MachineLearning1

Thisha Thiagarajan A15474979

10/21/2021

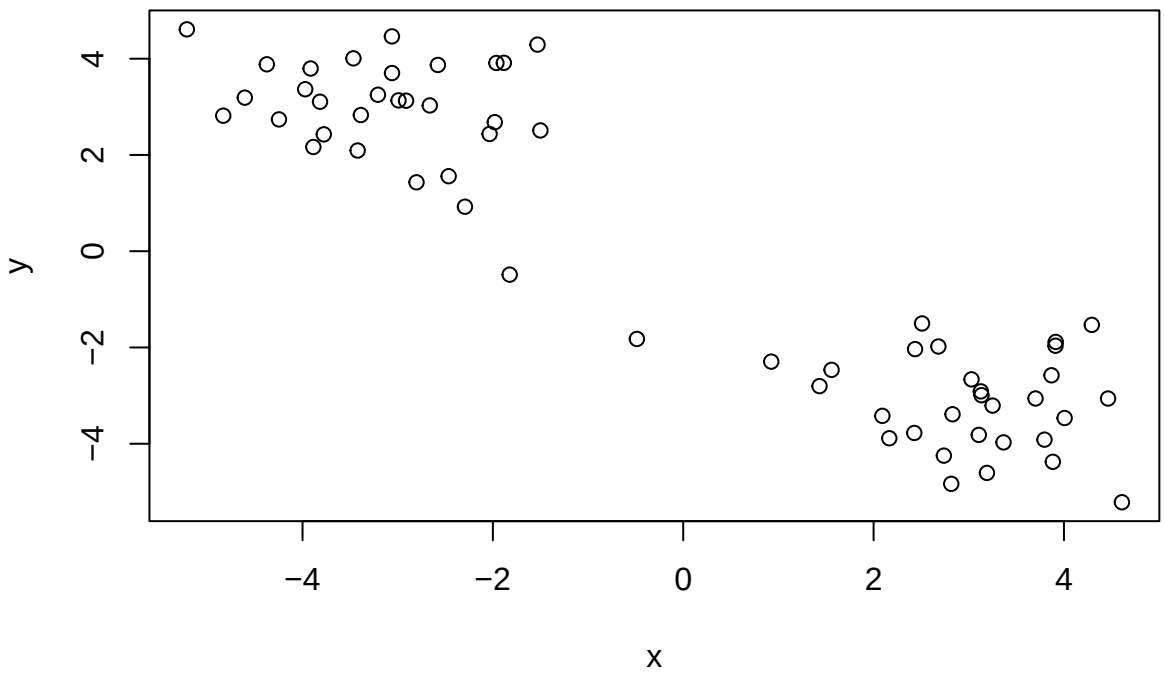
First up is clustering methods

Kmeans Clustering

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

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

```
#rnorm generates vector of normal distribution 30 values, centered around -3/3
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 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 -3.122263  2.959098
## 2  2.959098 -3.122263
##
## 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] 65.09791 65.09791
##  (between_SS / total_SS =  89.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Q. How many plots are in each cluster?

30, 30. We can use the function size to determine this information.

```
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 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
```

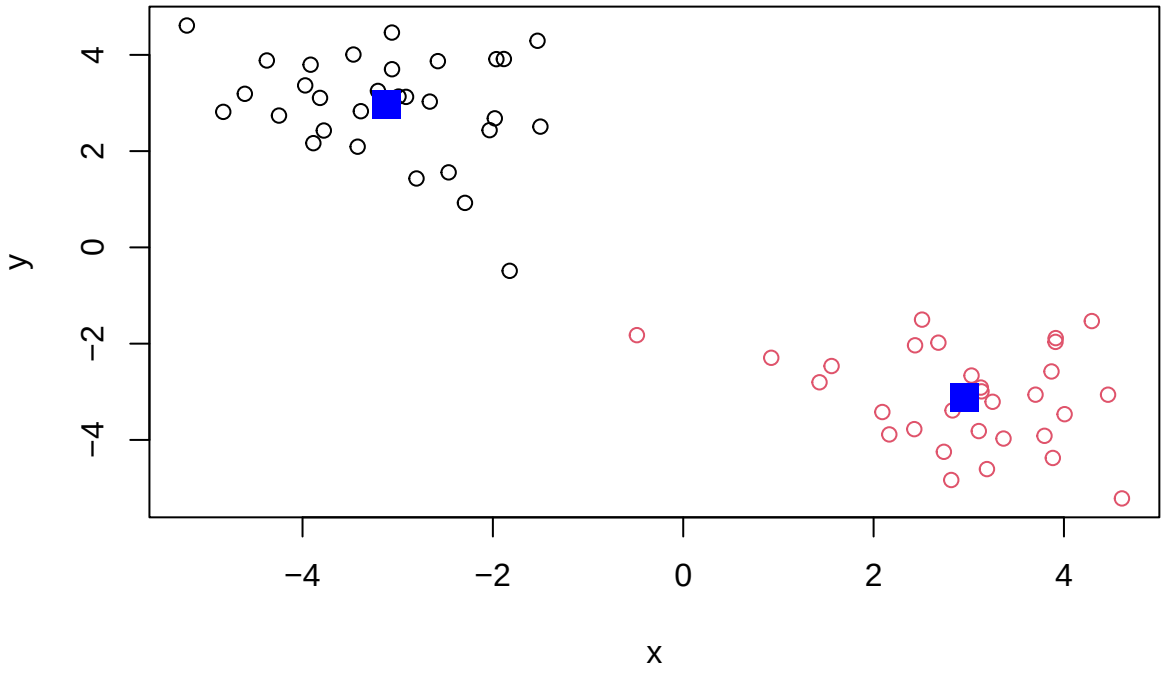
Q. What 'component' of your result object details cluster center?

