Class 7: Machine Learning 1

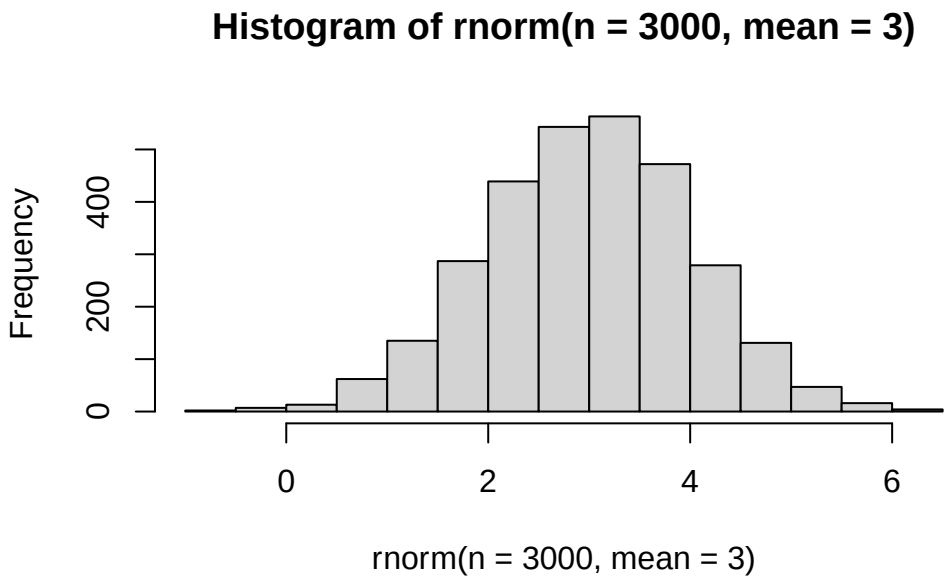
Yaniv Iny (PID:A18090586

Today we will explore unsupervised machine learning methods including clustering and dimensionality reduction methods.

Let's start by making up some data (where we know there are clear groups) that we cans use to test out different clustering methods.

We can use the `rnorm()` function to help us here.

```
hist( rnorm(n=3000, mean=3))
```



Maka data with two "clusters"

```r
rnorm(30, mean=-3)
```

```
 [1] -4.6509205 -3.5182382 -1.3451984 -1.9913666 -4.1015001 -4.6408188
 [7] -2.7362652 -4.1506600 -2.7372411 -1.2762107 -2.3451460 -2.1790611
[13] -1.6123614 -1.6455144 -3.0067532 -2.4498934 -4.2800646 -1.5693193
[19] -3.6700829 -4.3876040 -3.0731000 -3.5306117 -3.6284063 -3.1932948
[25] -3.1651918 -3.6891133 -4.0156109 -0.9890104 -3.6266892 -3.6581970
```

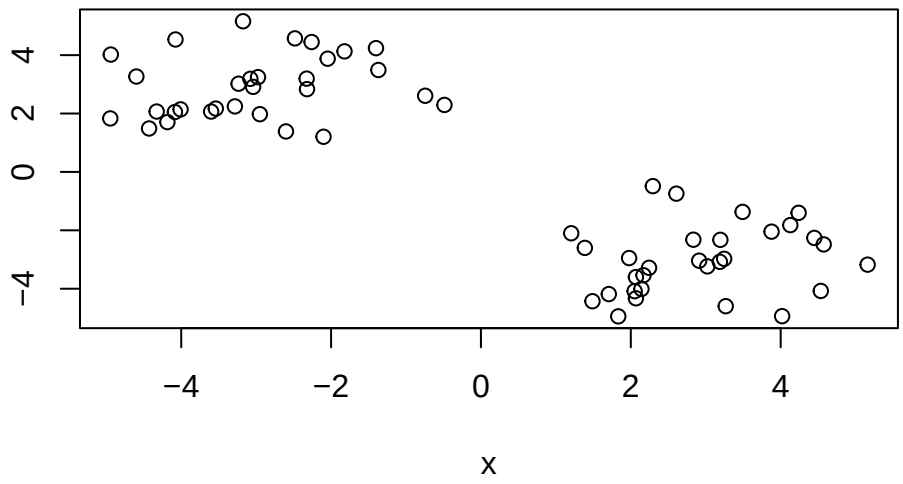
```r
rnorm(30, mean = +3)
```

```
 [1] 2.614250 2.144188 3.842839 2.394593 1.832834 3.483717 2.598581 4.463182
 [9] 2.675834 1.898151 1.094417 1.463168 2.442005 4.834874 2.435494 3.066762
[17] 2.618904 2.859983 2.133250 2.113777 1.575965 1.911324 2.290022 2.457374
[25] 4.215312 3.145305 4.640862 4.900828 3.853942 3.054499
```

```r
x<- c(rnorm(30, mean=-3),
rnorm(30, mean = +3))

z <- cbind(x=x,rev(x))
head(z)
```

```
            x
[1,] -4.940042 4.019674
[2,] -3.234489 3.021461
[3,] -3.538299 2.169452
[4,] -4.083971 2.051238
[5,] -4.428989 1.488085
[6,] -2.950248 1.977737
```

```r
plot(z)
```



How big is z

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

```
K-means clustering with 2 clusters of sizes 30, 30

Cluster means:
          x
1  2.912899 -3.014359
2 -3.014359  2.912899

Clustering vector:
 [1] 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 1 1 1 1 1 1 1 1
[39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Within cluster sum of squares by cluster:
[1] 75.06942 75.06942
 (between_SS / total_SS =  87.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
```

```
[6] "betweenss"    "size"         "iter"         "ifault"
```

