Lab 07: Machine learning I

Xiaoyan Wang(A16454055)

Today we are going to learn how to apply different machine learning methods, beginning with clustering:

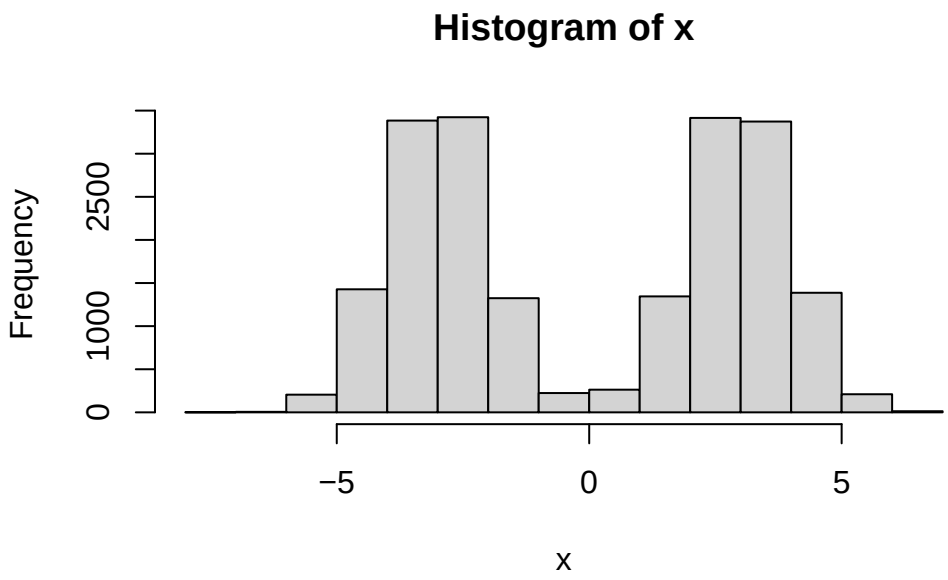
The goal here is to find groups/clusters in your input data.

First I will make up some data with clear groups. For this I will use the "rnorm()" function.

```
# generate 10 random numbers that are normally distributed
rnorm(10)
```

```
 [1] -0.808573254 -0.933968882  1.498375522 -0.559020188 -1.450691387
 [6] -1.418729059  0.131587008 -0.206328337  0.005700255  0.535807605
```

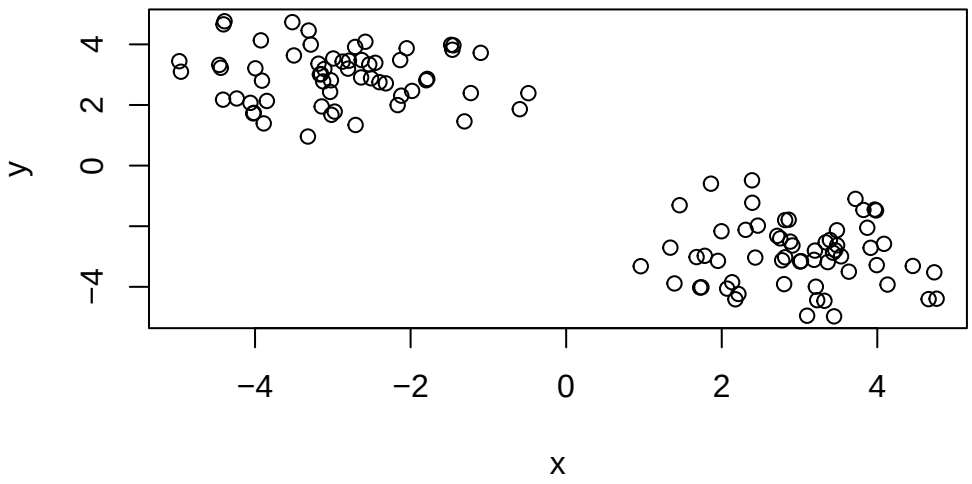
```
# generate a simple histogram with 2 groups of 10000 random numbers that center on 3 and -3
n<- 10000
x <- c(rnorm(n, mean=3), rnorm(n, mean=-3))
hist(x)
```




```
#
n<- 60
x<- c(rnorm(n, mean=-3),rnorm(n, mean=3))
y<-rev(x)

z<- cbind(x, y)
head(z)
```

```
             x        y
[1,] -3.500231 3.633931
[2,] -3.283349 3.992774
[3,] -1.226149 2.392789
[4,] -4.015897 1.739202
[5,] -3.184276 3.362493
[6,] -4.392261 4.762703
```

```
plot(z)
```



kmeans()

Use the "kmeans()" function setting k to 2 and n start 20

Q. How many points are in each?

```
km<-kmeans(z,centers=2)
km
```

```
K-means clustering with 2 clusters of sizes 60, 60

Cluster means:
          x         y
1 -2.931495  2.961711
2  2.961711 -2.931495

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 1 1 1 1 1 1
 [38] 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 2 2 2 2 2 2
 [75] 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 2 2 2 2 2 2
[112] 2 2 2 2 2 2 2 2 2
```

```
Within cluster sum of squares by cluster:
[1] 113.8566 113.8566
 (between_SS / total_SS =  90.1 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

Q. What 'component' of your cluster?

- Cluster size?

```
km$size
```

```
[1] 60 60
```

- Cluster assignment/members?

```
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 1 1 1 1 1 1
 [38] 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 2 2 2 2 2 2 2 2
 [75] 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 2 2 2 2 2 2
[112] 2 2 2 2 2 2 2 2 2
```

- Cluster center

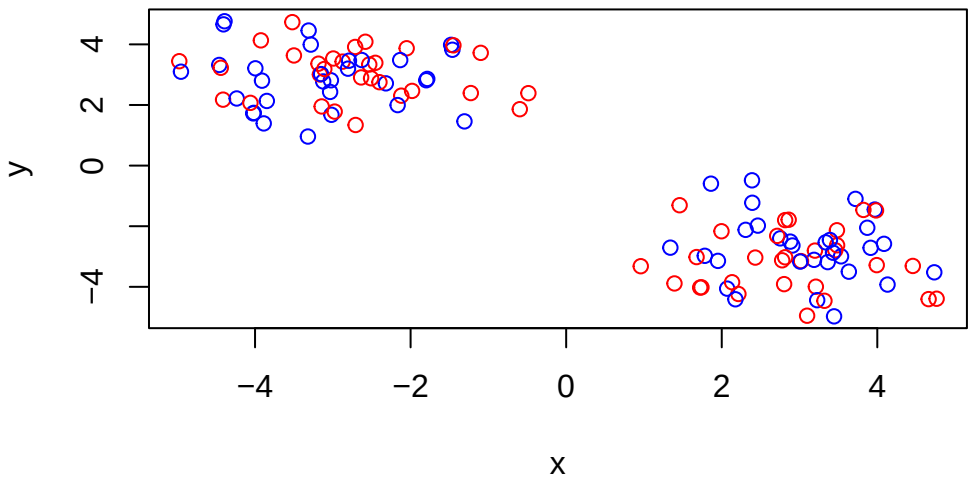
```
km$centers
```

