class08

Bangyan Hu (PID: A15540189)

10/21/2021

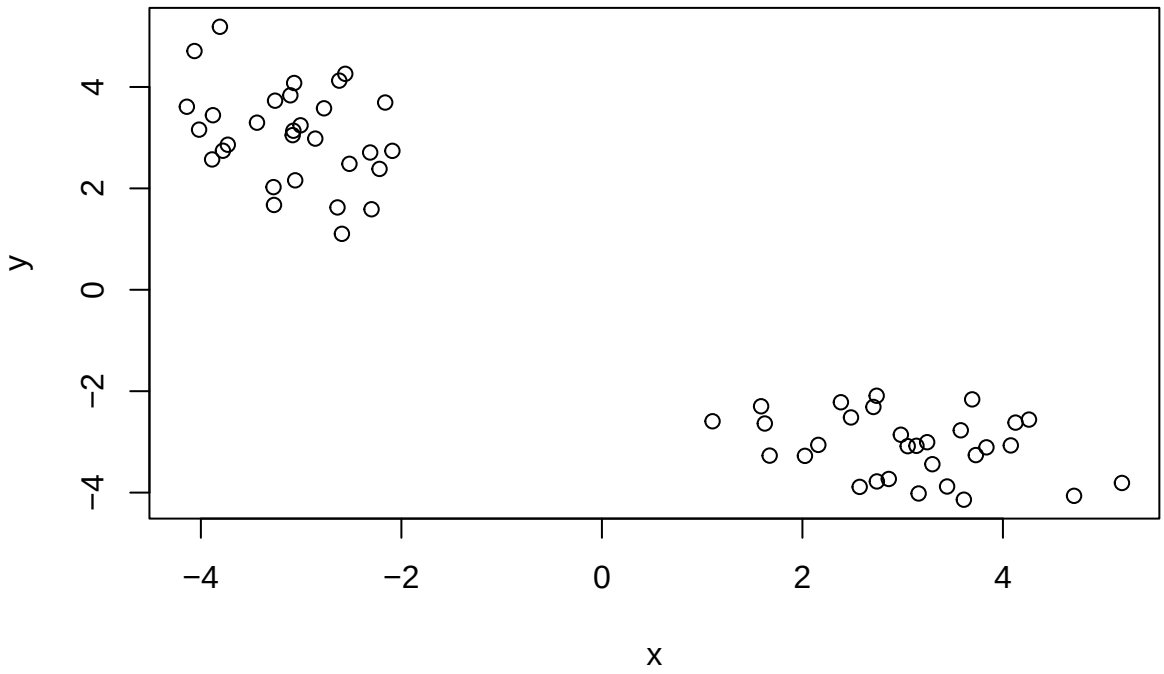
First up is clustering methods

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:

```
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 -3.086616  3.059440
## 2  3.059440 -3.086616
##
## 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
##
## Within cluster sum of squares by cluster:
## [1] 37.49149 37.49149
##  (between_SS / total_SS =  93.8 %)
##
## 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] 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
```

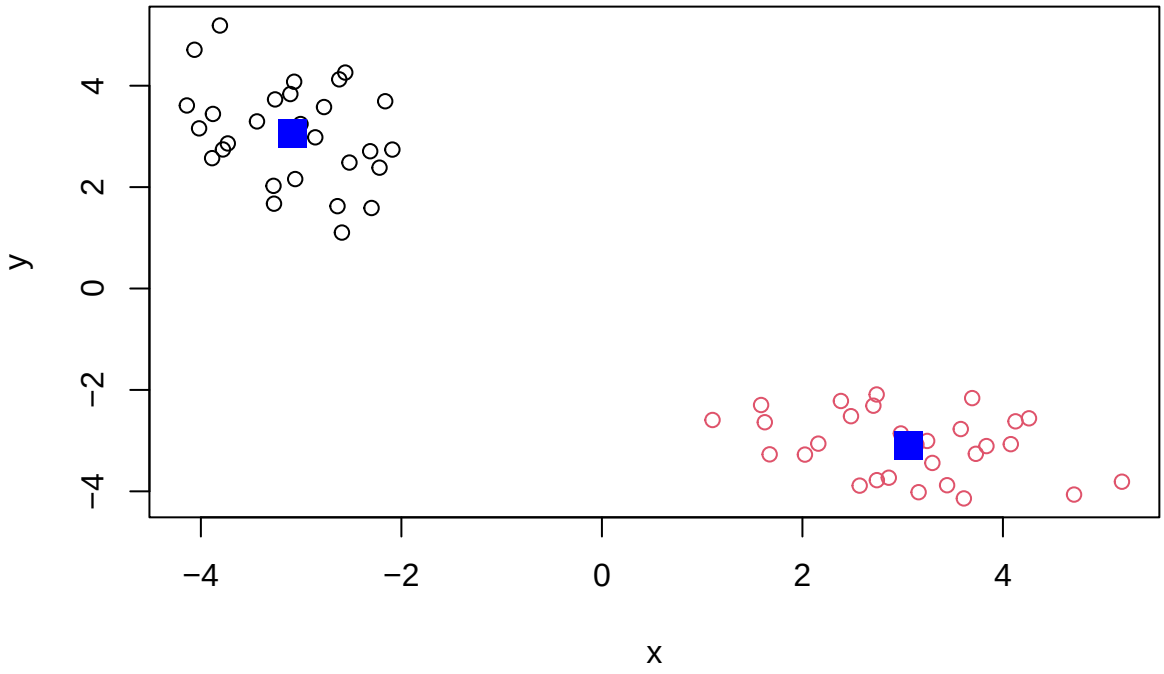
Q. What 'component' of your result object details cluster center?

```
km$centers
```

```
##           x         y
## 1 -3.086616  3.059440
## 2  3.059440 -3.086616
```

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()

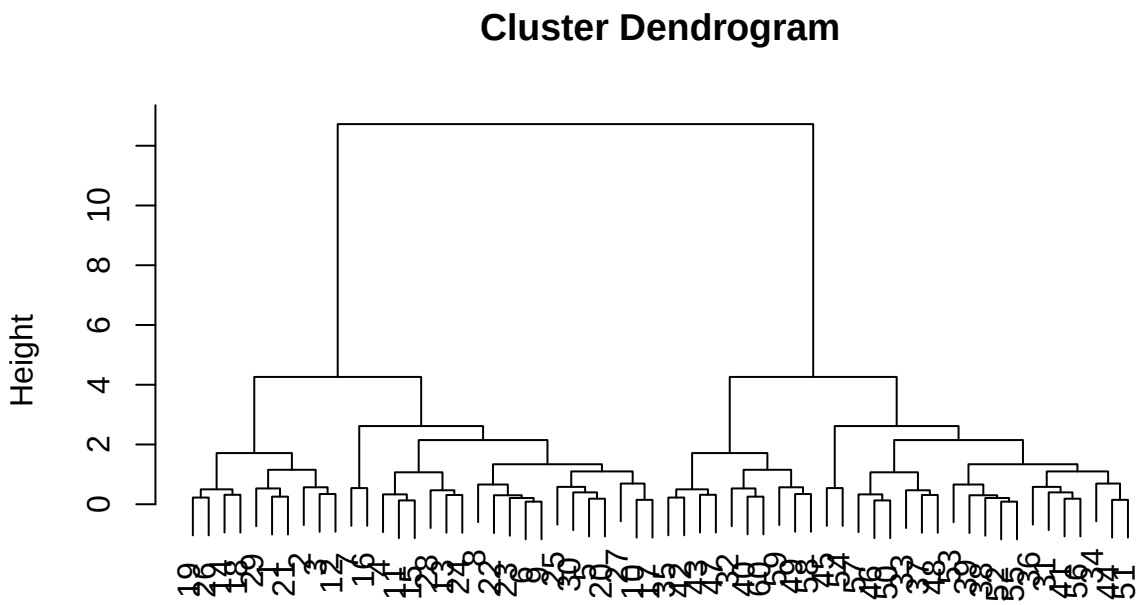
Demonstrate the use of dist(), hclust(), plot(), and cutree() functions to do clustering. Generate dendrograms and return cluster assignment/ membership vector...

```
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.