K-means clustering

The main function in "base" R for K-means clustering is called `kmeans()`

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

```
K-means clustering with 2 clusters of sizes 30, 30

Cluster means:
          x
1 -3.014359  2.912899
2  2.912899 -3.014359

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 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] 75.06942 75.06942
 (between_SS / total_SS =  87.5 %)

Available components:

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

```
attributes(k)
```

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

$class
[1] "kmeans"
```

Q. How many points lie in each cluster

```
k$size
```

```
[1] 30 30
```

Q. what componenet of our results tells us about the cluster membership (i.e which point likes in which cluster)?

```
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 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
```

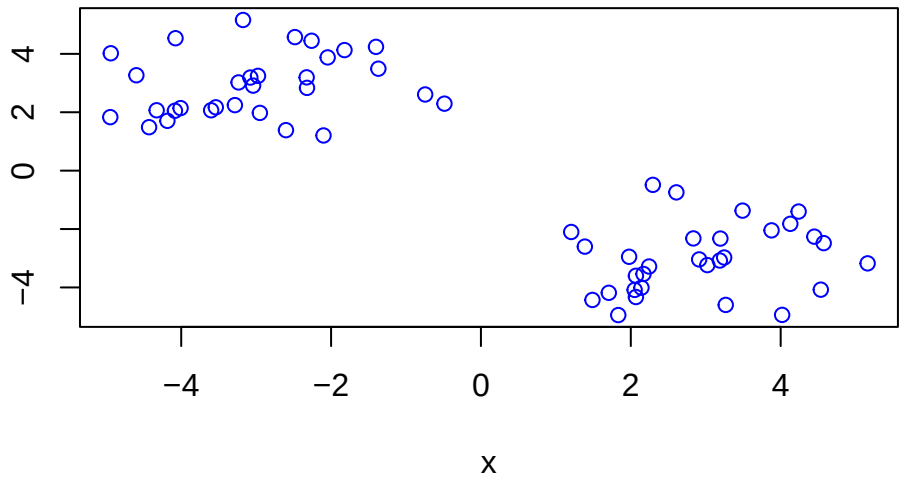
Q. Center of each cluster?

```
k$centers
```

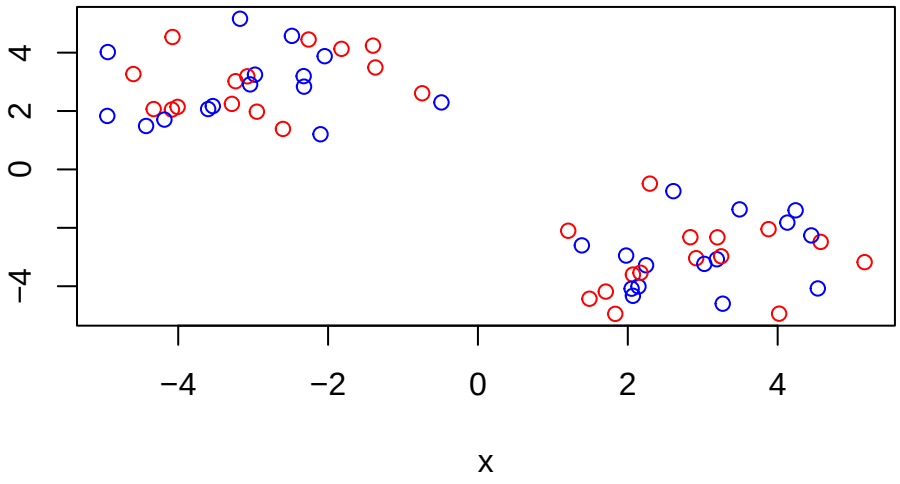
```
          x
1 -3.014359  2.912899
2  2.912899 -3.014359
```

Q. Put this result info together and make a little "base R" plot of our clustering results. Also add the cluster center points to this plot.