```
          x         y
1 -2.931495  2.961711
2  2.961711 -2.931495
```

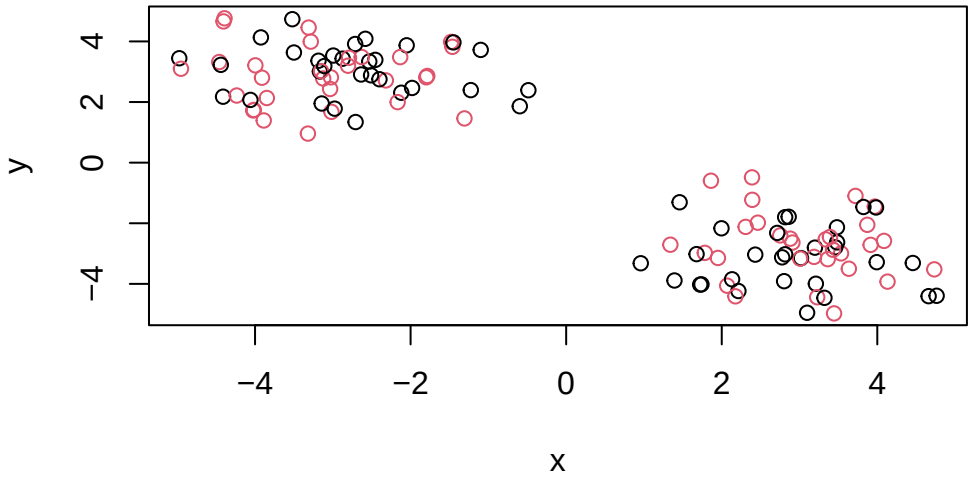
Q. Plot x colored by the 'kmeans()' cluster assignment and add cluster centers as blue points

R will re-cycle the shorter color vector to be the same length as the longer(number of data points) in z

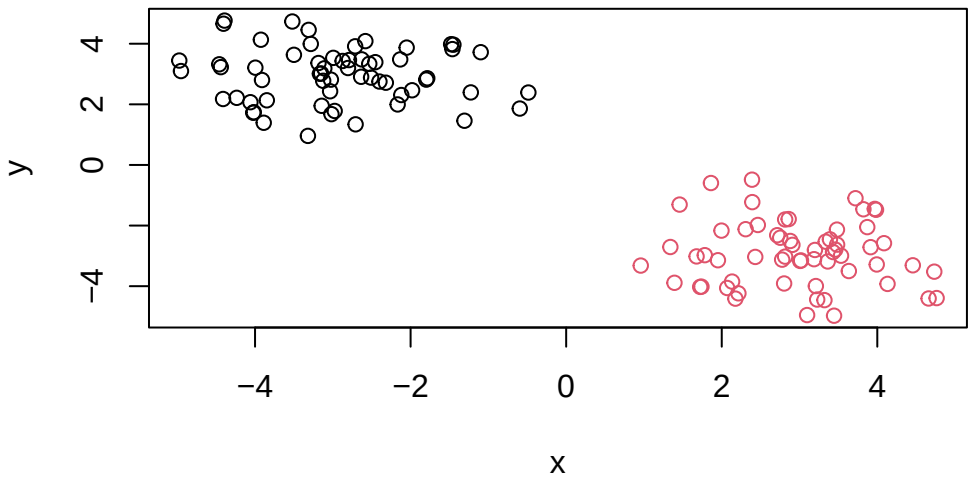
```
plot(z, col=c("red","blue"))
```



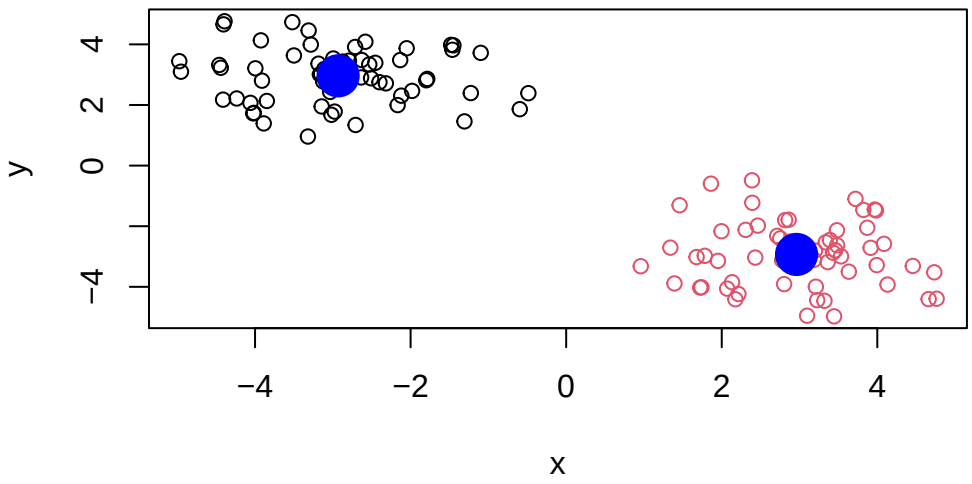

```
#numbers are associated with a color
plot(z, col=c(1,2))
```



```r
# assigning km clusters to the points
# as km clusters have 2 clusters, it is assigning color 1 and 2 to each cluster, just as 1 ar
plot(z, col=km$cluster)
```

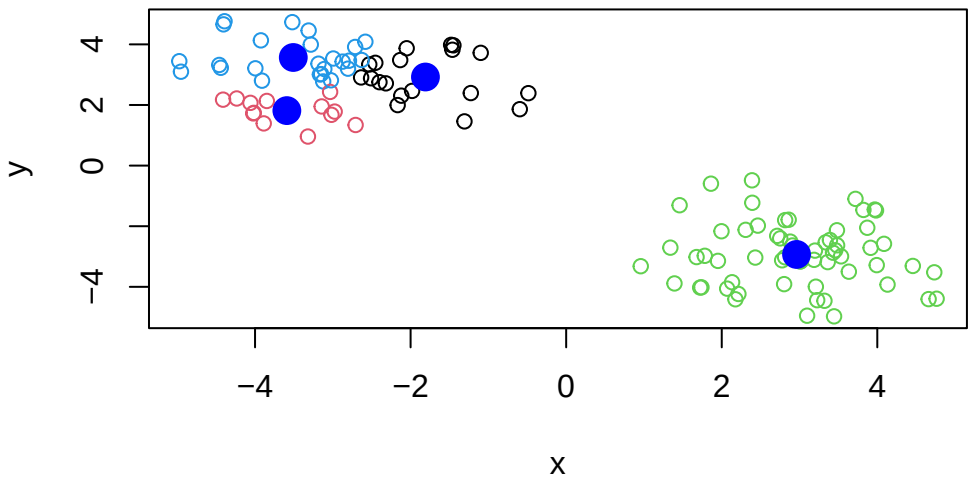



```r
# create a point at the data center, with blue color and a round dot(16= , 17=  , 18=  ) for s
plot(z, col=km$cluster)
points(km$centers, col="blue", pch = 16, cex =3)
```



Q. Can you run kmeans and ask for 4 clusters please and plot the results like me have done above?