```
km$centers
```

```
##           x         y
## 1 -3.122263  2.959098
## 2  2.959098 -3.122263
```

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()

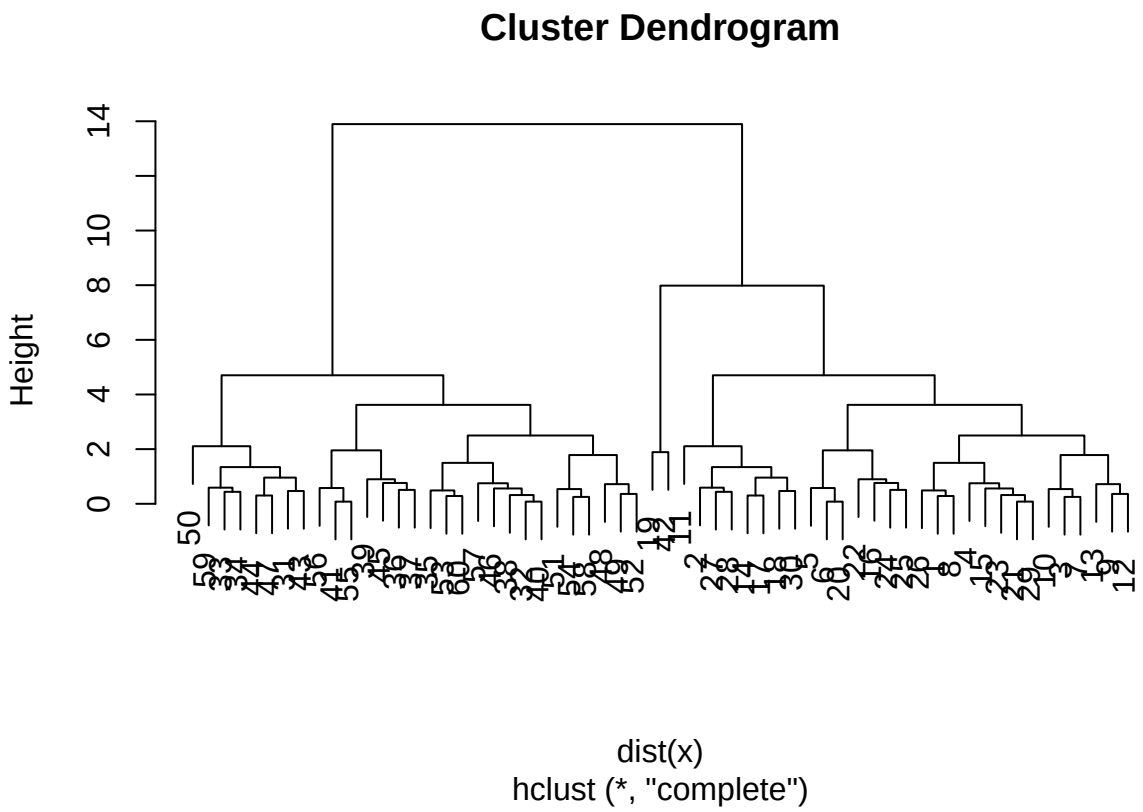
Demonstrate the use of dist(), hclust(), plot(), and cutree() functions to do clustering. Generate dendrograms and return cluster assignment/membership vector.

```
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. It plots a dendrogram.

```
plot(hc)
```



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

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

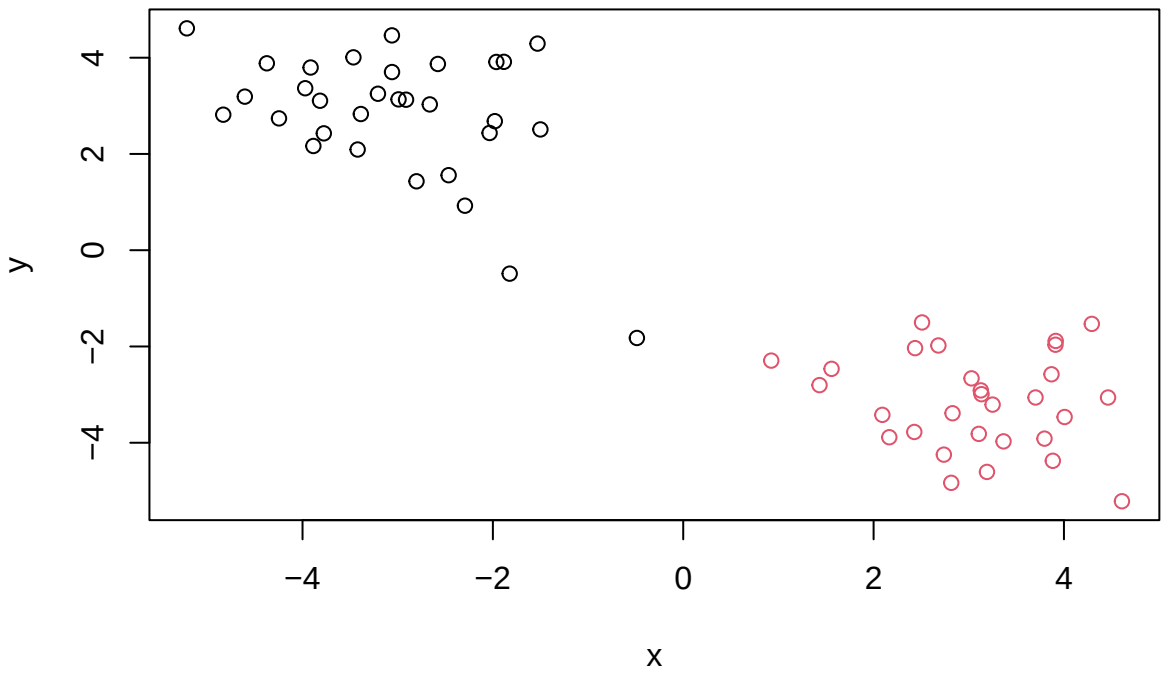
```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3
## [39] 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

You can also call `cutree()` setting k to the numbers of groups/clusters you want.

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

Make our results plot.

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



Principal Component Analysis of UK Food

Understanding the data frame

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

Q1. How many rows/columns are there in this new data frame?

```
dim(x)
```

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

Using dim(), we can determine there are 17 rows and 5 columns.

How would you preview the data frame.

```
#have to comment out View(), since this is an interactive function for R and will cause an issue when k
#View(x)
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
```

We actually have 4 columns of interest (not 5), and 17 rows.

Adjust the data with the new information.

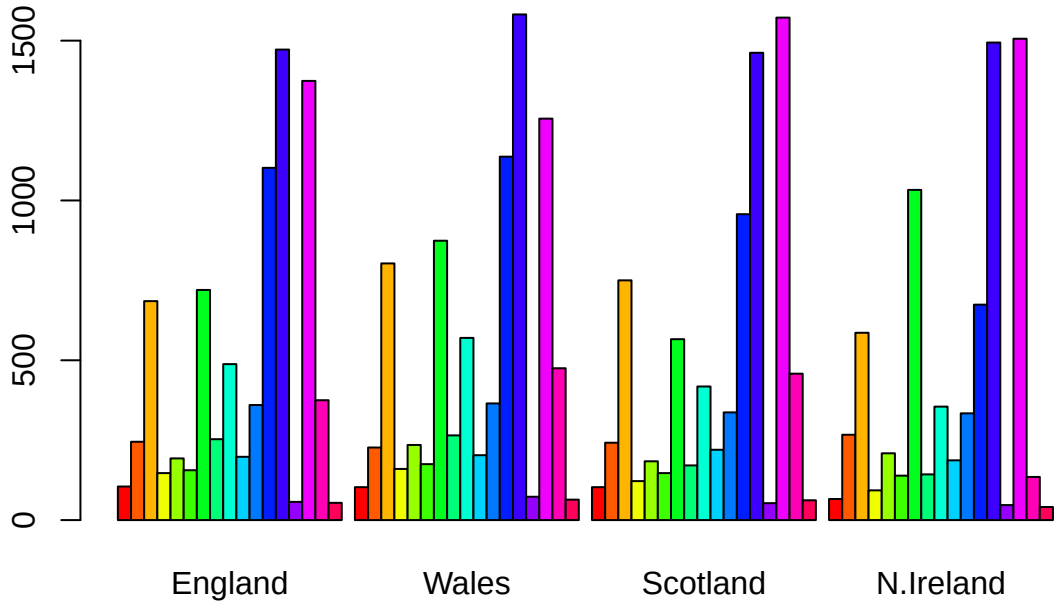
```
#currently rownames is just the numbering
#rownames(x)
#set rownames to info in column 1 (actual row names)

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

Q2.