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

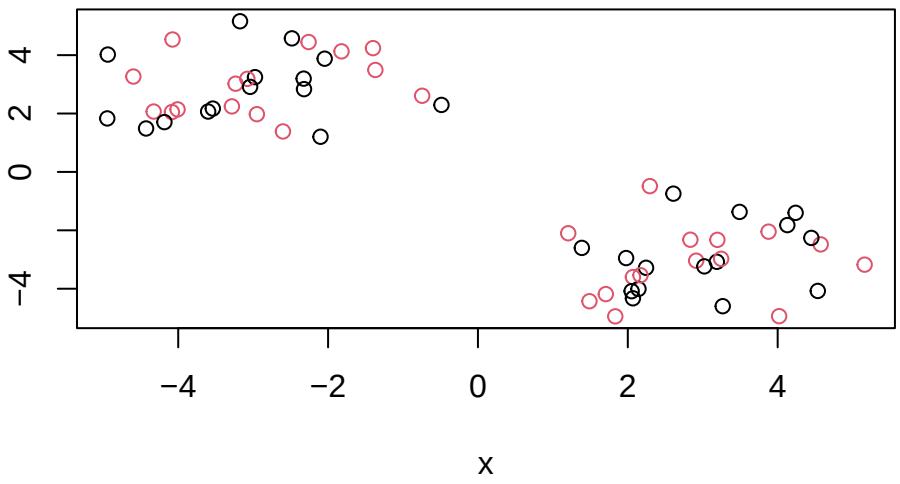


```
plot(z, col=c("blue", "red"))
```



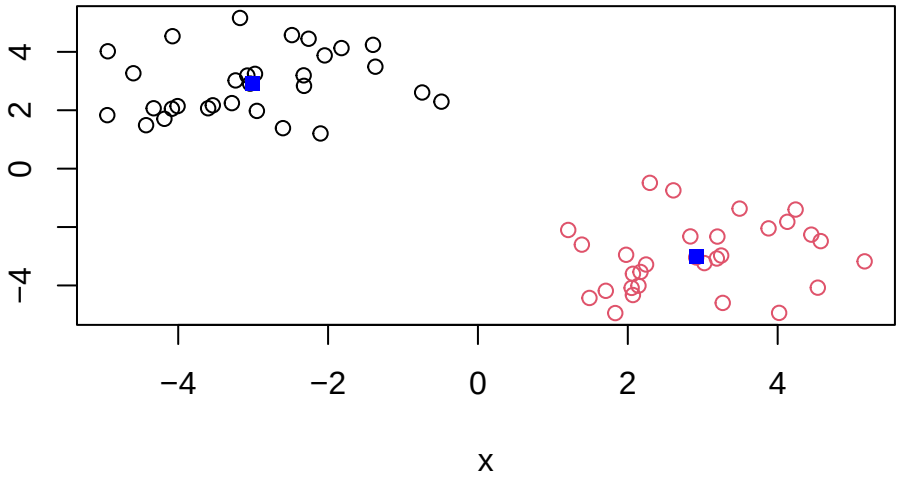
You can color by number

```
plot(z, col =c(1,2))
```



Plots colored by cluster membership:

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

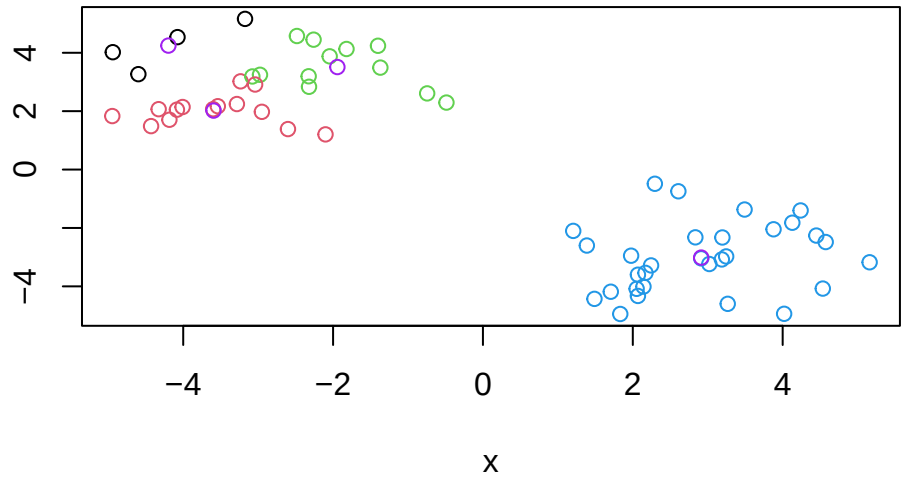


Q. Run kmeans on our input **z** and define 4 clusters making the same result vizualization plot as above(plot of z colored by cluster membership).

```
head(z)
```

```
            x
[1,] -4.940042 4.019674
[2,] -3.234489 3.021461
[3,] -3.538299 2.169452
[4,] -4.083971 2.051238
[5,] -4.428989 1.488085
[6,] -2.950248 1.977737
```

```
k4 <- kmeans(z, centers=4)
plot(z, col=k4$cluster)
points(k4$centers, col="purple")
```



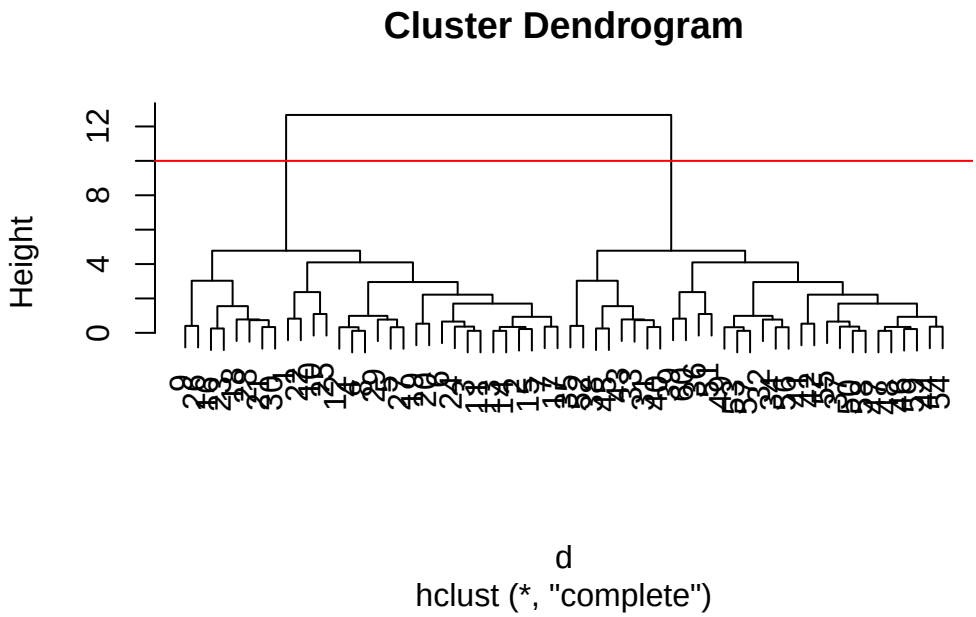
Hierarchical Clustering

The main function in base R for this is called `hclust()` it will take as input a distance matrix (key poiny is that you can't just give your "raw" data as input - you have to first calculate a distance matrix from your data).

