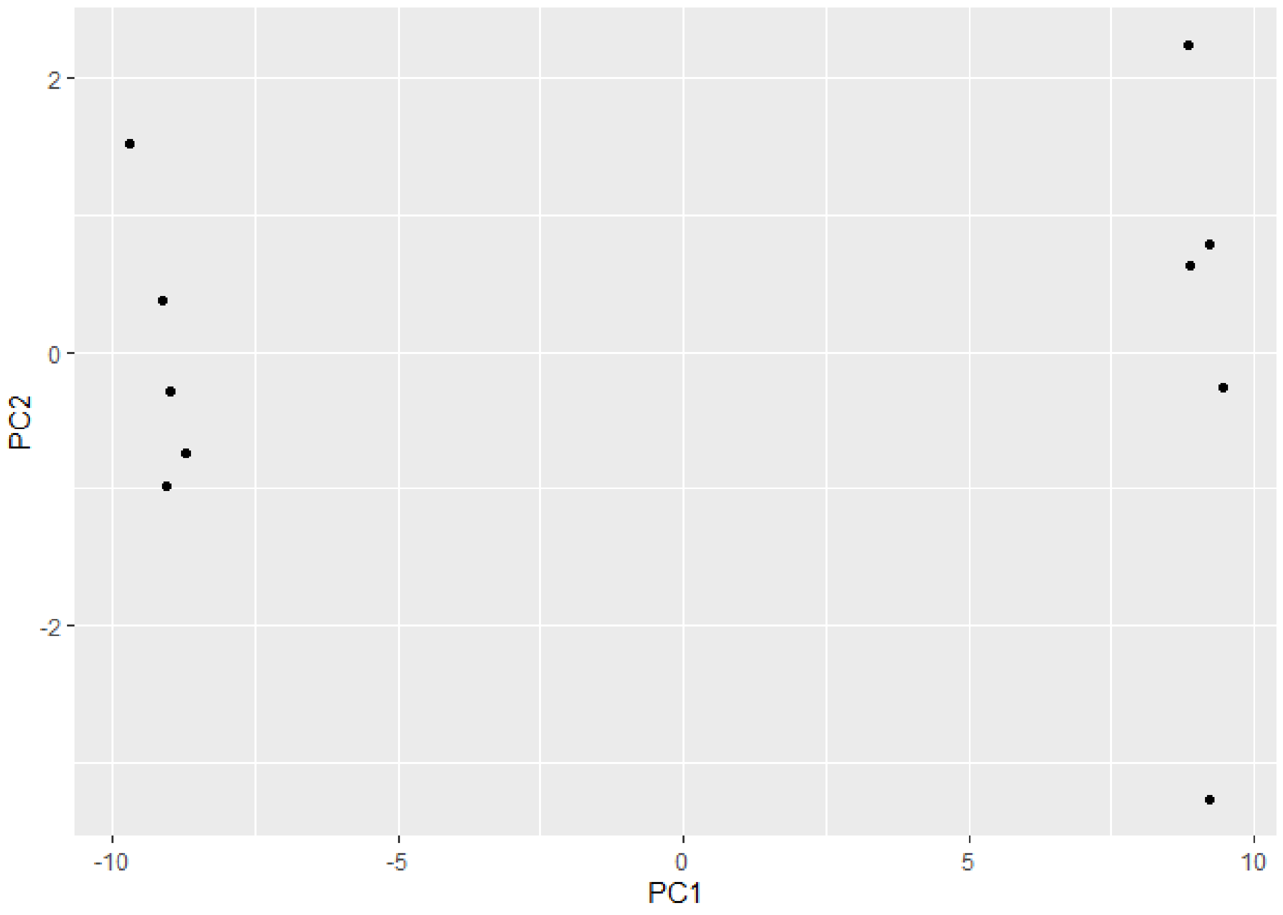
Making the PCA plot look colorful

```
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)))
```



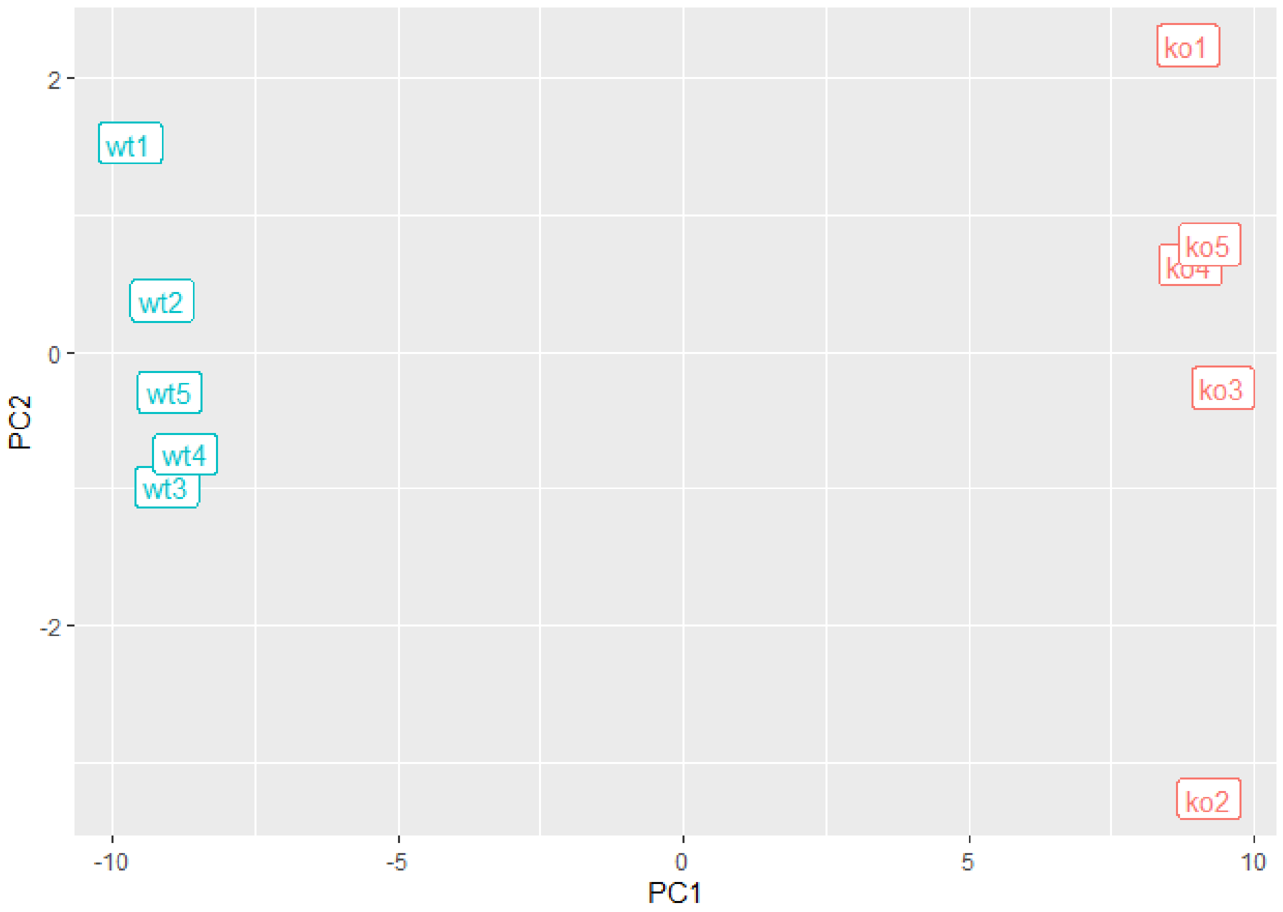
Using ggplot

```
library(ggplot2)