This code is commented out since it is an unsafe way to make the changes we want to make. Every time this line of code runs, it will basically delete another column of x since we are directly setting x to x - 1 column. To avoid this, we can adjust the data when we read it in.

Plot the data to better understand it.

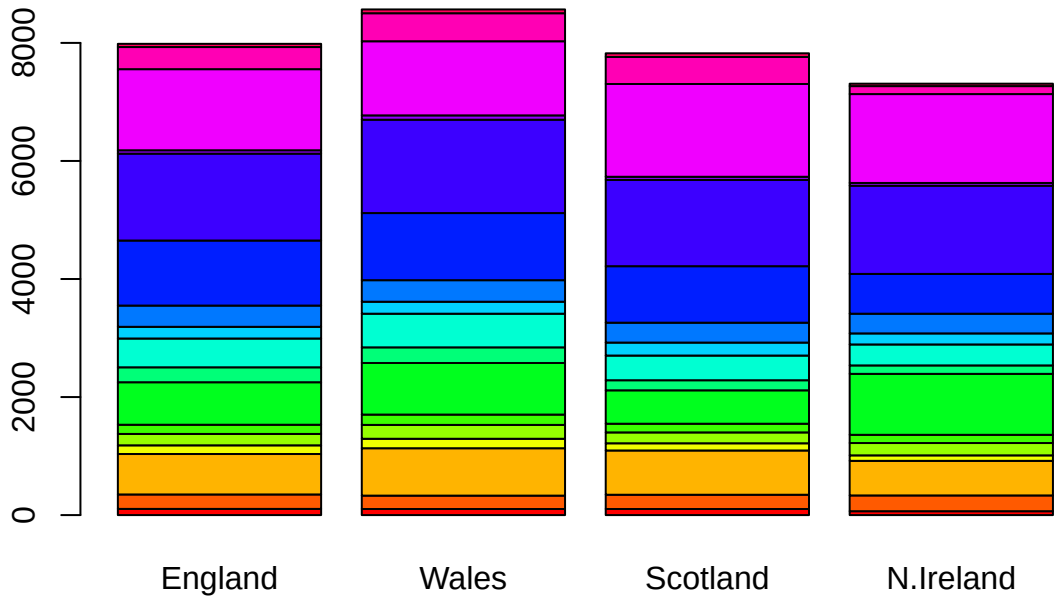
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

By setting beside = FALSE, we can change the barplot. function barplot() has beside set to the default of FALSE, so we can also do this just by leaving the arugment beside out.

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

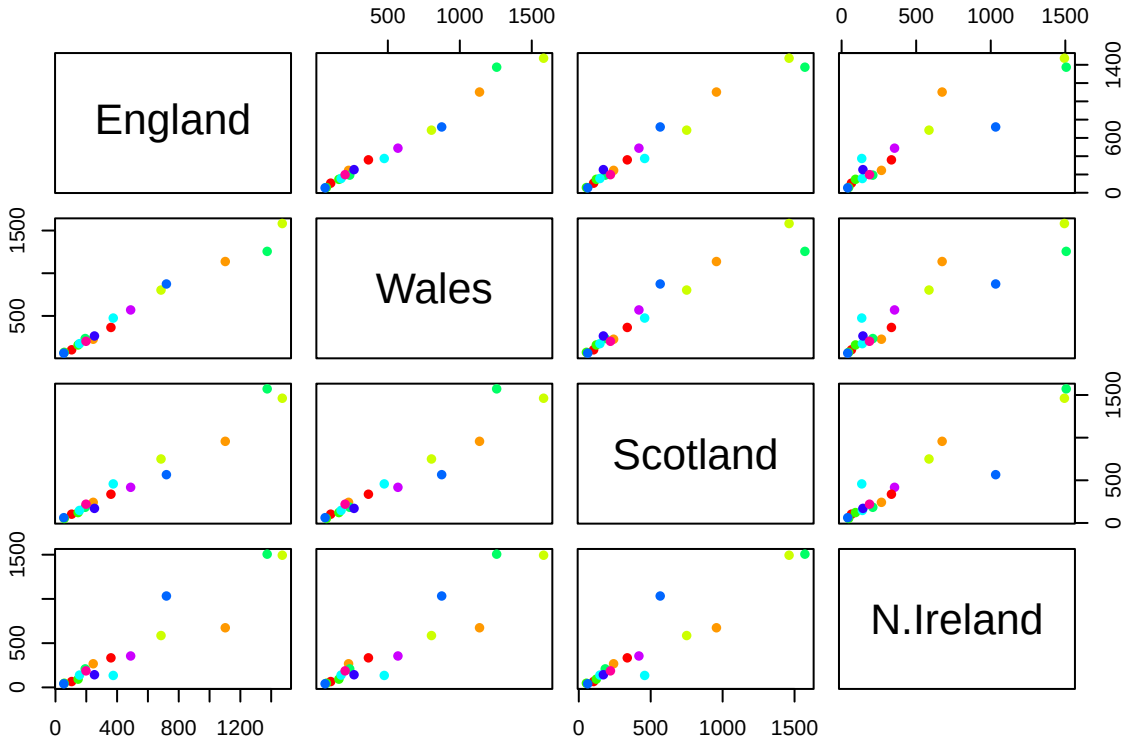


Q4. Missing in Handout

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?