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

```
Call:
hclust(d = d)

Cluster method   : complete
Distance         : euclidean
Number of objects: 60
```

```r
plot(hc)
abline(h=10, col="red")
```



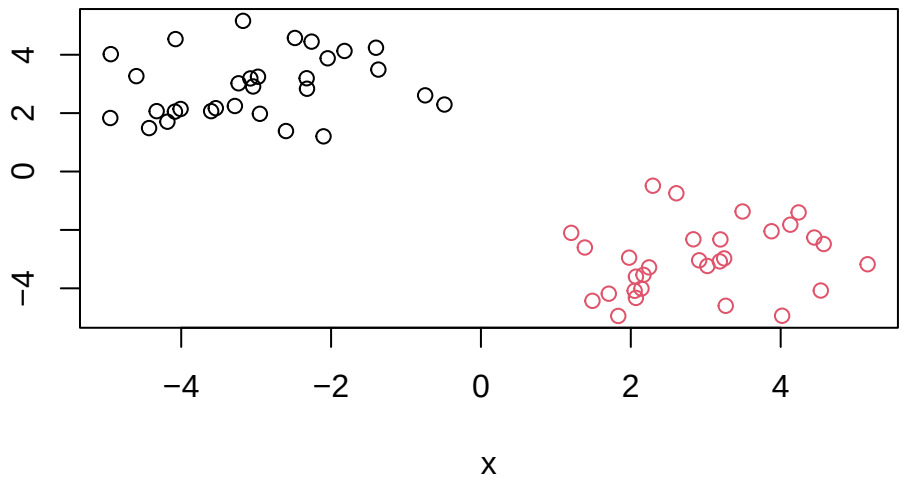
Once I inspect the Dendrogram I can "cut" the tree to yield my froupings or clusters. The function to do this is called `cutree()`

```r
cutree(hc, h=10)
```

```
 [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
```

```
grps <- cutree(hc, h=10)
```

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



Lets examine some silly 17-dimensional data detailing good consumption in the UK(England,Scotland,Wales, and N.Ireland). Are these countries eating habits different or similar and if so how.

Data import

```
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
```

> Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

we can use the dim() function, which will return both the number of rows and columns in a data frame. Alternatively, we can use nrow() for rows and ncol() for columns separately. There are 17 rows, 4 columns, and 17+4 columns + rows.

```
nrow(x)
```

```
[1] 17
```

```
ncol(x)
```

```
[1] 4
```

```
dim(x)
```

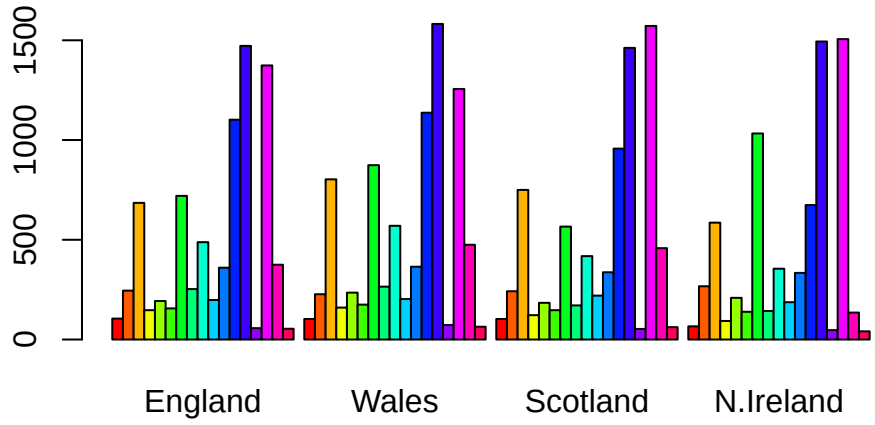
```
[1] 17  4
```

> 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 second approach, using row.names=1 when reading the CSV, is cleaner and more efficient, especially if you're working with a large dataset where you want to avoid mistakes from manually altering the data frame later.

The second approach is more robust as it eliminates the chance of modifying the dataset unintentionally.

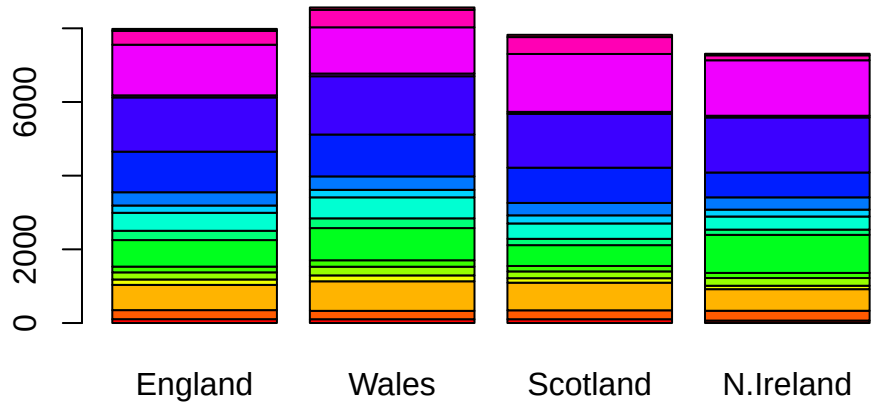
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

Changing the besides from True to False results in the bar graph.