```
plot(hc)
```



To get our cluster membership vector we have to 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.

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

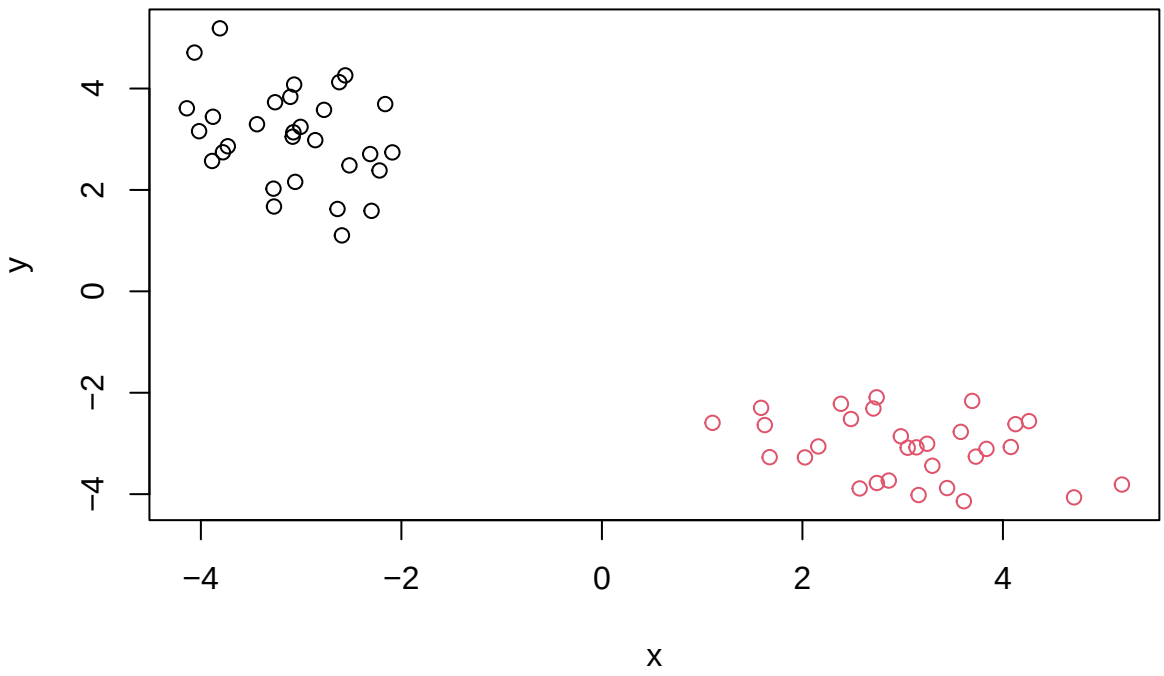
```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

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)
```



Principla Component Analysis

Class 8 Lab

First we will read the provided UK_foods.csv input file (note we can read this directly from the following tinyurl short link:

```
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
```

```
nrow(x)
```

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

```
ncol(x)
```

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

```
View(x)
```

17 rows and 5 columns.

However, this should be a 17 x 4 dimension (the row-names here were not set properly as we were expecting 4 columns (one for each of the 4 countries of the UK - not 5 as reported from the dim() function)).

one way