Yes. If a given point lies on the diagonal then it means it is similar to the rest of the dataset, if it is not on the diagonal it indicated variation (dissimilar to the existing trend). Even with this analysis, it is difficult to understand the data well/in a quantifiable way.

```
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?

When N. Ireland is compared in a piecewise plot, there is a lot of variation from the ideal diagonal line. This is because, N. Ireland's data is dissimilar to the data from all of the other countries. Comparatively, we can determine that N. Ireland is the most dissimilar overall.

PCA to the rescue

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

```
#t() returns transpose of our data
t(x)
```

```
##             Cheese Carcass_meat  Other_meat  Fish Fats_and_oils  Sugars
## England        105           245         685  147            193    156
## Wales          103           227         803  160            235    175
## Scotland       103           242         750  122            184    147
## N.Ireland       66           267         586   93            209    139
##             Fresh_potatoes  Fresh_Veg  Other_Veg  Processed_potatoes
## England                720        253        488                 198
## Wales                  874        265        570                 203
## Scotland               566        171        418                 220
## N.Ireland             1033        143        355                 187
```

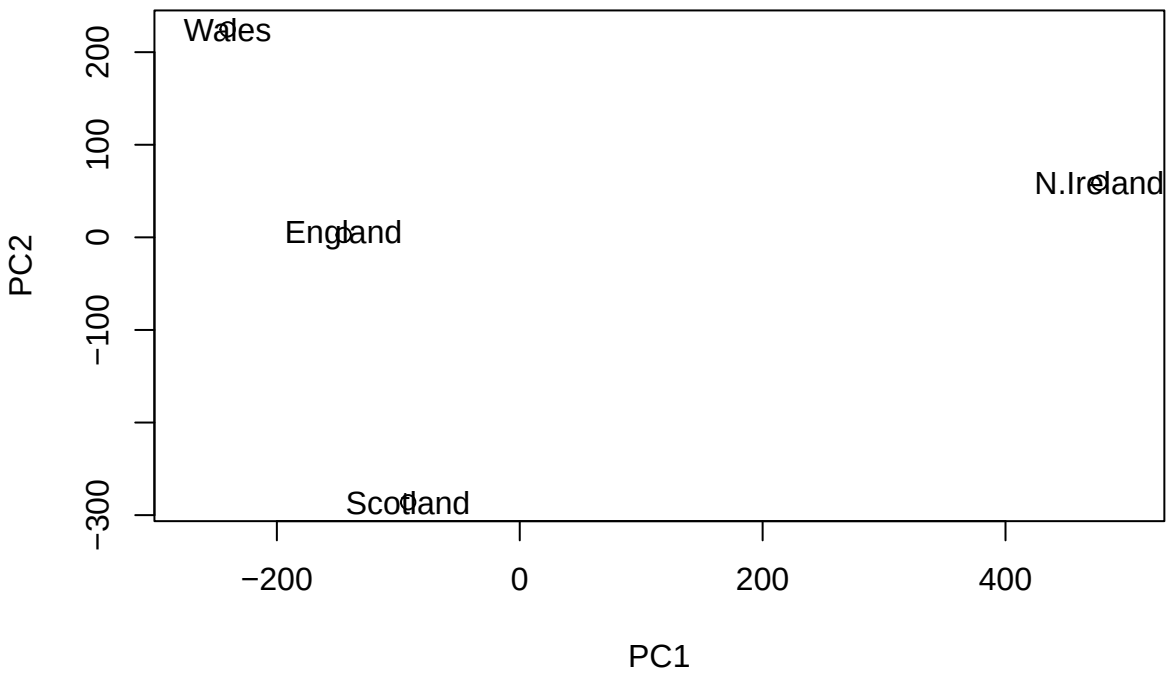
```
##           Processed_Veg  Fresh_fruit  Cereals  Beverages Soft_drinks
## England              360         1102     1472         57         1374
## Wales                365         1137     1582         73         1256
## Scotland             337          957     1462         53         1572
## N.Ireland            334          674     1494         47         1506
##           Alcoholic_drinks  Confectionery
## England                375             54
## Wales                  475             64
## Scotland               458             62
## N.Ireland              135             41
```

```
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
```

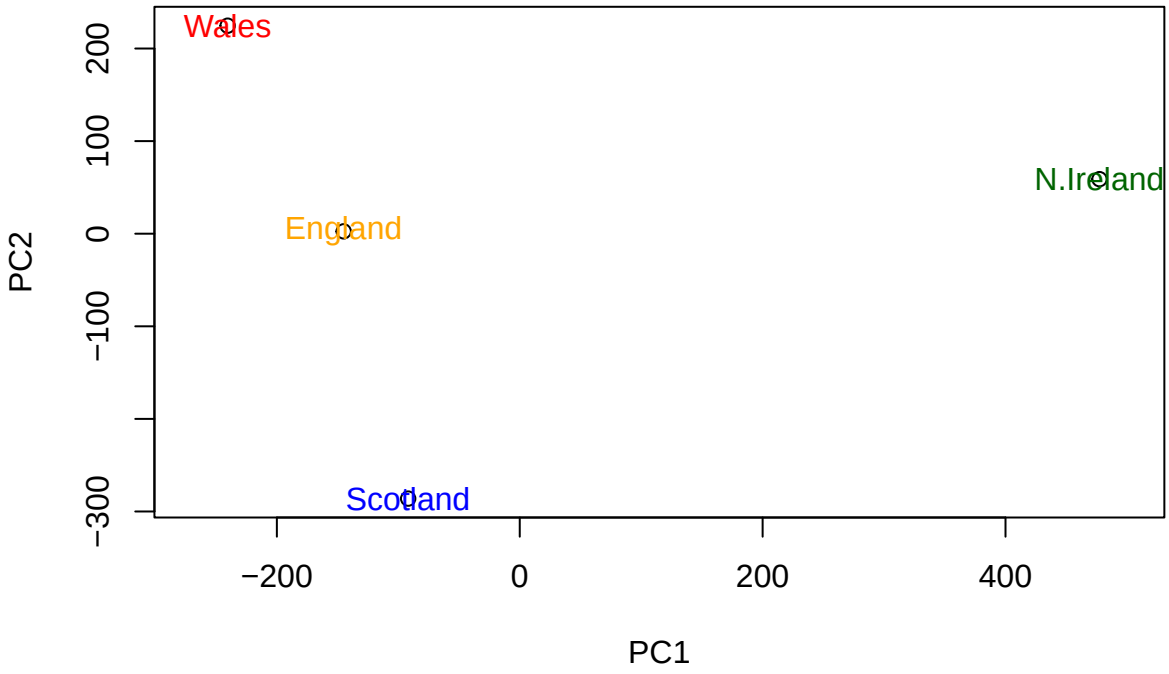
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
#pca$x[,1] -> pca1
plot(pca$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 start of this document.

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
color <- c("orange", "red", "blue", "dark green" )
text(pca$x[,1], pca$x[,2], colnames(x), col = color)
```



Calculate how much variation in the original data each PC accounts for.

Almost 97% of the variance in the data is accounted for by just PC1 and PC2. In the notes, it is mentioned that in practice you only need to include enough principal components to account for at least 70% of the variance.