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

# Basic PCA1 vs PCA2 plot
ggplot(df) +
  aes(PC1, PC2) +
  geom_point()
```



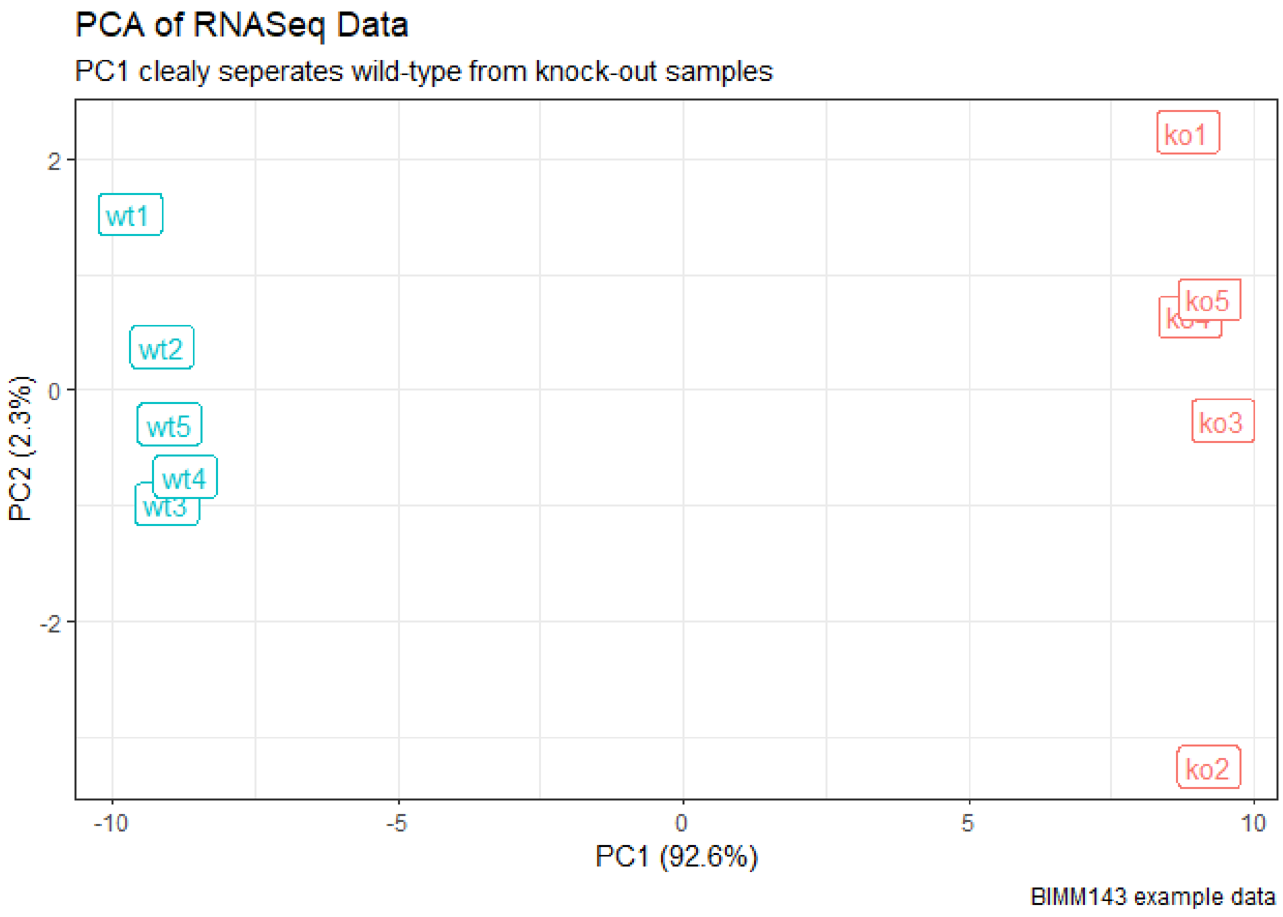
Labeling WT and knockout samples

```
# 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
```



Adding themes and title+subtitles + percent var to axis labels

```
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
PC2 (2.3%)
PC1 (92.6%)
BIMM143 example data