```
# create 4 centers for the clusters
km_4<-kmeans(z,centers=4)
# plot the cluster
plot(z, col=km_4$cluster)
# create a point at each center
points(km_4$centers, col="blue", pch = 16, cex =2)
```

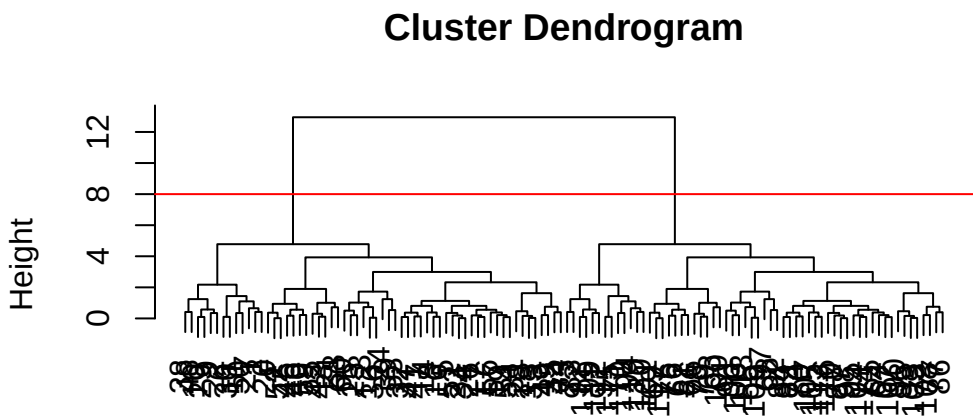


Hierarchical clustering

```
d<-dist(z)
hc<-hclust(d)
hc
```

```
Call:
hclust(d = d)

Cluster method   : complete
Distance         : euclidean
Number of objects: 120
```

```
plot(hc)
abline(h=8, col="red")
```




```
cutree(hc, h=8)
```

```
  [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 1 1 1 1 1
 [38] 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 2 2 2 2 2
 [75] 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 2 2 2 2 2
[112] 2 2 2 2 2 2 2 2 2
```

Data import

```
# import data from the url and store in a var called x
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?

```
# use dim() to get data structure
dim(x)
```

```
[1] 17  5
```

17 rows and 15 columns

Checking your data

```
# previewing 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
```

```
#set the first row as the row-name instead of a column
#rownames(x) <- x[,1]
#x <- x[,-1]
# This is another approach:
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
```

```
#check data structure again
dim(x)
```

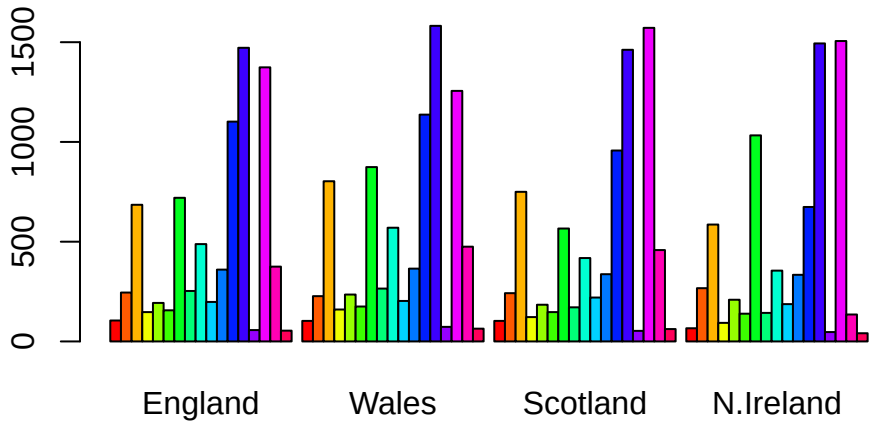
```
[1] 17  4
```

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?

The read.csv(url, row.names=1) approach is better because running rownames(x) <- x[,1] multiple times will make the first data row as row names as they are now at position 1. This have a risk of destroy the dataset, so we want to be careful.

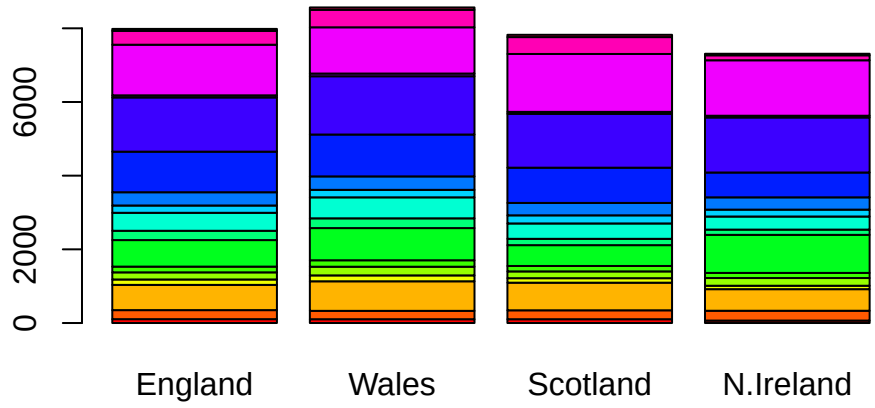
Spotting major differences and trends

```
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?

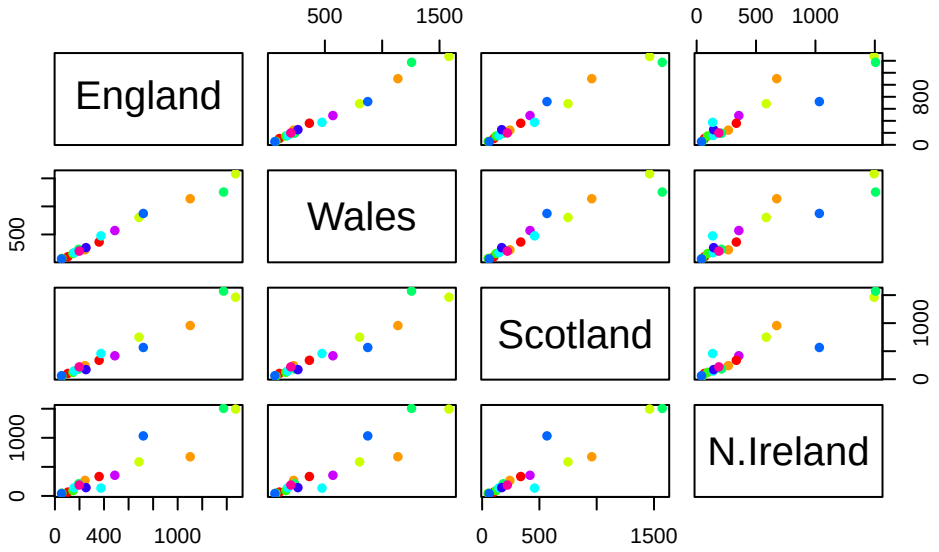
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



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 points from each plot represents a specific food. The points lies on the diagonal represents the consumption of the food is similar between two countries, as the points lies far away from the diagonal represents the difference between.