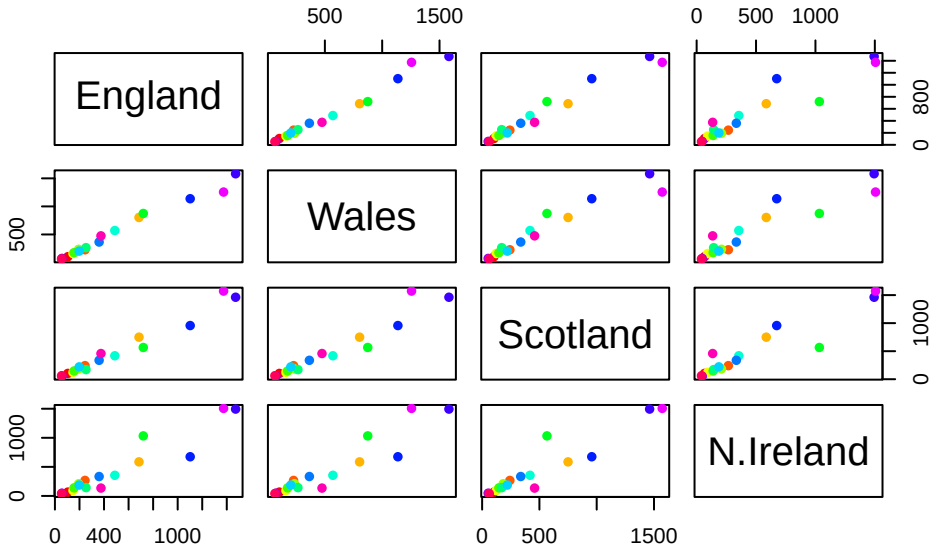
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



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?

If a point lies on the diagonal of a pairwise plot, it indicates that the two variables being compared are perfectly correlated with each other, meaning they change in tandem. In this case, for the pairwise plots, the diagonal shows that the variables are compared against themselves.

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



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

Looking at this data set we can see that northern Ireland has a significantly different food consumption thatn the other countries, although it is still hard to see this on these type of plots as the details are not very easily understod.

Looking at these types of "pairwise plots" cam be helpful but it does not scale well and is much more time consuming. There has got to be a better way…

PCA to the rescue!

The main function for PCA in base R is called `prcomp()`. This function wants transpose of our input data - i.e the important foods in as columns and the countries as rows.

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

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 2.921e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

Lets see what is in our result object `pca`

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

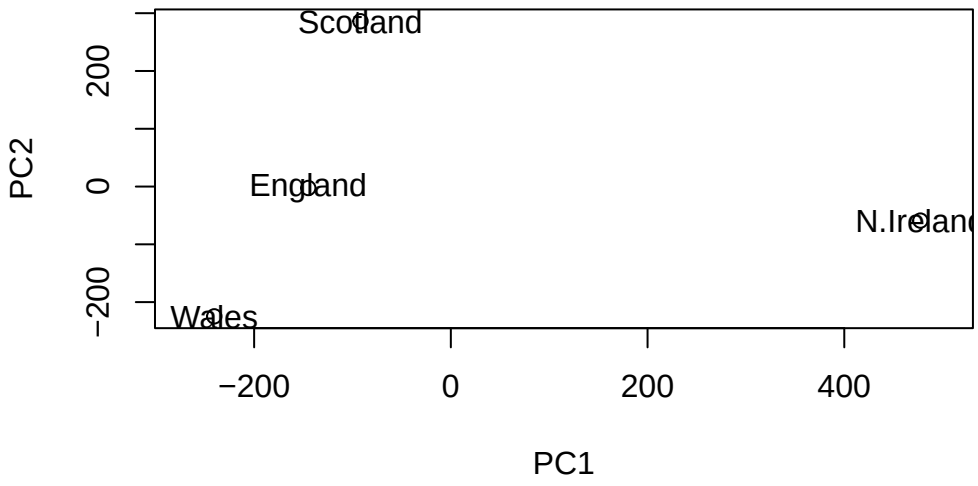
The `pca$x` results object is wehre I will focus first as this details how the countries are related to eachother in terms of our new "axis" (aka "PCs"," eigenvectors", etc.)

```
head(pca$x)
```

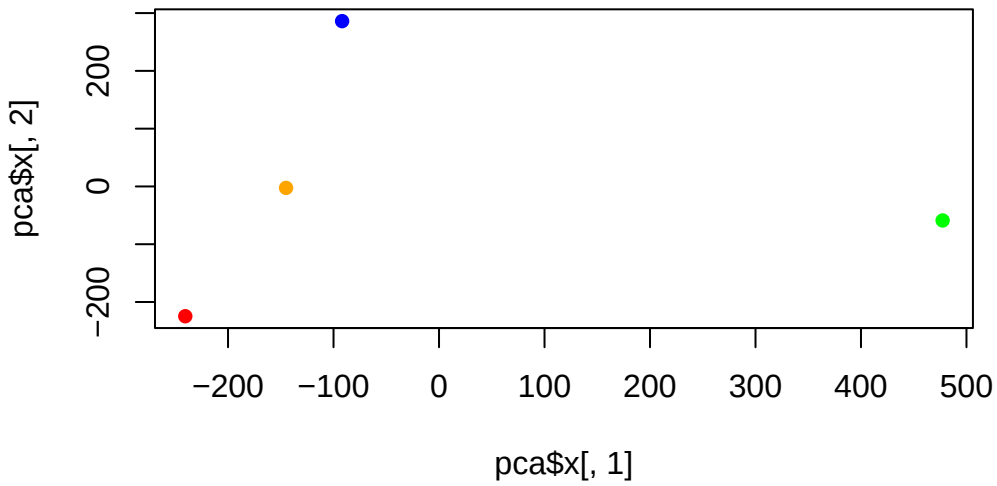
```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -9.152022e-15
Wales     -240.52915 -224.646925 -56.475555  5.560040e-13
Scotland   -91.86934  286.081786 -44.415495 -6.638419e-13
N.Ireland  477.39164  -58.901862  -4.877895  1.329771e-13
```