```
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
```

or

```
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
```

```
dim(x)
```

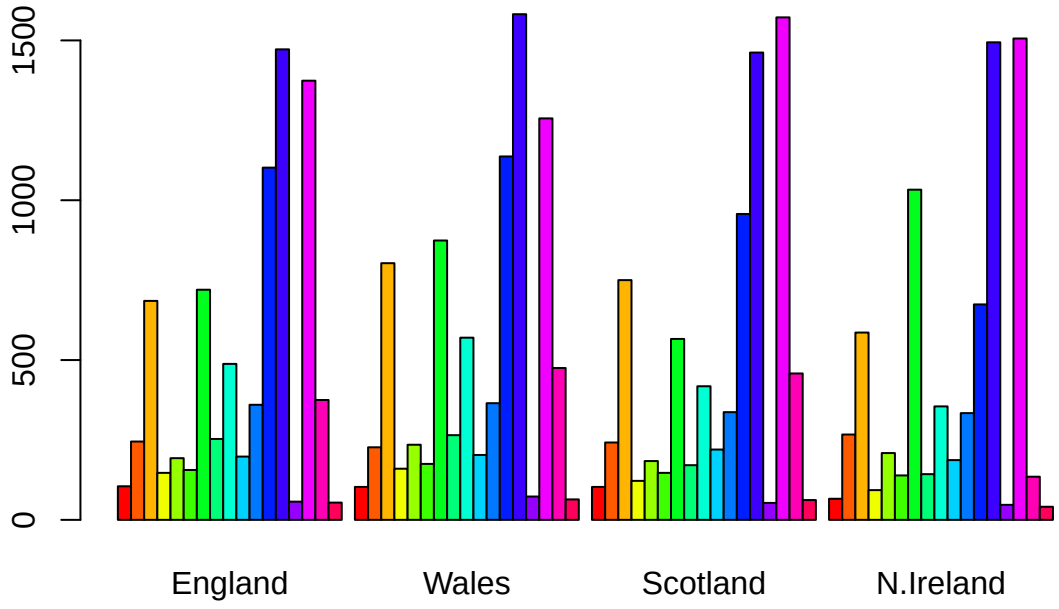
```
## [1] 17  4
```

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?

I do prefer the second method: x <- read.csv(url, row.names=1) head(x) This method assigns the first column to the row.names, which is more robust than the first method that simply removes the troublesome first column (with the -1 column index). The first method can only be applied once, and if it runs multiple times, it would remove other columns.

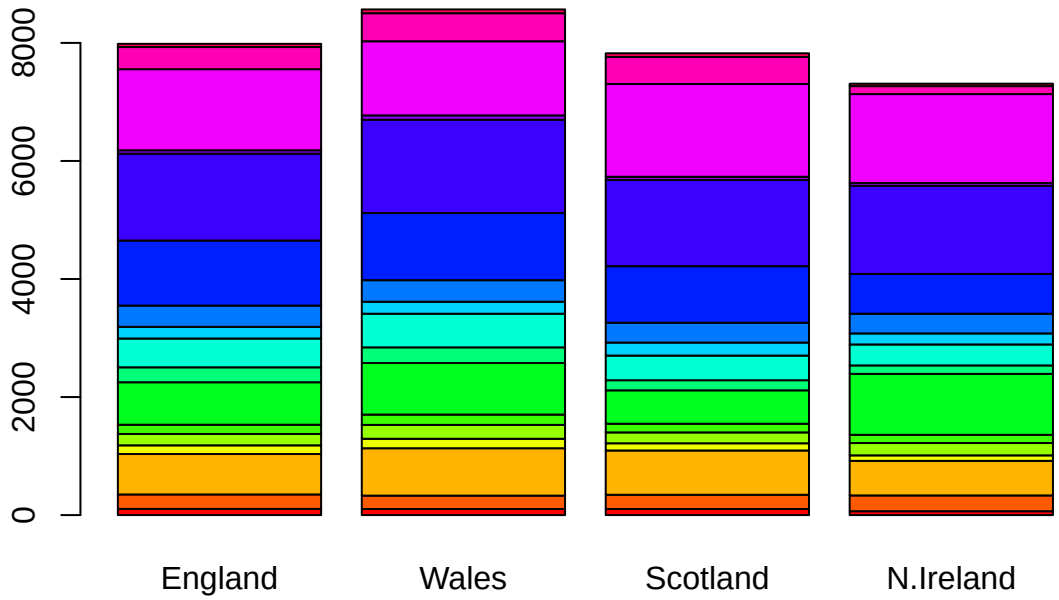
Q3 Changing what optional argument in the above barplot() function results in the following plot?

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



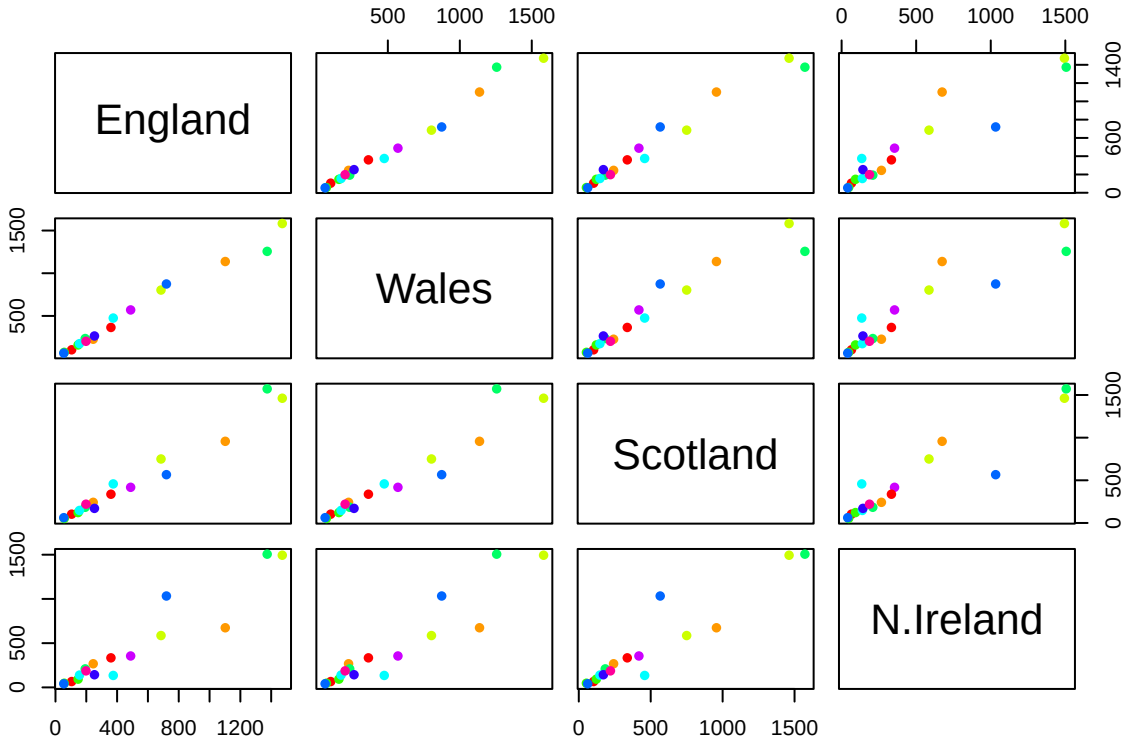
Changing beside=TRUE to beside=FALSE in the barplot() code results in the following plot.

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



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)
```



I can understand the following code and resulting figure. If a given point lies on the diagonal for a given plot, it means that the consumption in grams of a certain type of food-stuff is same at both countries

However, it might be hard to fully make sense of even this relatively small data set, so a powerful analytical method is absolutely necessary if we wish to observe trends and patterns in larger datasets – we need PCA!

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

Based on the figures above, N. Ireland seems to have much more differences in the consumption in grams of these 17 different types of food-stuff compared to other three countries of the United Kingdom in 1997 (the main differences: constitution and distribution of the consumption in grams of these 17 different types of food-stuff).

PCA to the rescue

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

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

```
## Importance of components:
##                             PC1      PC2      PC3       PC4
## Standard deviation     324.1502 212.7478 73.87622 5.552e-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)
```

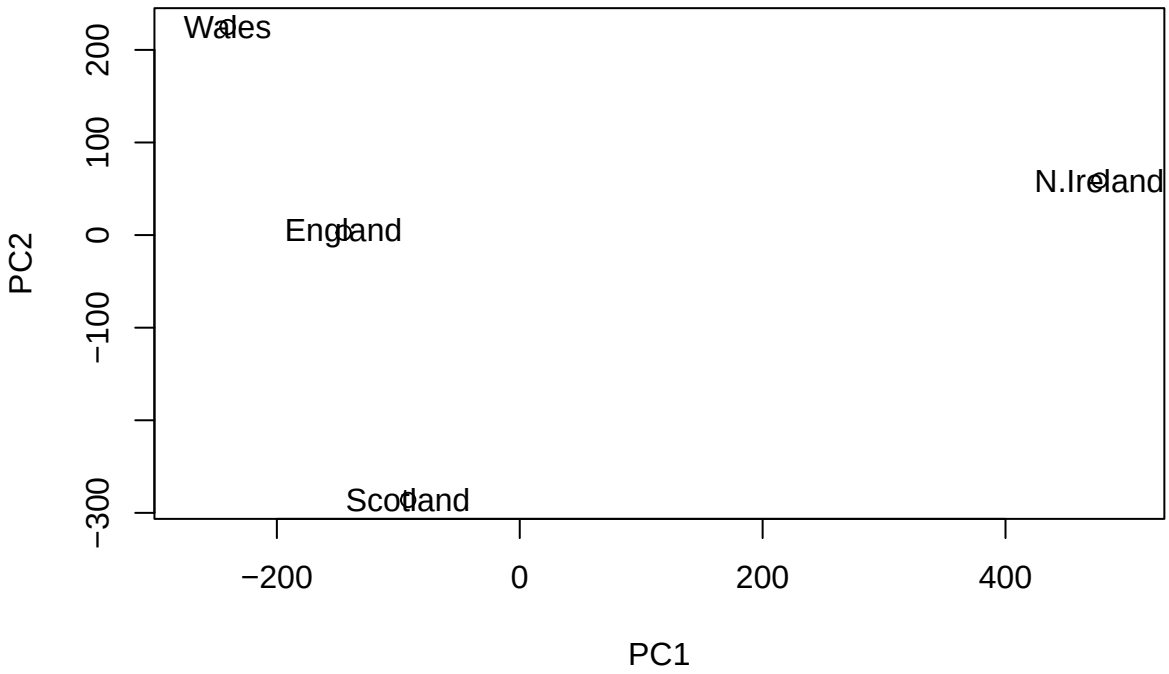
```
## $names
## [1] "sdev"     "rotation" "center"   "scale"    "x"
##
## $class
## [1] "prcomp"
```

```
pca$x[,1]
```

```
##    England      Wales   Scotland  N.Ireland
## -144.99315 -240.52915  -91.86934  477.39164
```