```
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?

It seems the blue and green dots are obviously different between N. Ireland and other countries.

It is hard to see structure and trends in even this small data-set. How will we never do this when we have big datasets with 1000s or 10s thousands of things we are measuring…

PCA to the rescue

Lets see how PCA deals with this datasets. So main function in base R to do PCA to do PCA is called 'prcomp()'

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

```
Importance of components:
                            PC1      PC2      PC3          PC4
Standard deviation     324.1502 212.7478 73.87622 3.176e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

Let's see what is inside this pca project that we created from running 'prcomp()'

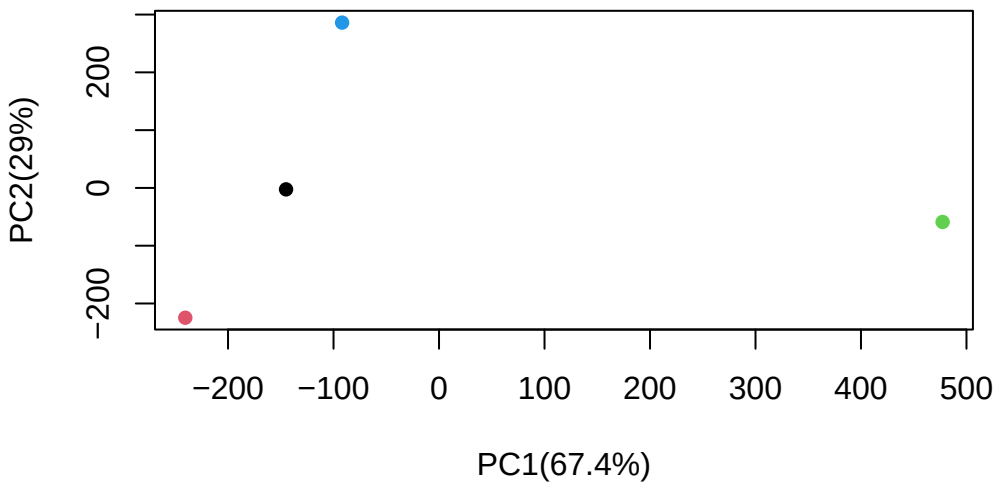
```
attributes(pca)
```

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

$class
[1] "prcomp"
```

```
pca$x
```

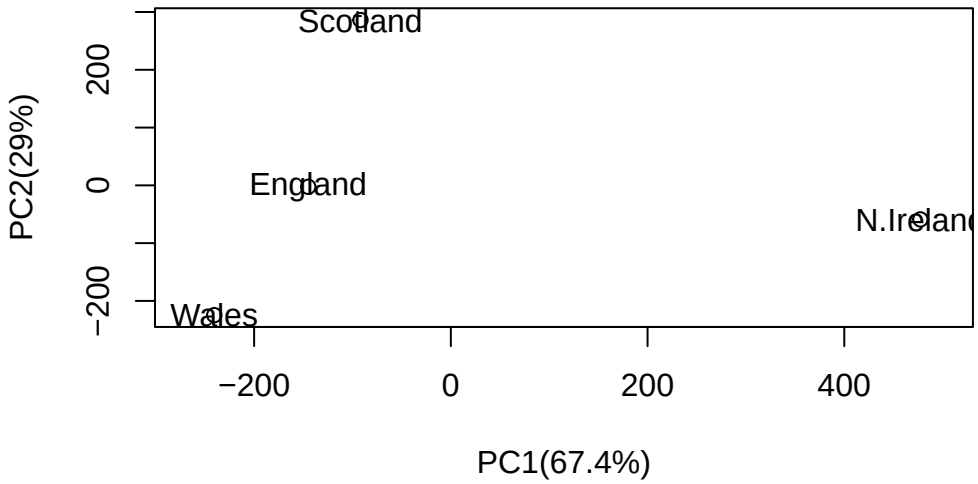
```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -4.894696e-14
Wales     -240.52915 -224.646925 -56.475555  5.700024e-13
Scotland   -91.86934  286.081786 -44.415495 -7.460785e-13
N.Ireland  477.39164  -58.901862  -4.877895  2.321303e-13
```

```
plot(pca$x[,1],pca$x[,2],col=c(1,2,4,3), pch=16,
     xlab = "PC1(67.4%)", ylab = "PC2(29%)")
```



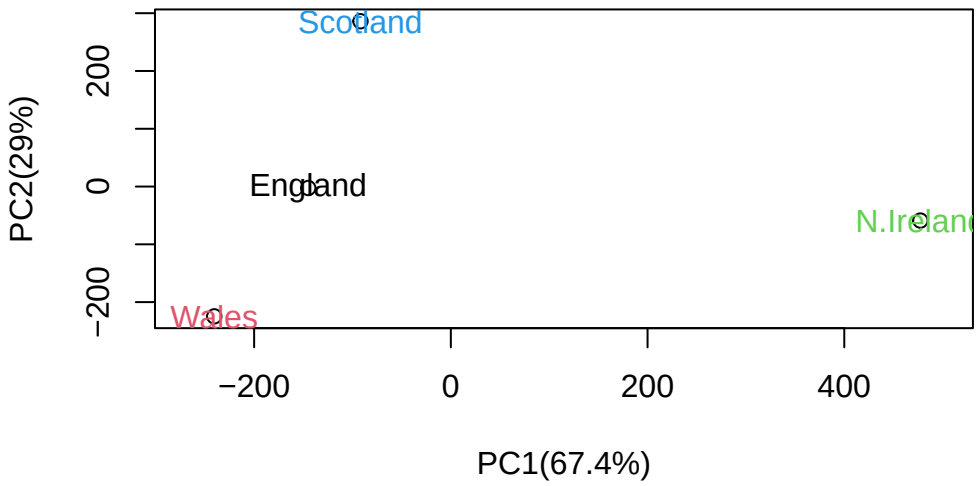
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
plot(pca$x[,1],pca$x[,2], xlab = "PC1(67.4%)", ylab = "PC2(29%)", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x))
```



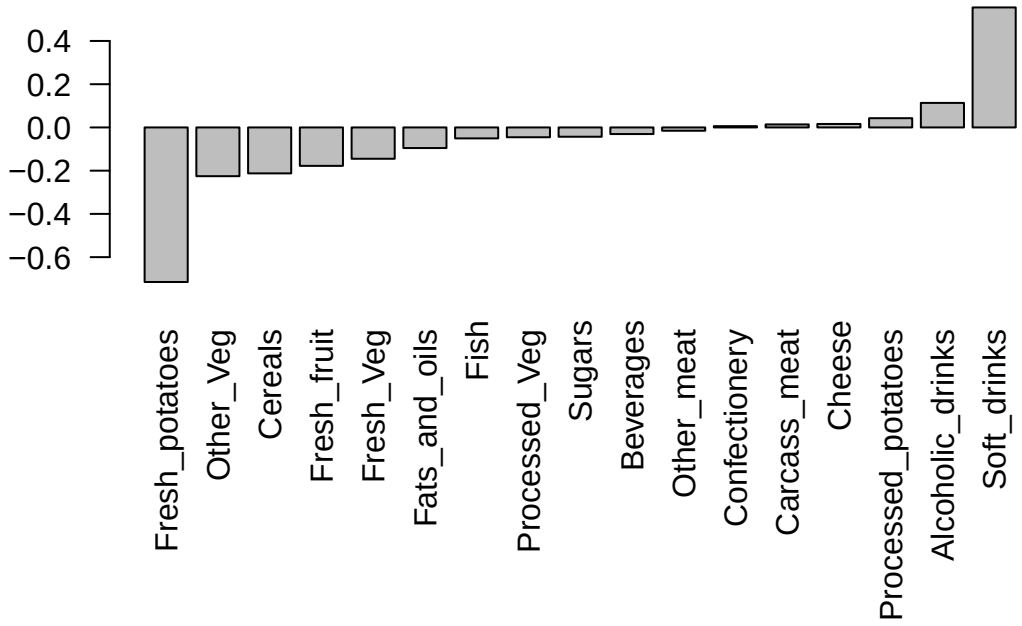
Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
plot(pca$x[,1],pca$x[,2],xlab = "PC1(67.4%)", ylab = "PC2(29%)", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x),col=c(1,2,4,3), pch=16)
```