> 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", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x))
```

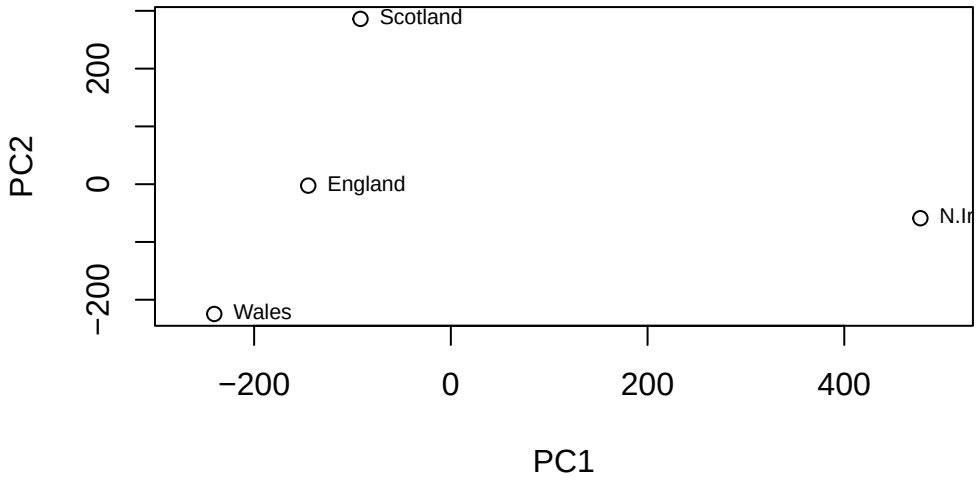



```
plot(pca$x[,1], pca$x[,2],pch=16, col = c("orange", "red","blue","green"))
```



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", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], labels=rownames(pca$x), pos=4, cex=0.7)
```



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.

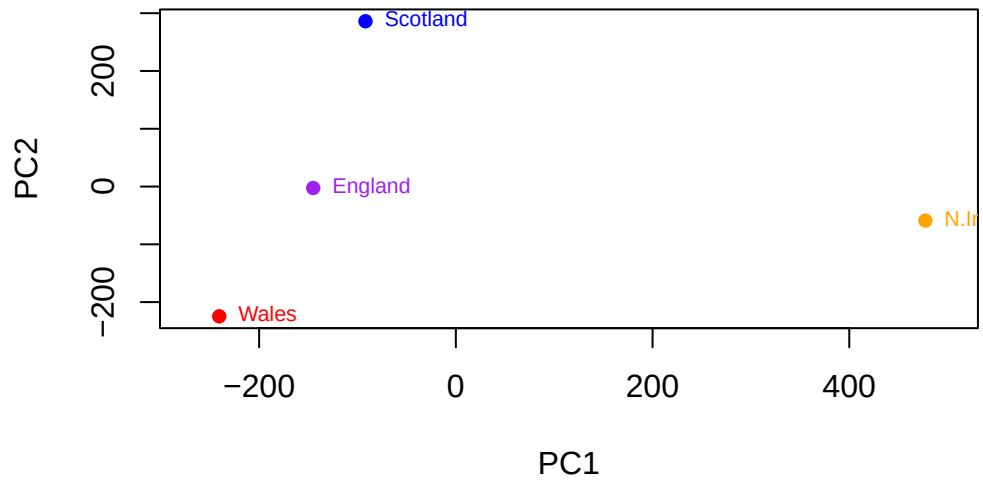
```
country_vector <- c("UK", "Ireland", "Scotland", "Wales", "Northern Ireland")
```

```
country_colors <- c("UK" = "red",
                    "Ireland" = "green",
                    "Scotland" = "blue",
                    "Wales" = "purple",
                    "Northern Ireland" = "orange")
```

```
countries <- factor(country_vector)
```

```
point_colors <- country_colors[countries]
```