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

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

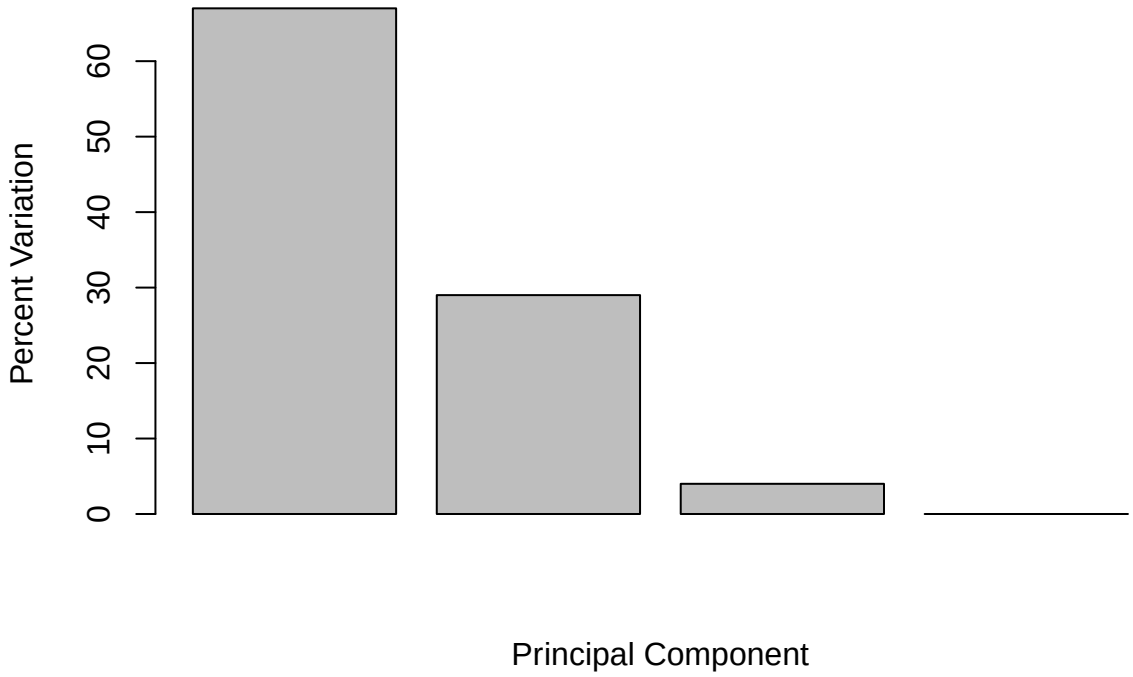
This information can also be found through this method.

```
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
```

We can also plot this information.

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

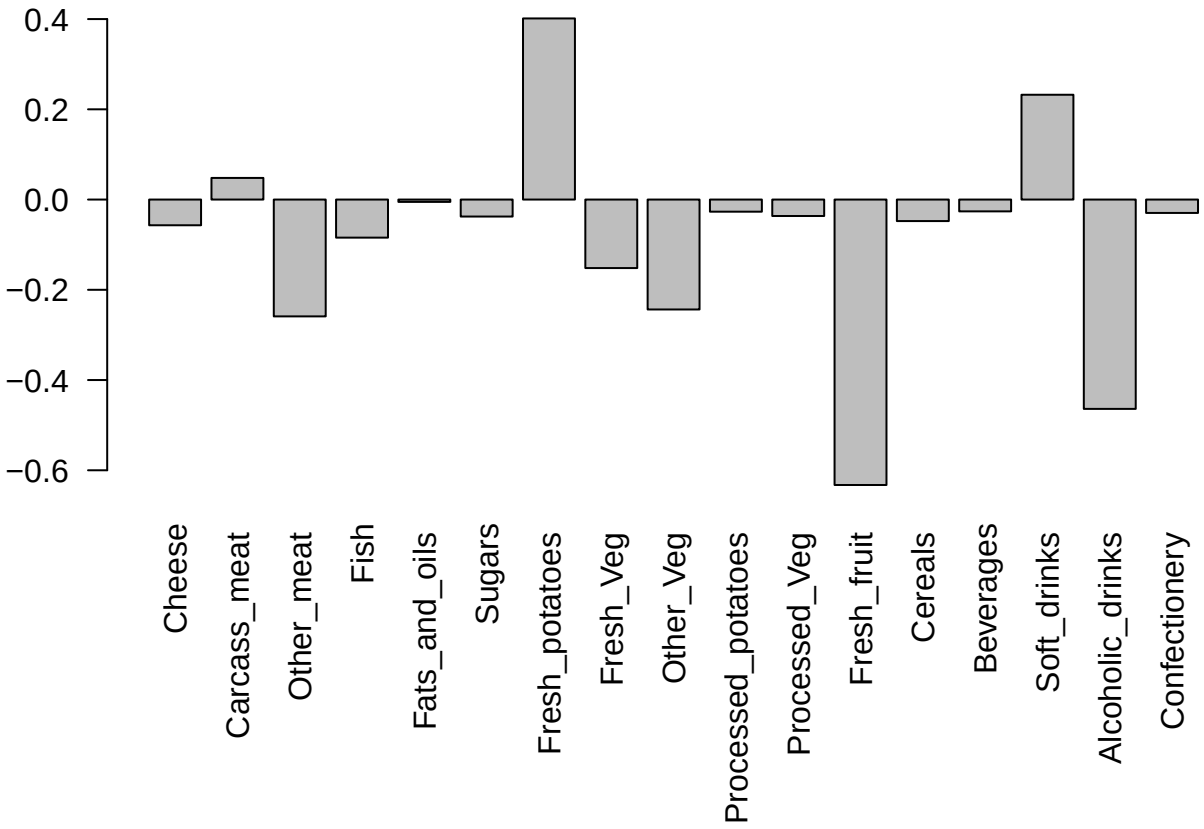


Variable Loadings

We can also determine the influence of each of the original variables (17 rows) upon the principal components (loading scores) using $rotation.

pca$rotation[,1] -> how much each variable contributes to pca1 (which accounts for most of the variation in data)