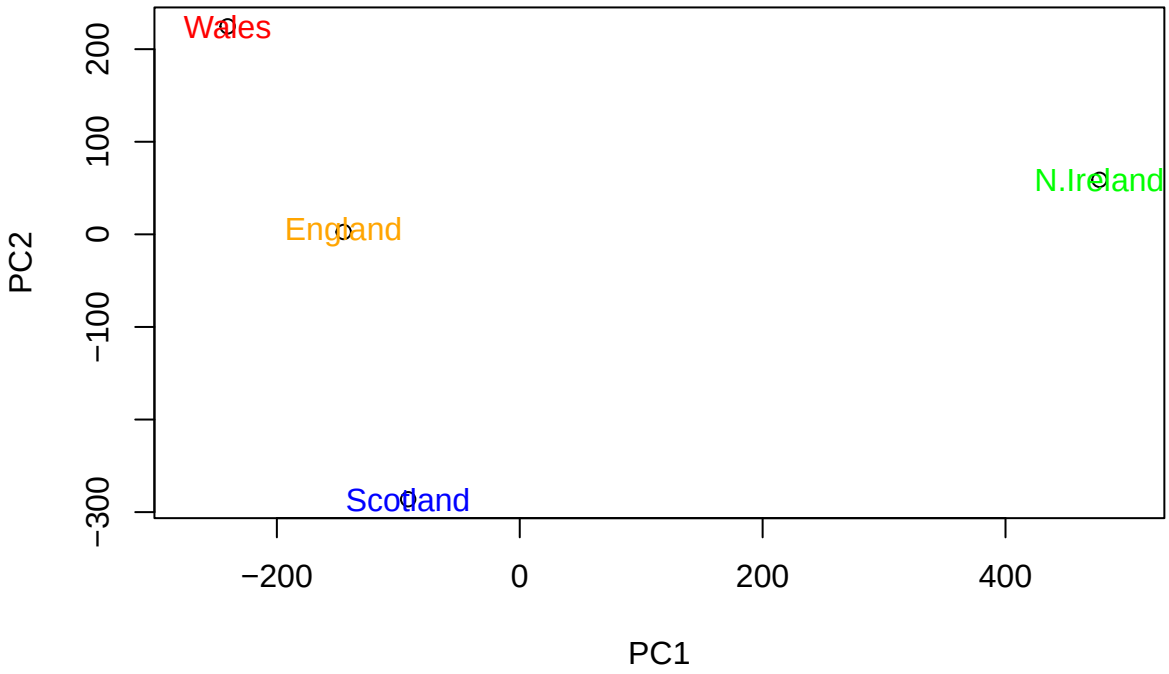
Q7 Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
# 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))
```



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 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"))
```



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

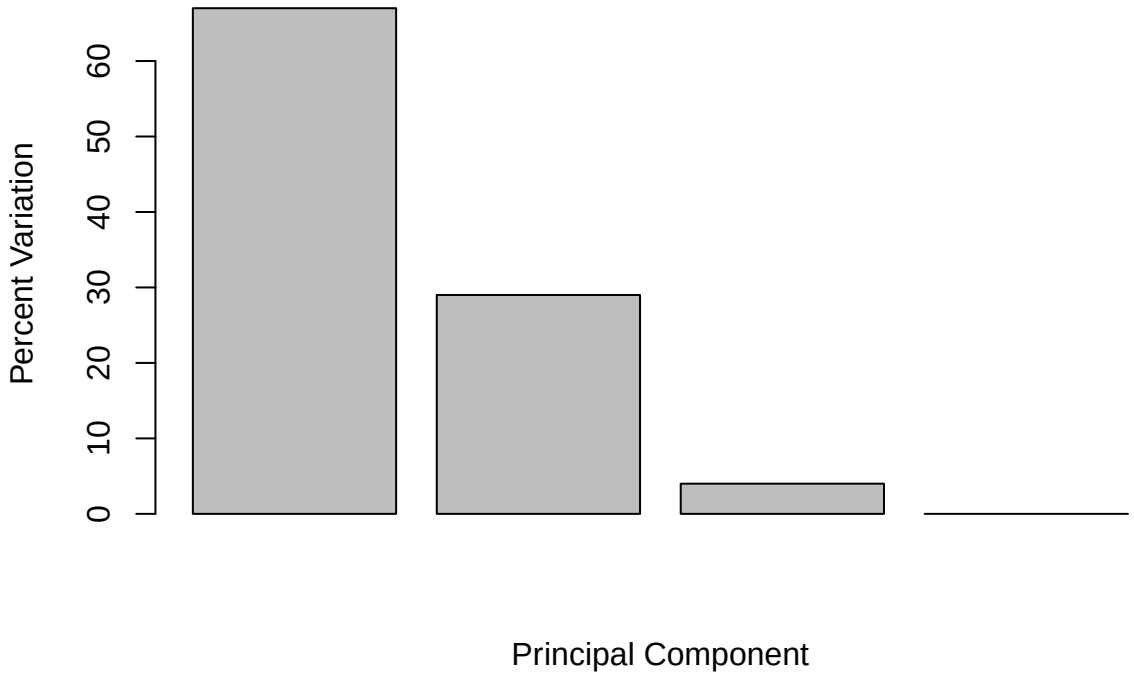
```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

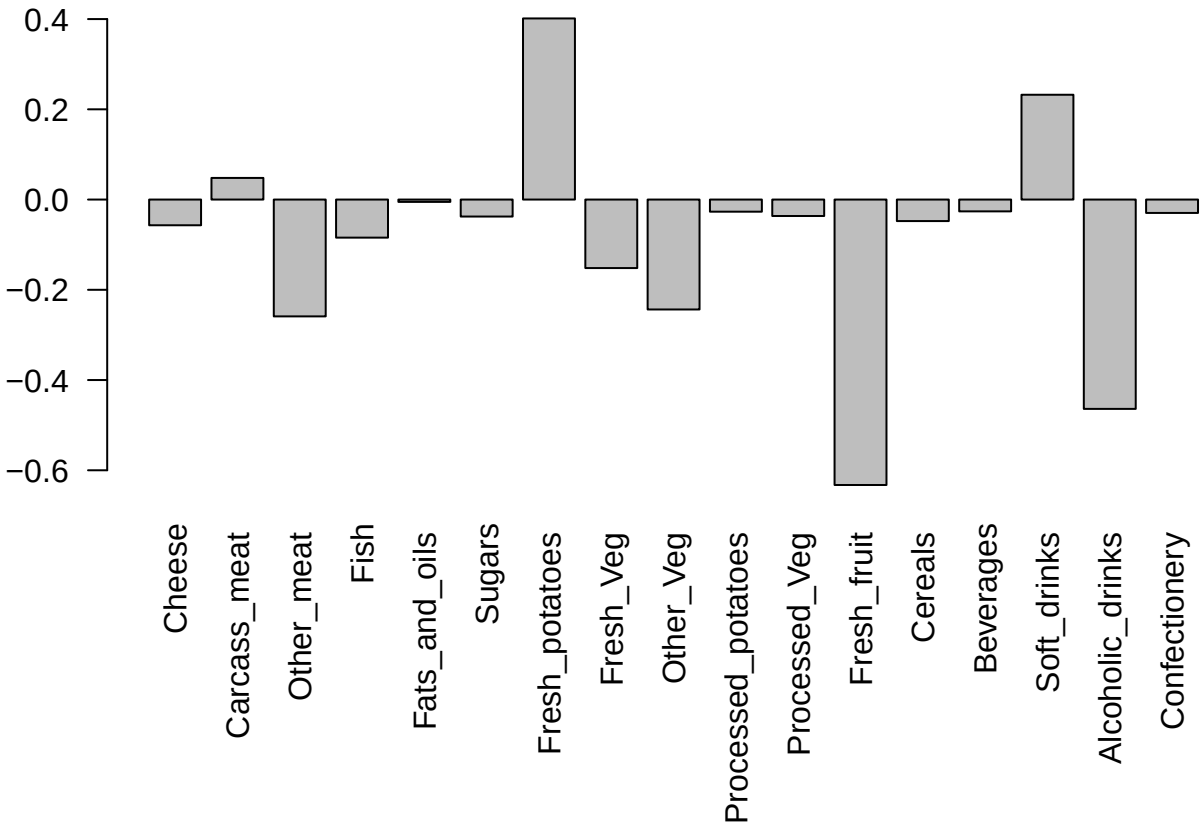
```
z <- summary(pca)
z$importance
```