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500), col=point_colors, pch=16
text(pca$x[,1], pca$x[,2], labels=rownames(pca$x), col=point_colors, pos=4, cex=0.7)
```



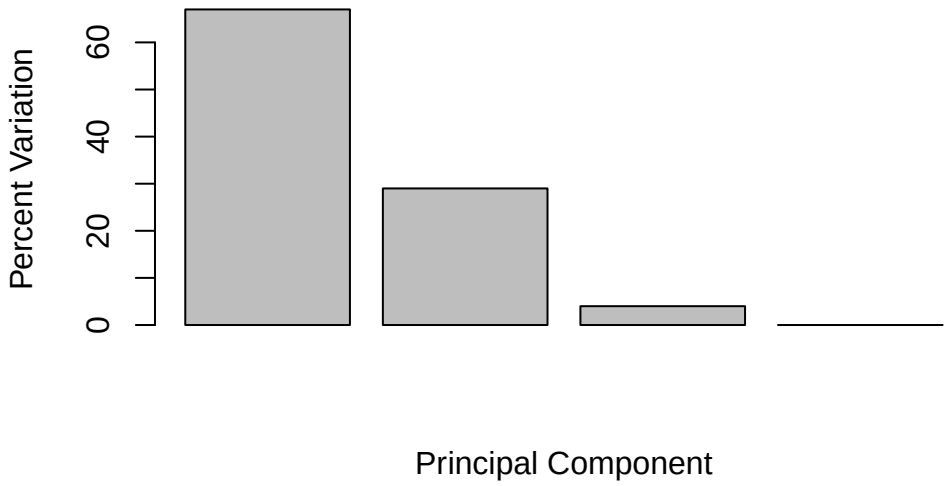

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

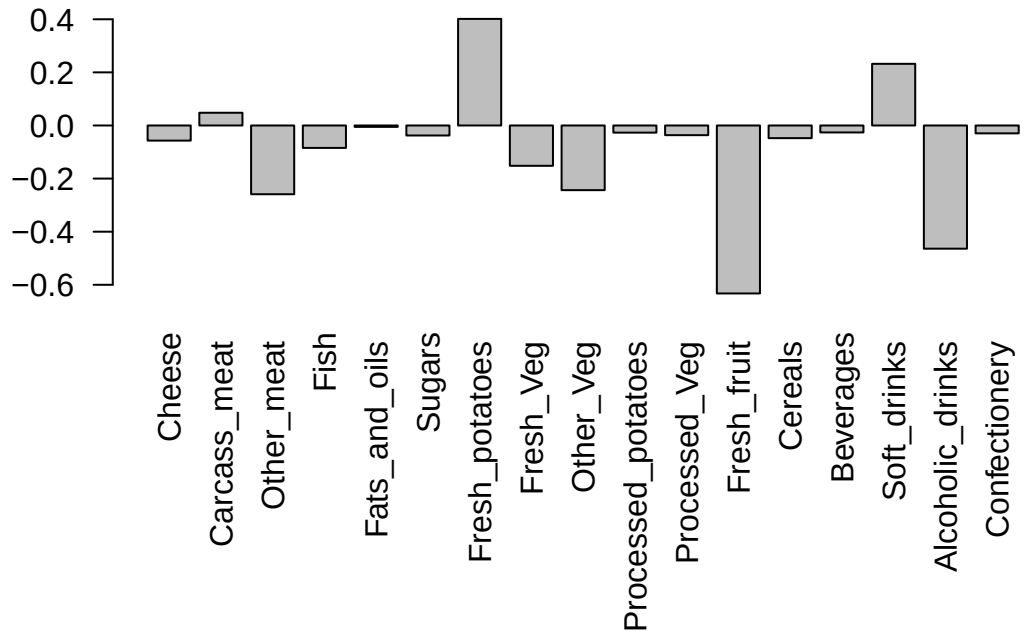
```
z <- summary(pca)
z$importance
```

```
                             PC1       PC2      PC3          PC4
Standard deviation     324.15019 212.74780 73.87622 2.921348e-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")
```



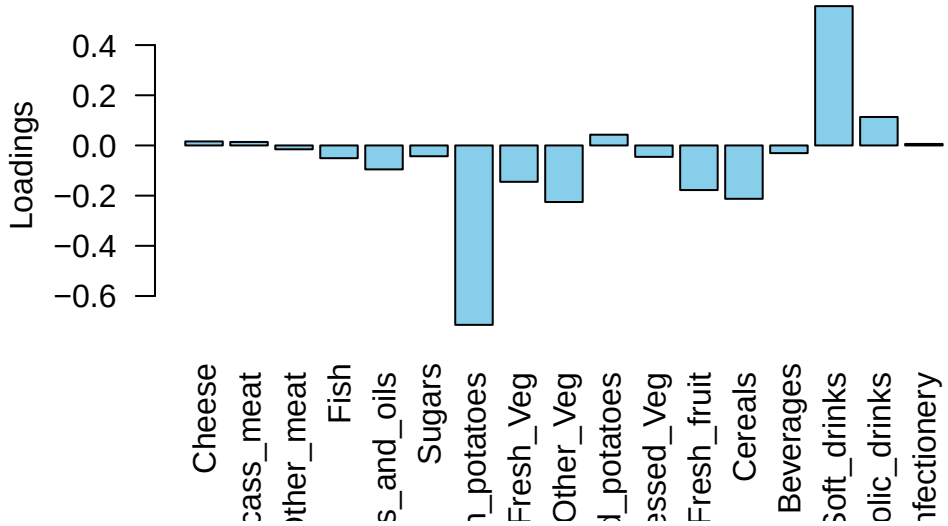

```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



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))
```

```
bar_midpoints <- barplot(pca$rotation[,2], las=2, col="skyblue",
                         main="PC2 Loadings", ylab="Loadings")
```




```
dev.flush()
```

```
[1] 0
```

PCA OF RNA SEQ DATA

n this example, a small RNA-seq count data set (available from the class website (expression.csv and the tinyurl short link: "https://tinyurl.com/expression-CSV") is read into a data frame called rna.data where the columns are individual samples (i.e. cells) and rows are measurements taken for all the samples (i.e. genes).

```
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 10 samples, and there are, 100 genes.

```
ncol(rna.data)
```

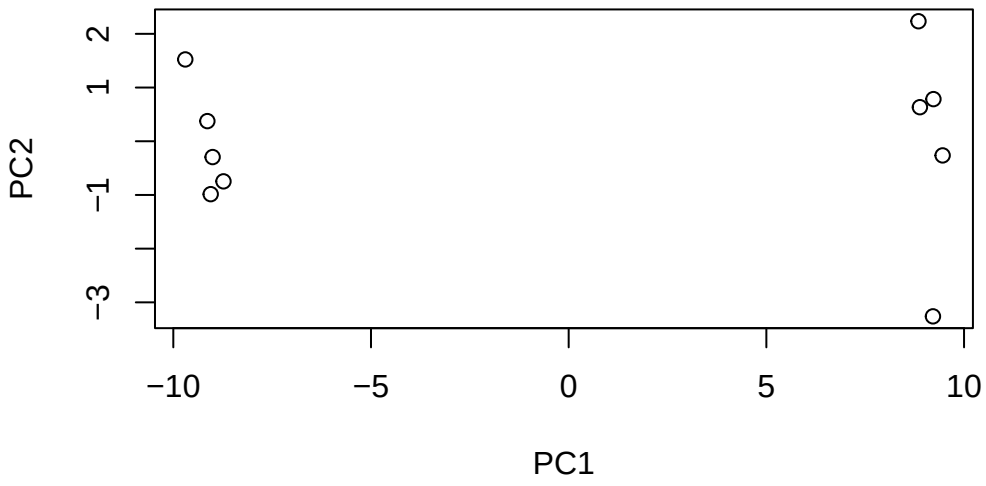
```
[1] 10
```

```
nrow(rna.data)
```

```
[1] 100
```