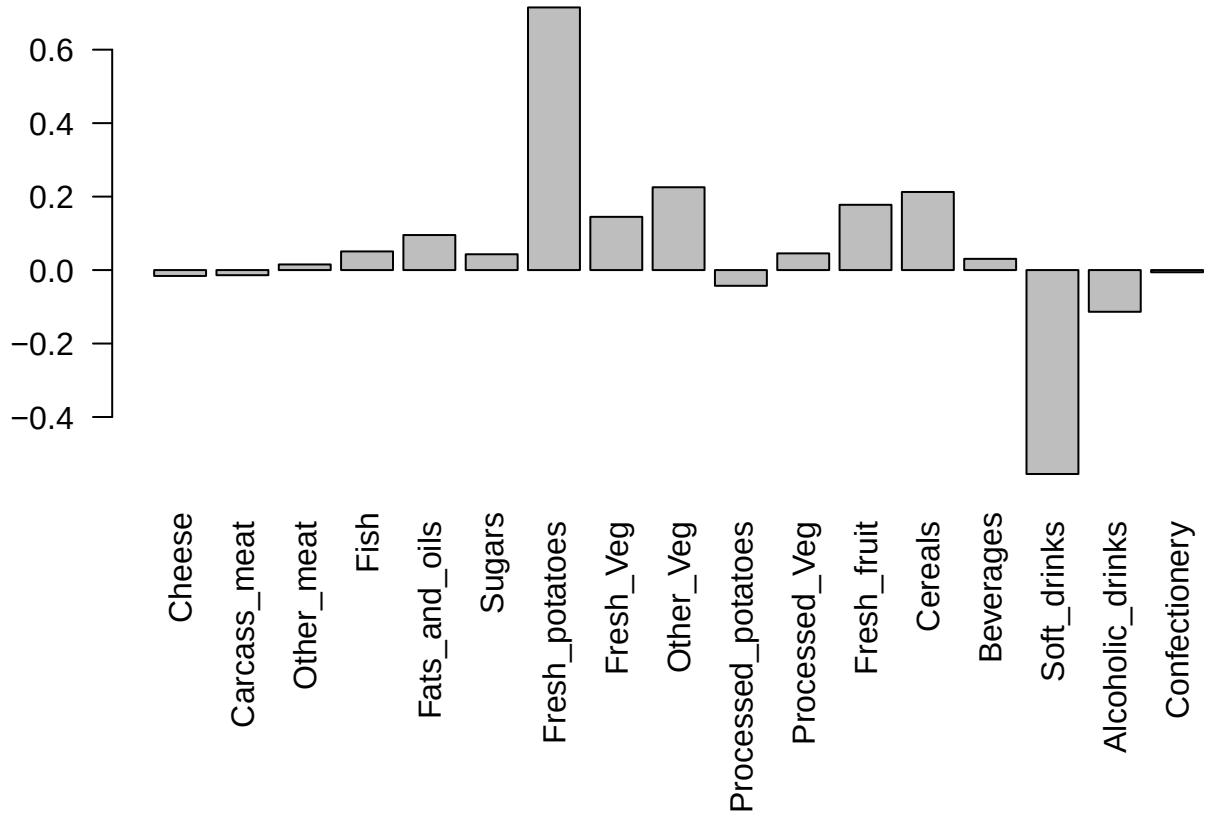
```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



Note: Largest positive loading scores push N.Ireland to the right. Largest negative loading scores push other countries to the left.

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

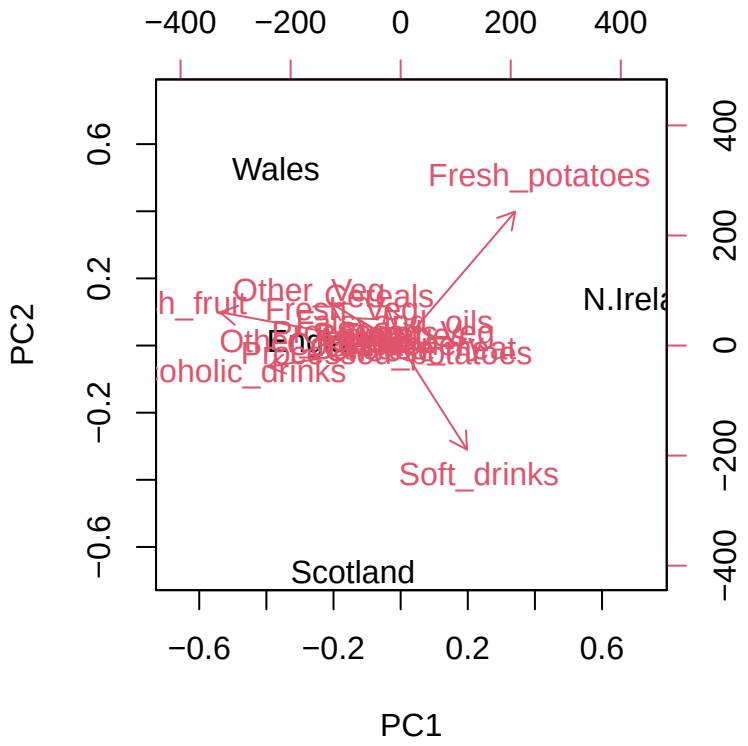
```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```



The two food groups are fresh potatoes and soft drinks. Fresh potatoes are the highest positive loading score and soft drinks are the highest negative loading score. PC2 accounts for less variance in the data than PC1 (only about 29% compared to the 67%). You can see this graphically due to the barplot's more similar distribution across (loading scores are closer to 0). Additionally, we also know that fresh potatoes and soft drinks are the categories that contribute most to PC2 (fresh potatoes push N.Ireland to the right of the plot while soft drinks push the other countries to the left of the plot).

We can also see this information with the biplot.

```
biplot(pca)
```



PCA of RNA-Seq Data

Load the 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?

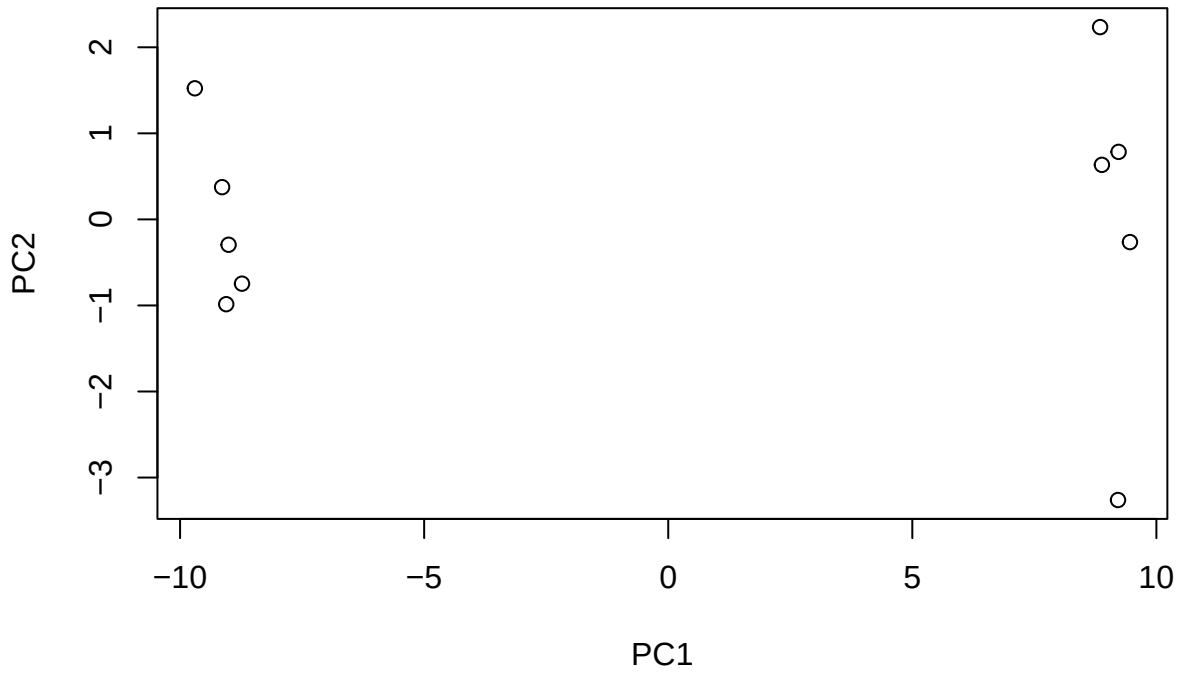
There are 100 genes and 10 samples.