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

```
## loadings plot for PC2, sorted
par(mar=c(10, 3, 0.35, 0))
barplot(sort(pca$rotation[,2]),las=2)
```



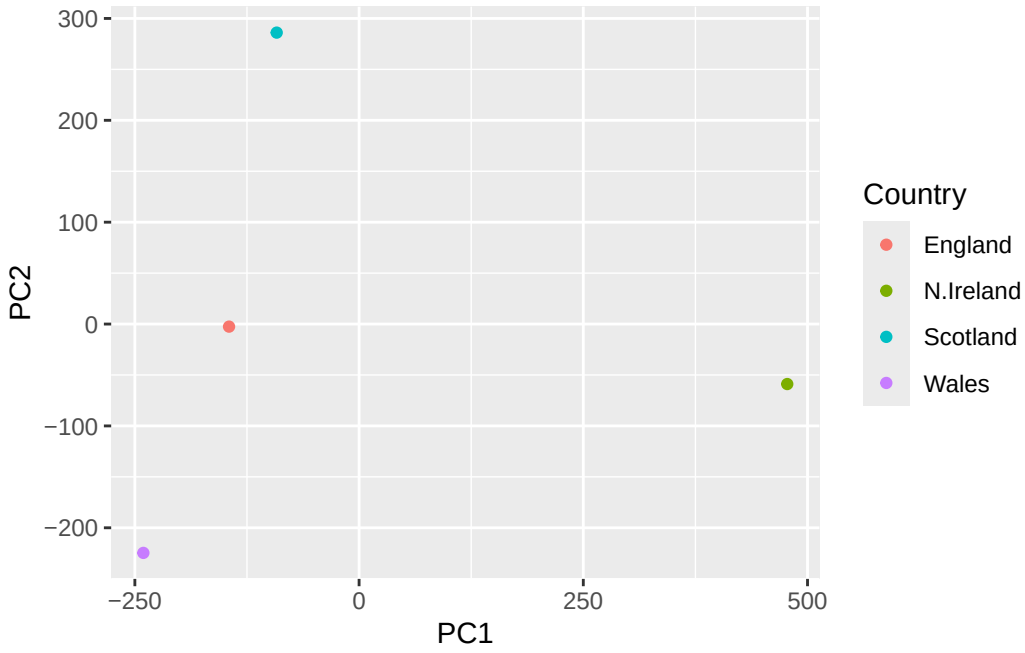
Fresh potatos and Soft drinks.

Using ggplot for these figures

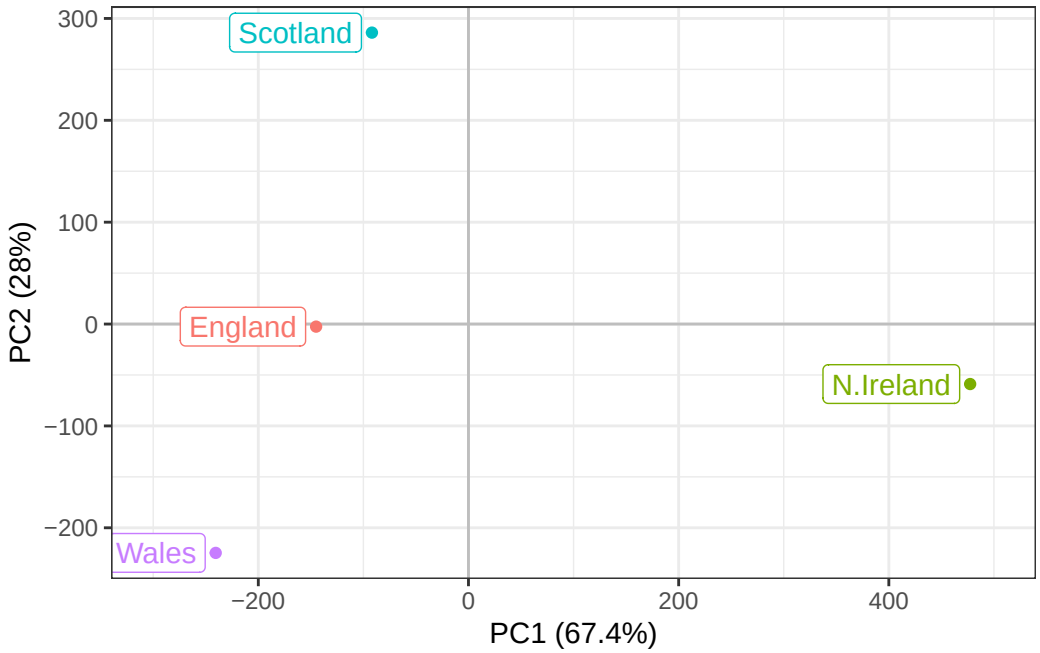
```
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 4.3.3
```