```
##                             PC1       PC2      PC3          PC4
## Standard deviation     324.15019 212.74780 73.87622 5.551558e-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")
```




```
## Lets firstly focus on PC1.
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```

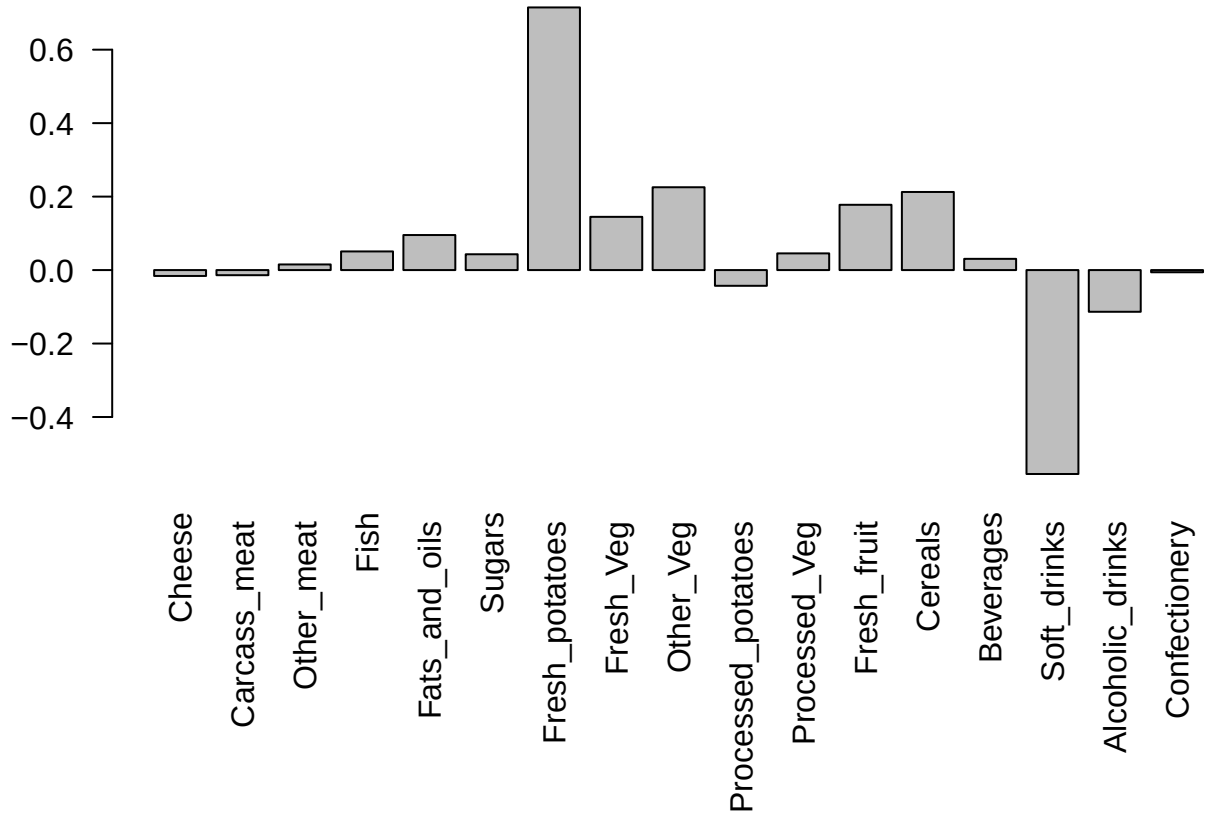


Here we see observations (foods) with the largest positive loading scores that effectively "push" N. Ireland to right positive side of the plot (including Fresh_potatoes and Soft_drinks).

We can also see the observations/foods with high negative scores that push the other countries to the left side of the plot (including Fresh_fruit and Alcoholic_drinks).

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

```
## Generate a similar 'loadings plot' for PC2.
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```



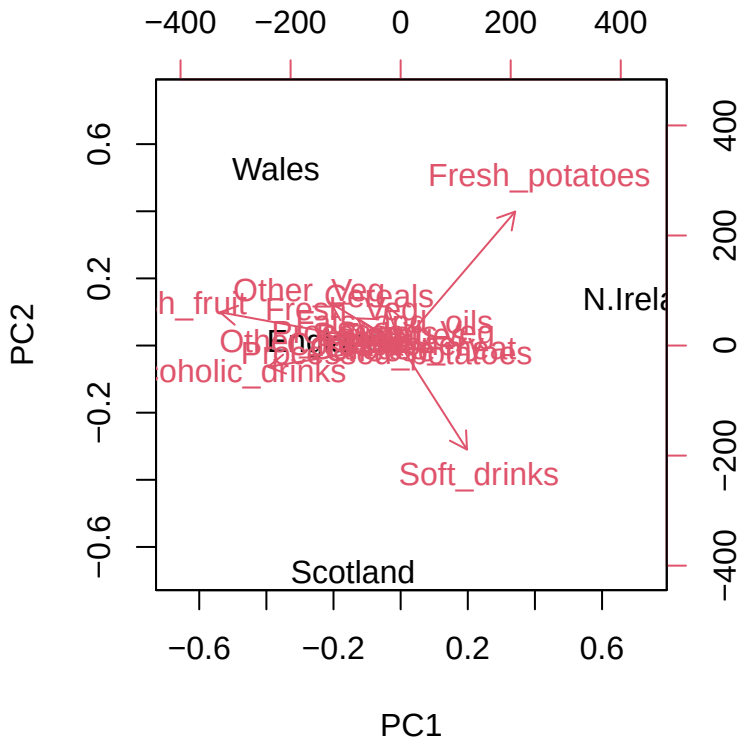
From the "loadings plot" for PC2, we see observations (foods) with the largest positive loading scores that effectively "push" Wales to the positive top of the plot (including Fresh_potatoes).

We can also see the observations/foods with high negative scores that push the other countries to the middle/bottom of the plot (including Soft_drinks).

Thus, two food groups feature prominantely are Fresh_potatoes and Soft_drinks. PC2 accounts for 29% of the sample variance, which can give us important information about variance or constitution/distribution of the data set for view and further investigation, but it demonstrates relatively less sample variance compared to PC1.

Biplots

```
## 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
```

Q10How many genes and samples are in this data set?