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

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

PCA and plot

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

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




```
#how much variance does each pc account for
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
```

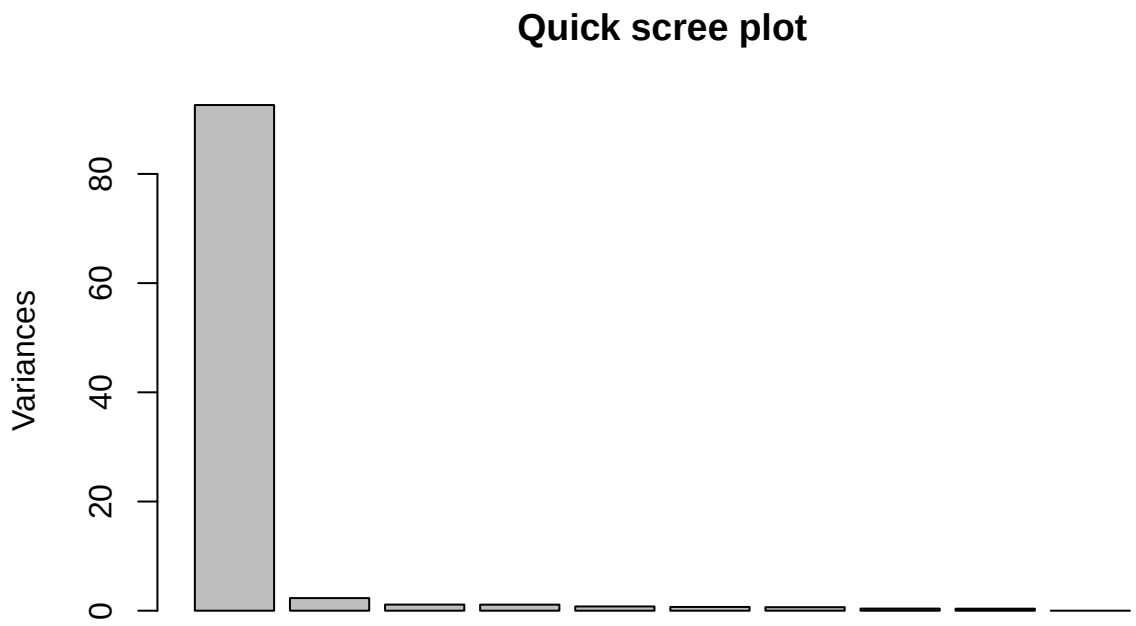
```
p <- round(pca$sdev^2/sum(pca$sdev^2)*100, 1)
p
```

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

PC1 accounts for about 93% of variance and PC2 accounts for about 2% of variance.

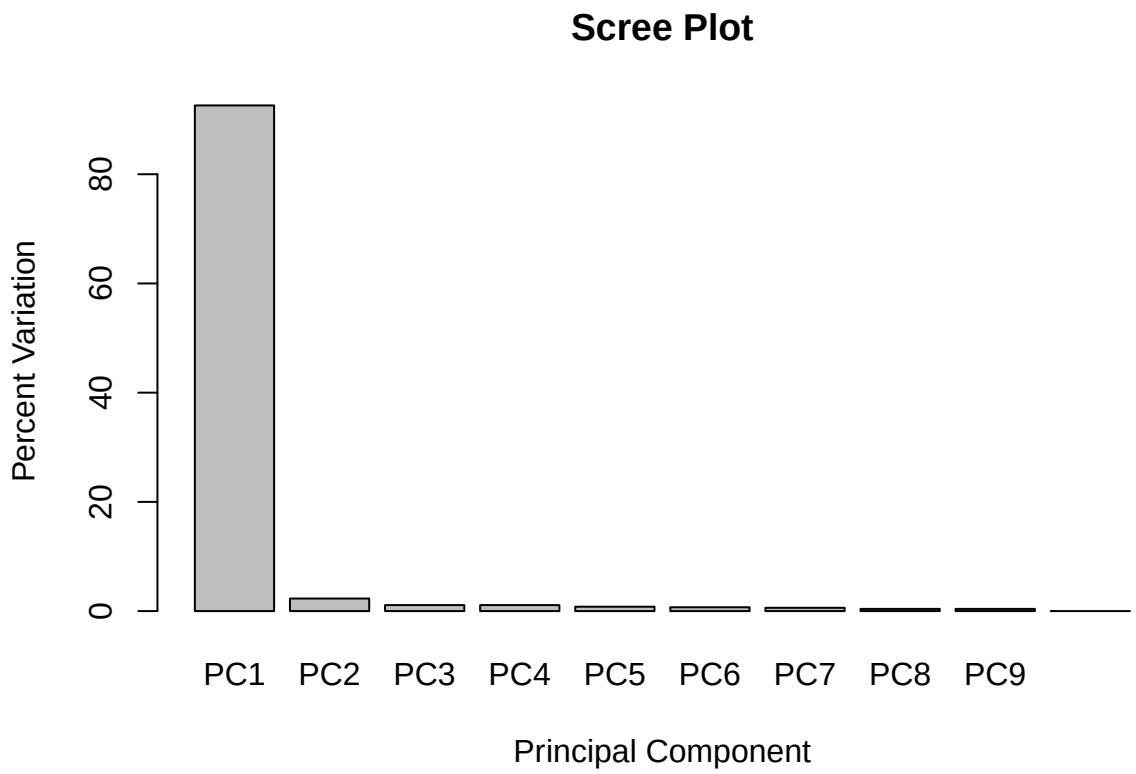
Create a barplot to represent proportion of variance accounted by each PC. We can do this using simply with plot(pca)