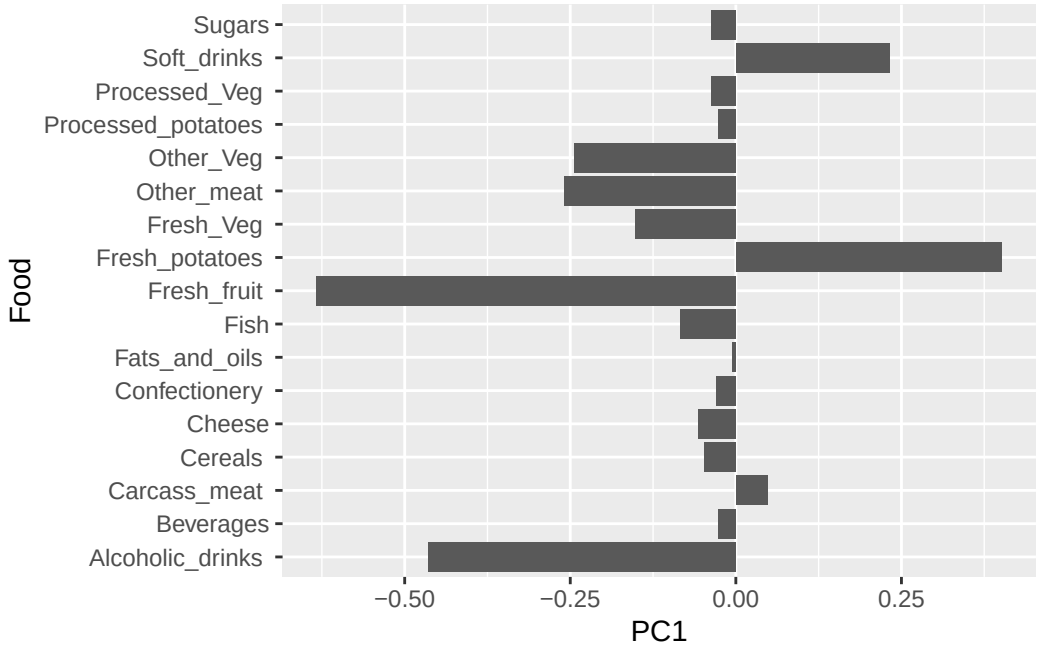
```
df <- as.data.frame(pca$x)
df_lab <- tibble::rownames_to_column(df, "Country")

# Our first basic plot
ggplot(df_lab) +
  aes(PC1, PC2, col=Country) +
  geom_point()
```



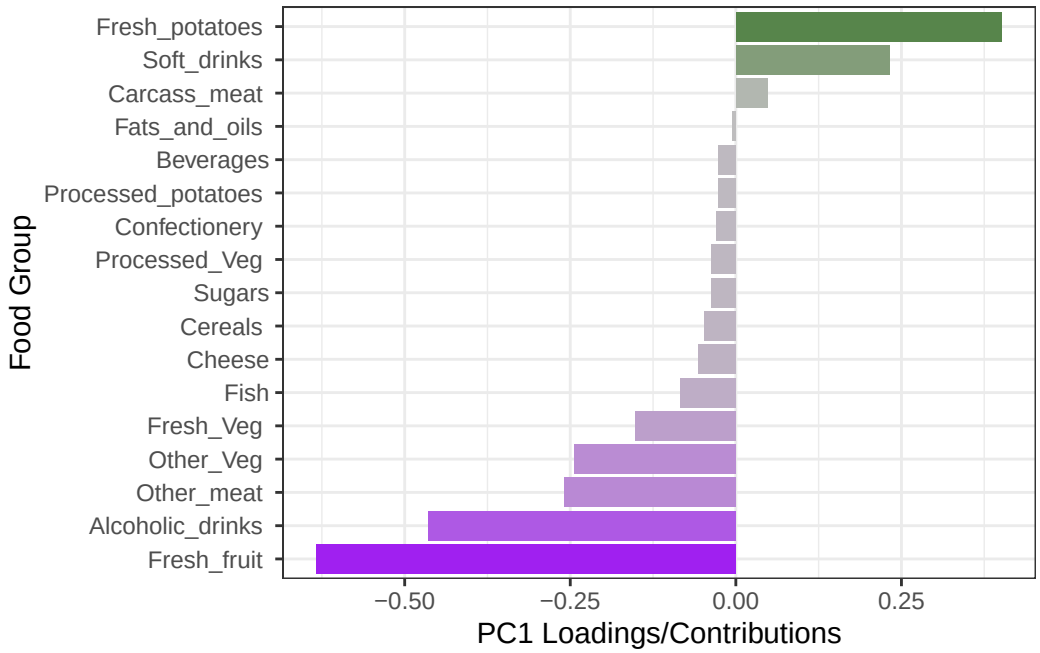

```
# make it look nicer
ggplot(df_lab) +
  aes(PC1, PC2, col=Country, label=Country) +
  geom_hline(yintercept = 0, col="gray") +
  geom_vline(xintercept = 0, col="gray") +
  geom_point(show.legend = FALSE) +
  geom_label(hjust=1, nudge_x = -10, show.legend = FALSE) +
  expand_limits(x = c(-300,500)) +
  xlab("PC1 (67.4%)") +
  ylab("PC2 (28%)") +
  theme_bw()
```




```
# bar plot with ggplot
ld <- as.data.frame(pca$rotation)
ld_lab <- tibble::rownames_to_column(ld, "Food")

ggplot(ld_lab) +
  aes(PC1, Food) +
  geom_col()
```

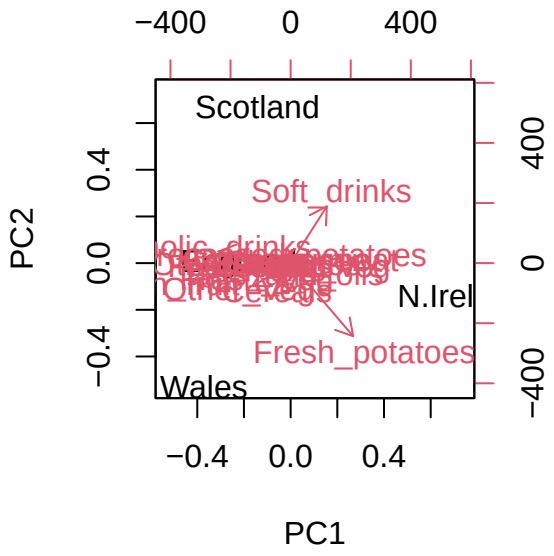



```
ggplot(ld_lab) +
  aes(PC1, reorder(Food, PC1), bg=PC1) +
  geom_col() +
  xlab("PC1 Loadings/Contributions") +
  ylab("Food Group") +
  scale_fill_gradient2(low="purple", mid="gray", high="darkgreen", guide=NULL) +
  theme_bw()
```



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
```

Q10: How many genes and samples are in this data set?