```
#NOTE: The samples are columns, and the genes are rows!
dim(rna.data)
```

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

```
nrow(rna.data) #rows
```

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

```
ncol(rna.data) #columns
```

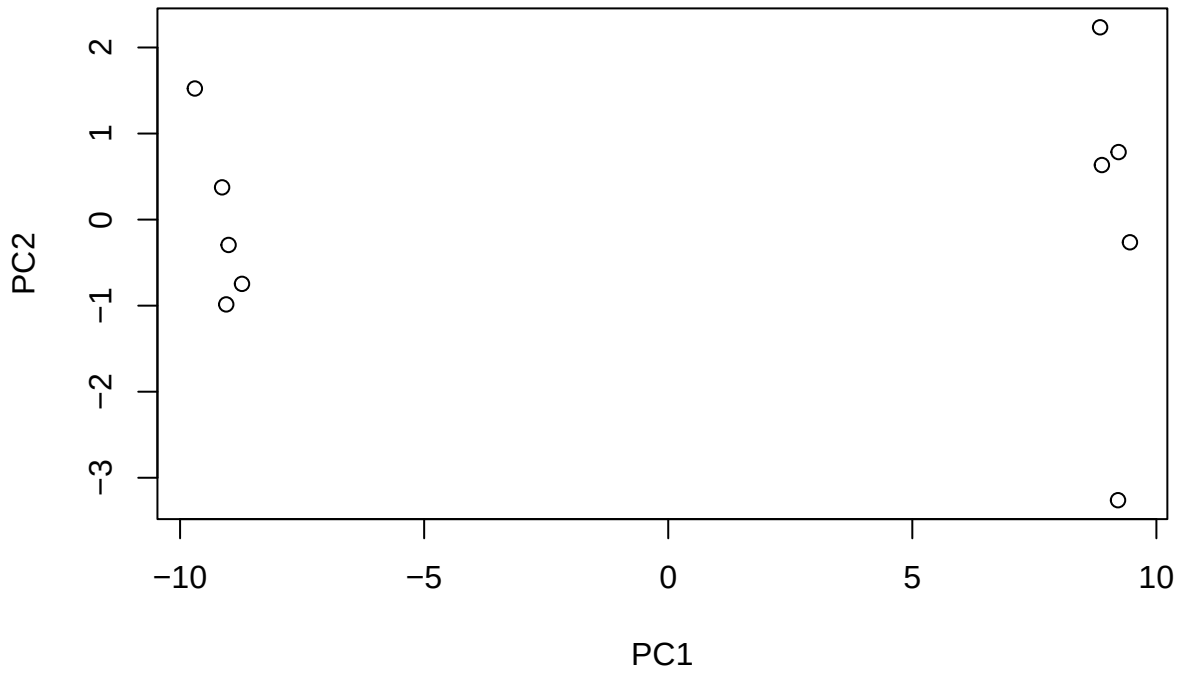
```
## [1] 10
```

```
View(rna.data)
```

Thus, there are 100 genes and 10 samples in this data set.

#PCA and plot the results of data set:

```
## Again we have to take the transpose of our data
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")
```



Examine a summary of how much variation in the original data each PC accounts for:

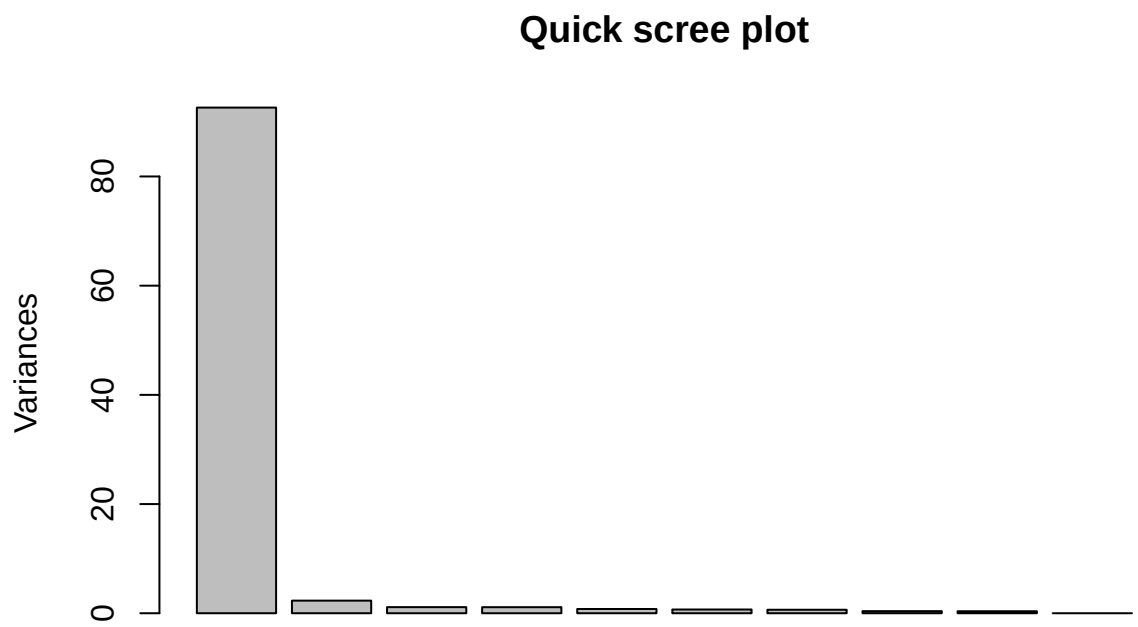
```
summary(pca)
```

```
## 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 3.327e-15
## Proportion of Variance 0.00385 0.00364 0.000e+00
## Cumulative Proportion  0.99636 1.00000 1.000e+00
```

A quick barplot summary of this Proportion of Variance for each PC

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

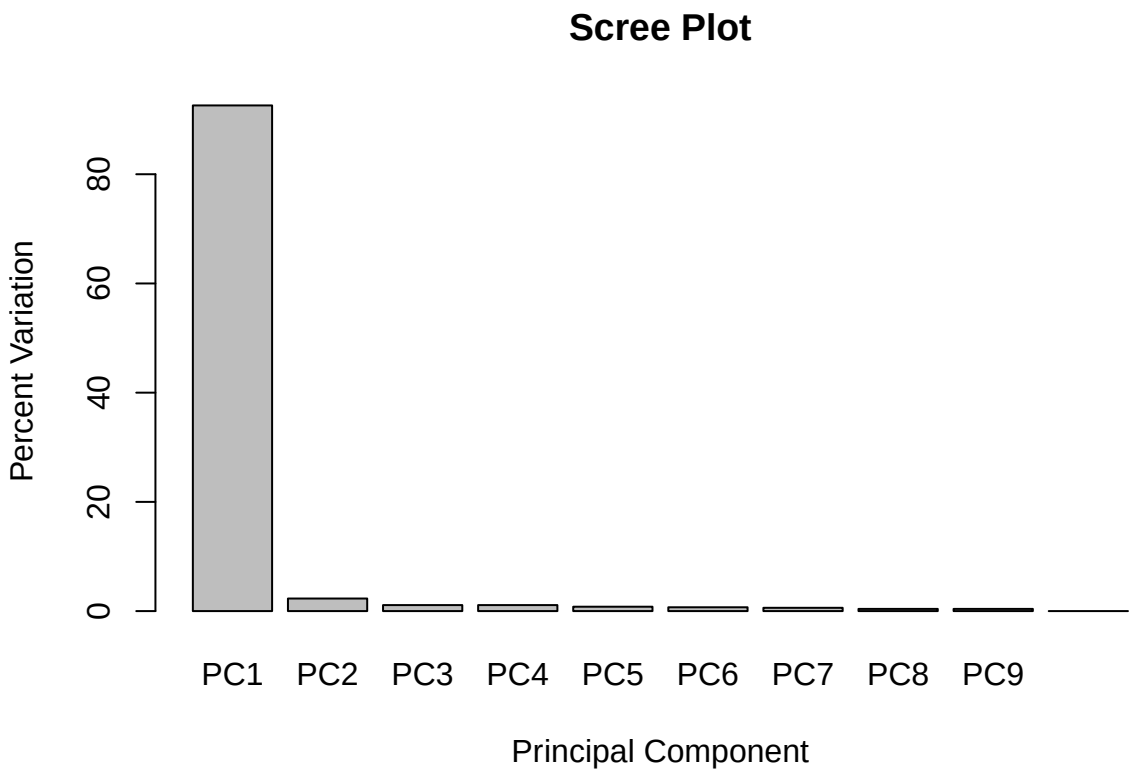