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



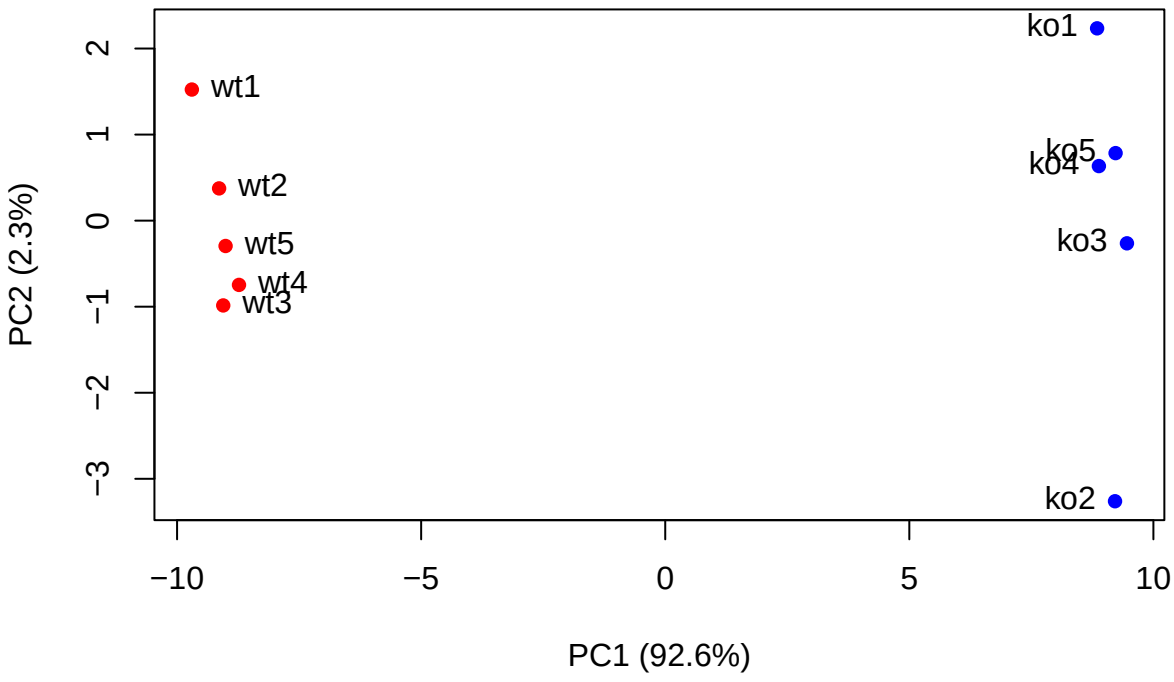
You can also manually plot this same graph using...

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



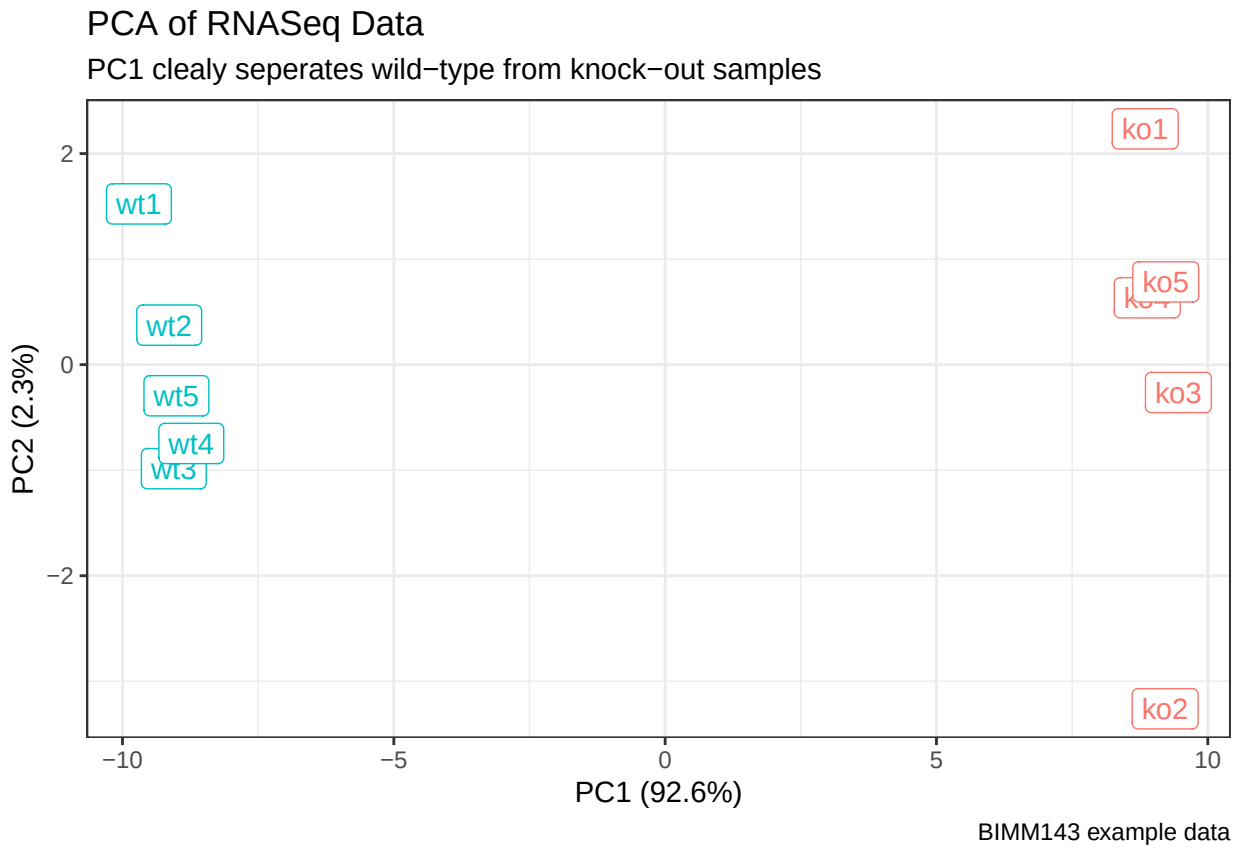
Let's update our main pca plot now

```
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 (", p[1], "%)"),
     ylab=paste0("PC2 (", p[2], "%"))

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



Plot using ggplot

```
library(ggplot2)

df <- as.data.frame(pca$x)
df$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) +
        labs(title="PCA of RNASeq Data", subtitle = "PC1 clealy seperates wild-type from knock-out sampl
        theme_bw()

p
```



Now let's determine the loading scores

```
loading_scores <- pca$rotation[,1]

## sort for top 10
gene_scores <- abs(loading_scores)
gene_score_ranked <- sort(gene_scores, decreasing=TRUE)

## show ttop 10
top_10_genes <- names(gene_score_ranked[1:10])
top_10_genes
```

```
##  [1] "gene100" "gene66"  "gene45"  "gene68"  "gene98"  "gene60"  "gene21"
##  [8] "gene56"  "gene10"  "gene90"
```