```
dim(rna.data)[1]
```

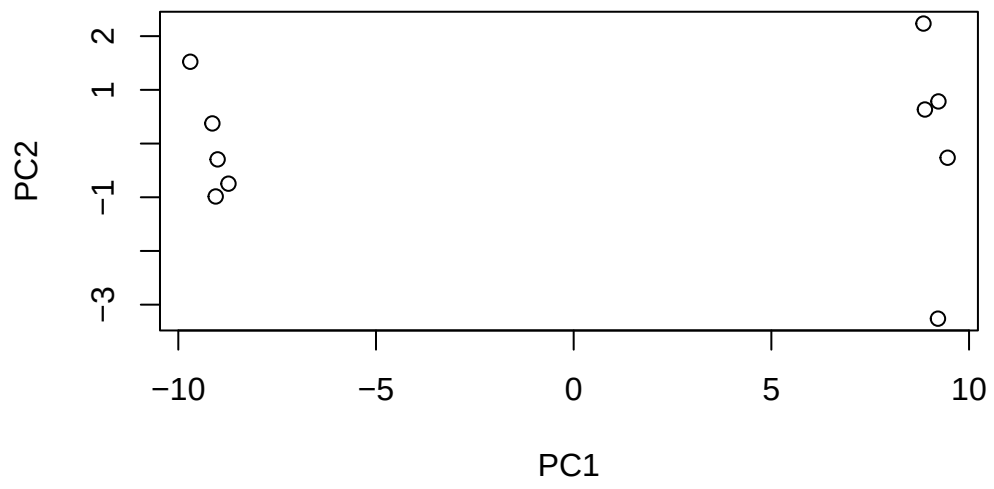
```
[1] 100
```

```
dim(rna.data)[2]
```

```
[1] 10
```

100 gene and 10 samples

```
## 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")
```

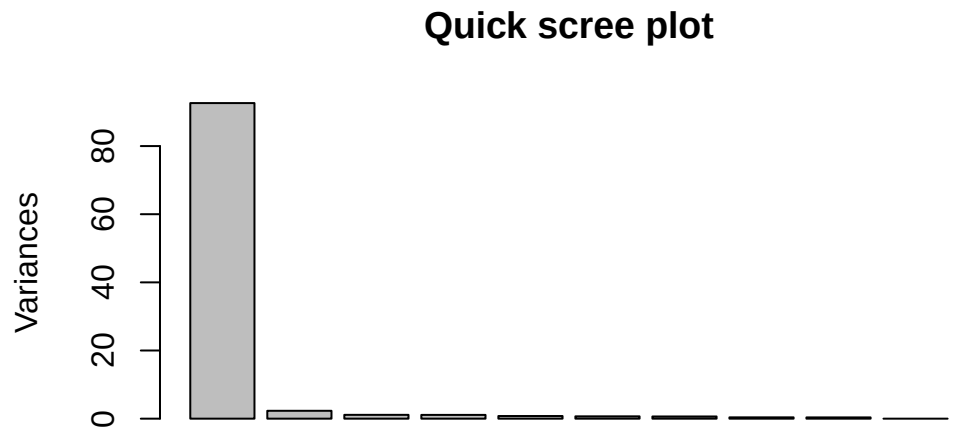



```
summary(pca)
```

```
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 3.457e-15
```

```
Proportion of Variance 0.00385 0.00364 0.000e+00
Cumulative Proportion  0.99636 1.00000 1.000e+00
```

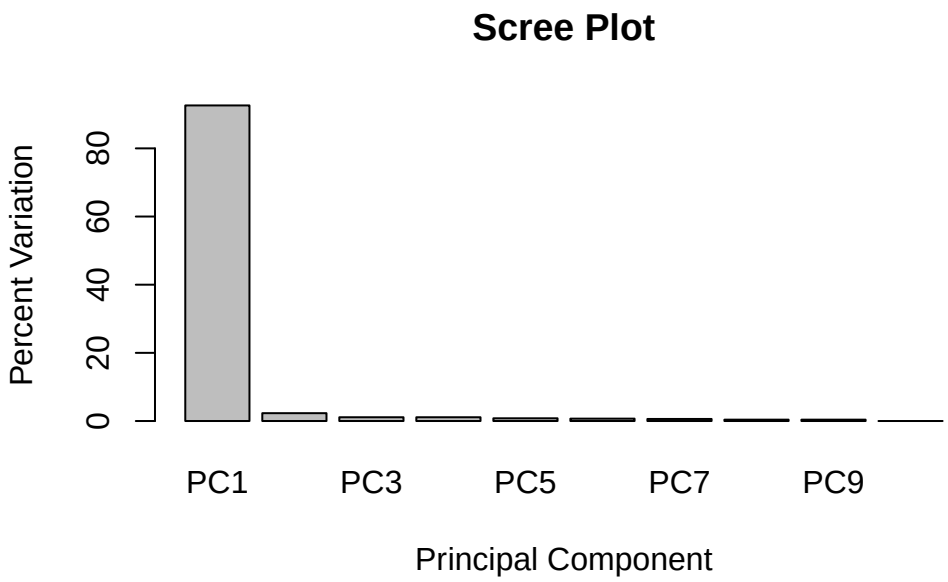
```
plot(pca, main="Quick scree plot")
```



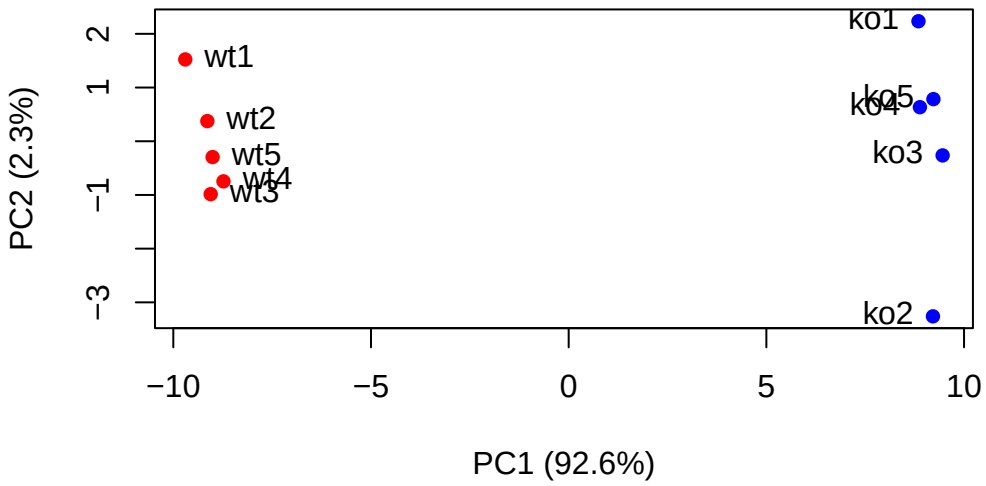