```
## Variance captured per PC
pca.var <- pca$sdev^2

## 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
```

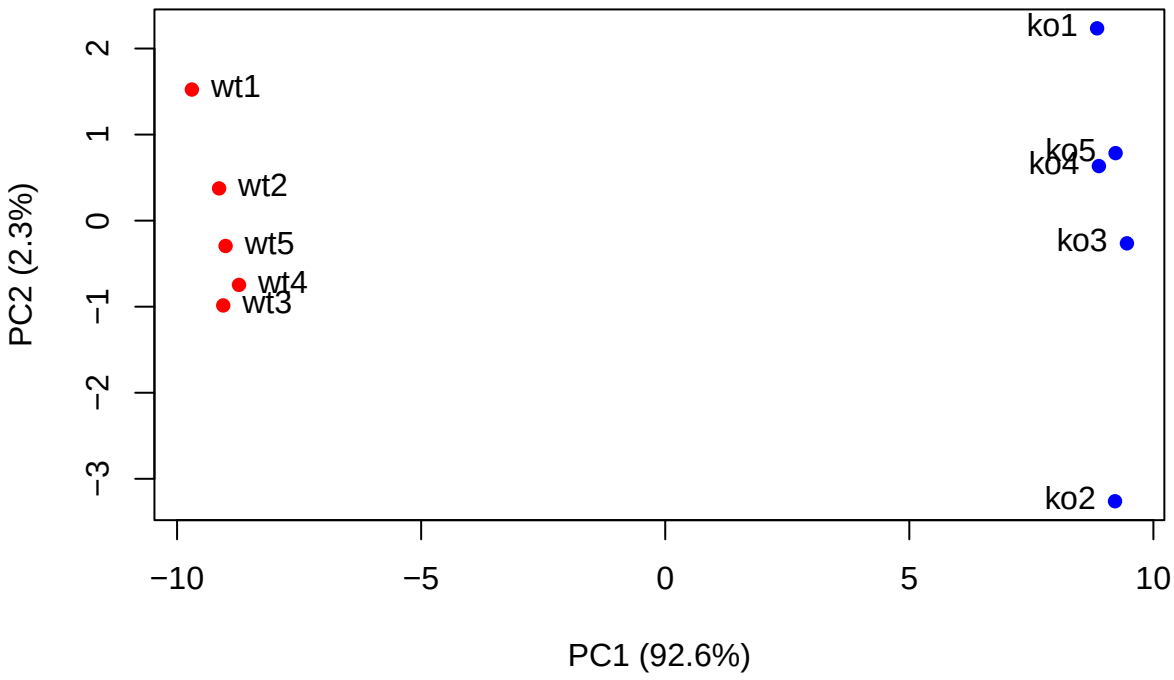
Generate the scree-plot:

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



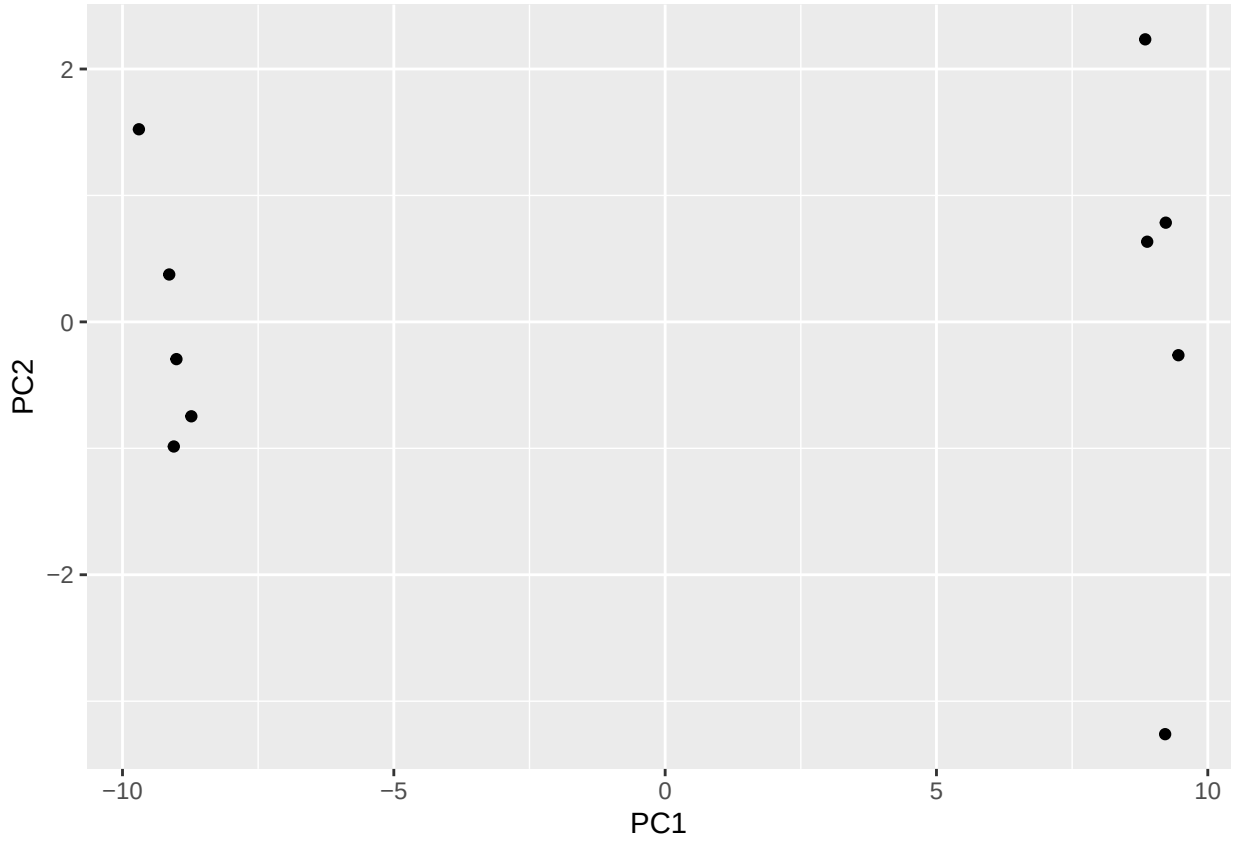
Customize the plot to be more intelligible.

```
## 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)))
```



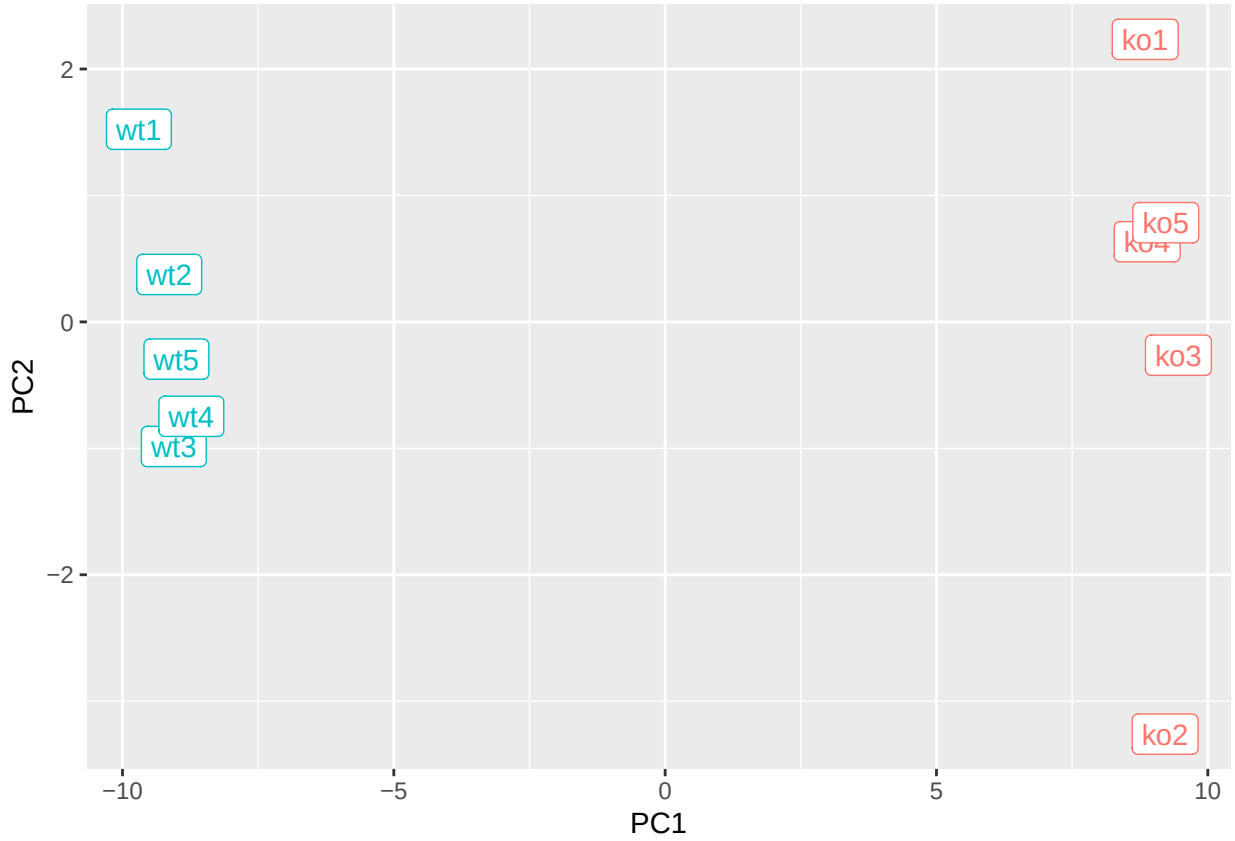
Then, using ggplot to visualize it.

```
library(ggplot2)

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

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



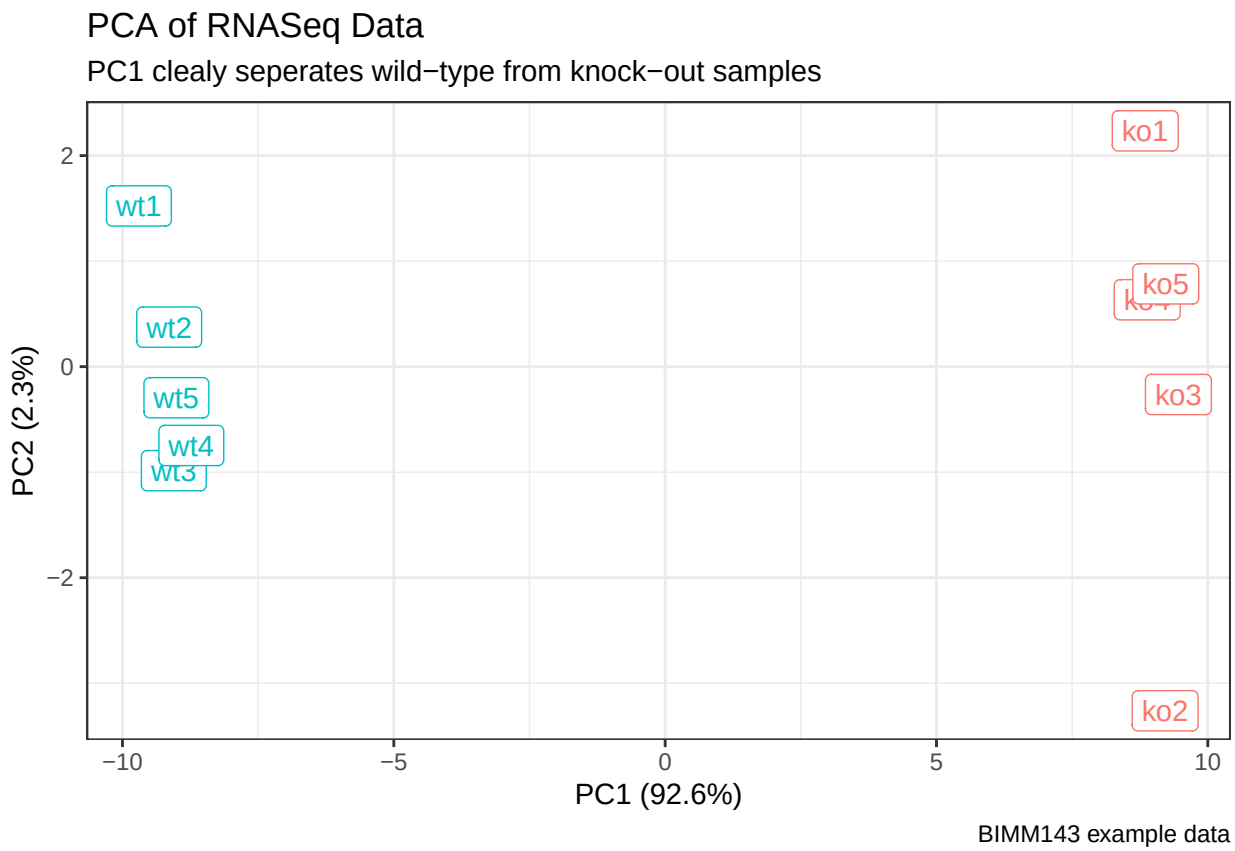
Customize the plot.

```
# 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
```



Moreover, 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()
```



#Optional: Gene loadings

Find the top 10 measurements contributing most to pc1 in either direction (+ or -).

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

## Find the top 10 measurements (genes) that contribute
## most to PC1 in either direction (+ or -)
gene_scores <- abs(loading_scores)
gene_score_ranked <- sort(gene_scores, decreasing=TRUE)

## show the names of the top 10 genes
top_10_genes <- names(gene_score_ranked[1:10])
top_10_genes
```

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

As their expression changes are statistically significant, these may be the genes that we would like to focus on for further analysis.