```
## Variance captured per PC
pca.var <- pca$sdev^2

## 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
```

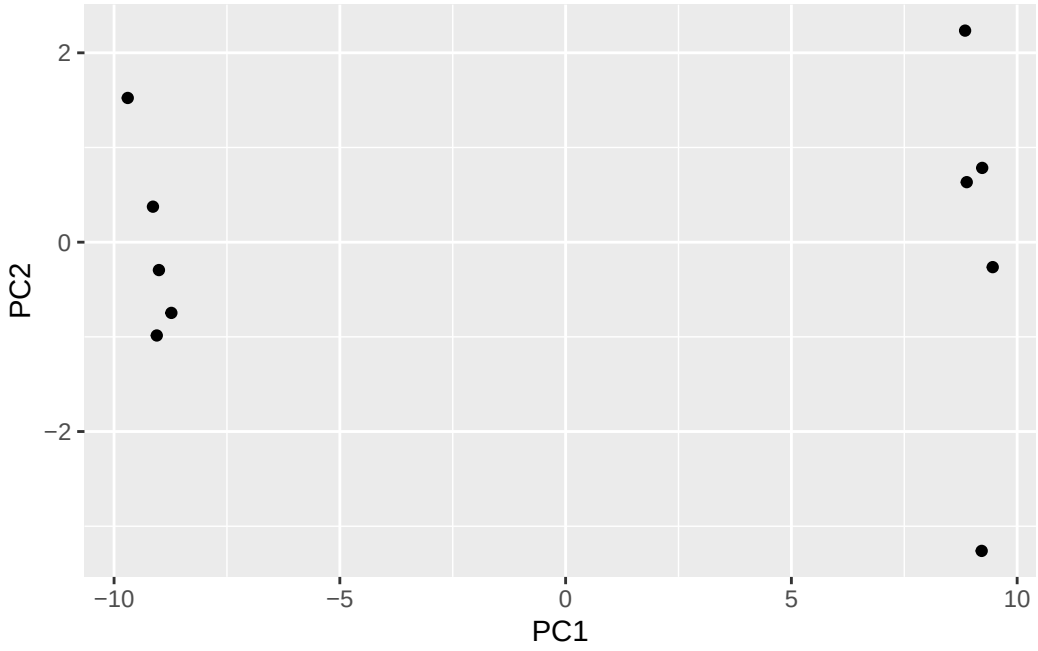
```
barplot(pca.var.per, main="Scree Plot",
        names.arg = paste0("PC", 1:10),
        xlab="Principal Component", ylab="Percent Variation")
```



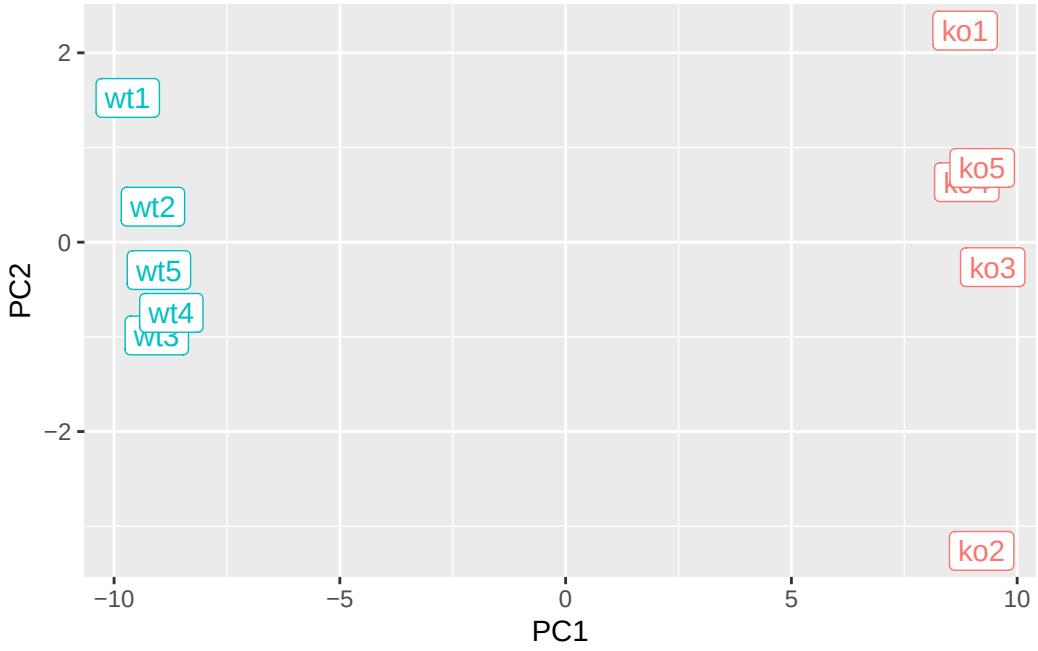

```
## A vector of colors for wt and ko samples
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

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)))
```



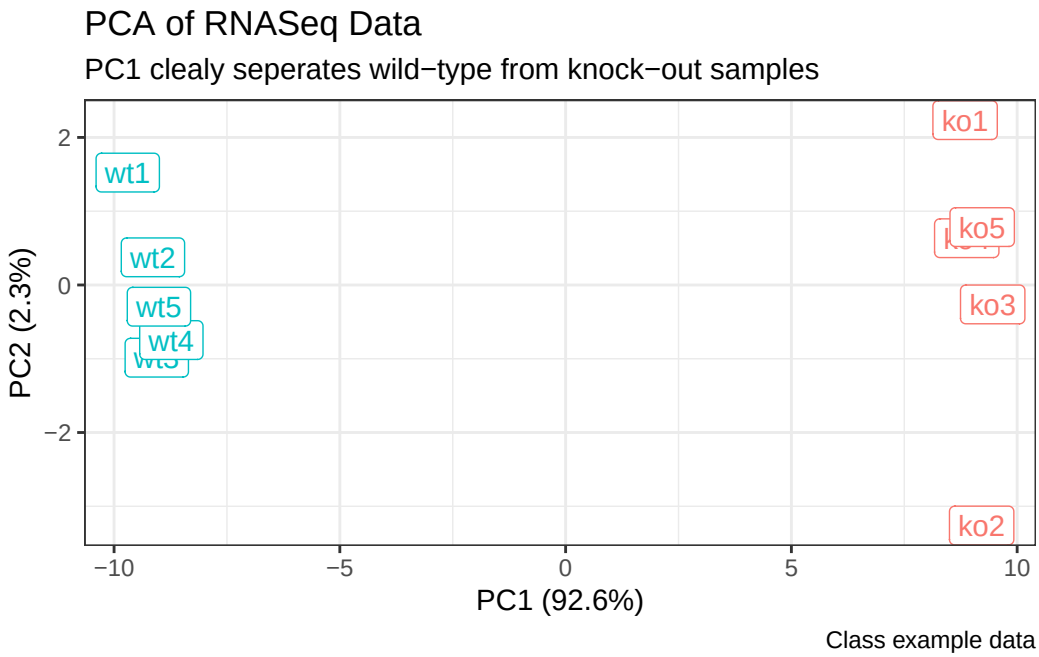

```
#library(ggplot2)

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="Class example data") +
     theme_bw()
```


PCA of RNASeq Data
PC1 clealy seperates wild−type from knock−out samples
ko1
wt1
ko5
wt2
ko3
wt5
wt4
ko2
2
0
−2
PC2 (2.3%)
−10
−5
0
5
10
PC1 (92.6%)
Class example data