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

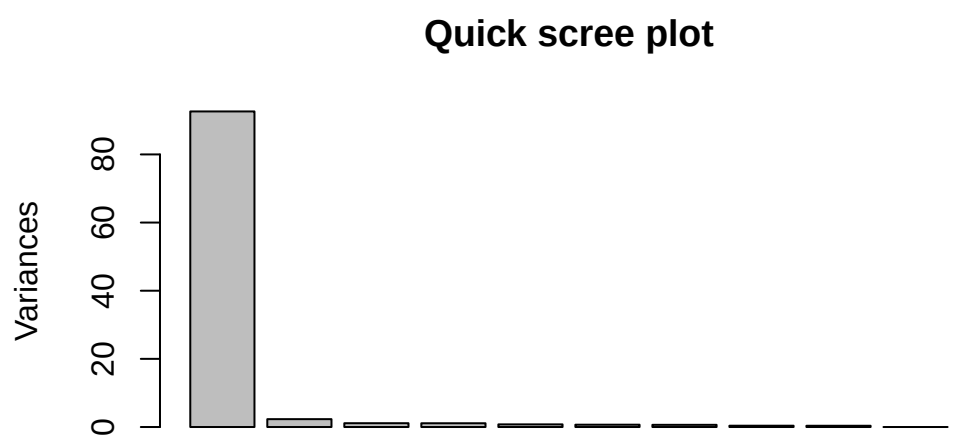
```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```




```
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 3.345e-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")
```

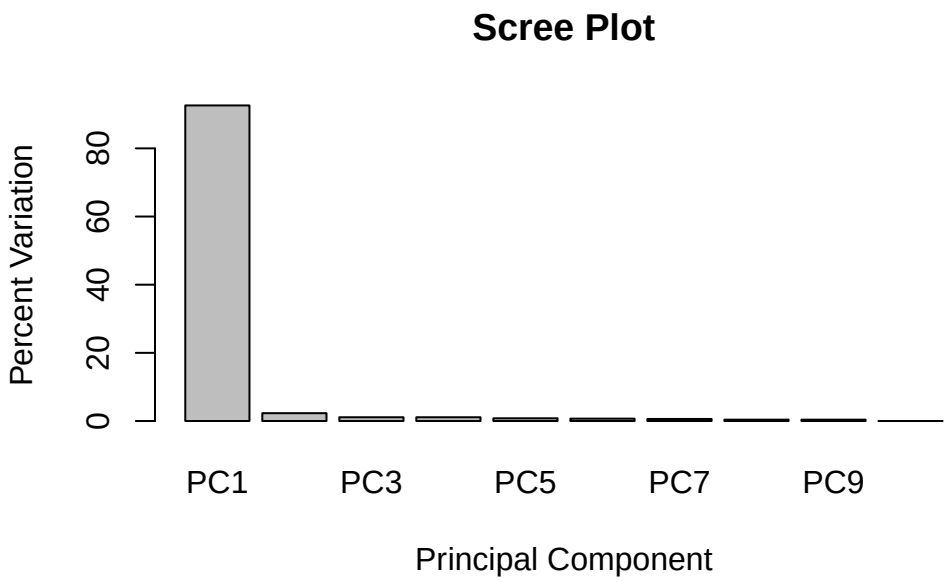



```
pca.var <- pca$sdev^2
```

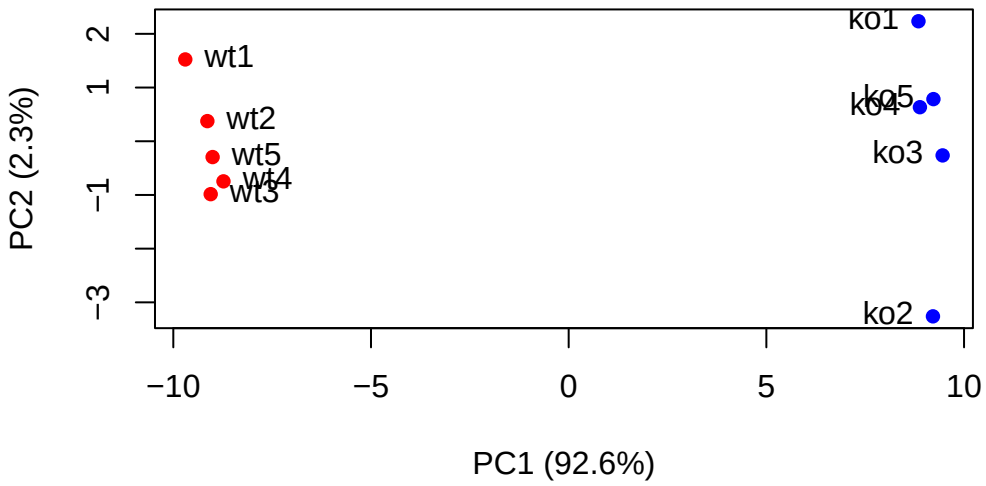
```
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)
pca.var.per
```

```
 [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")
```



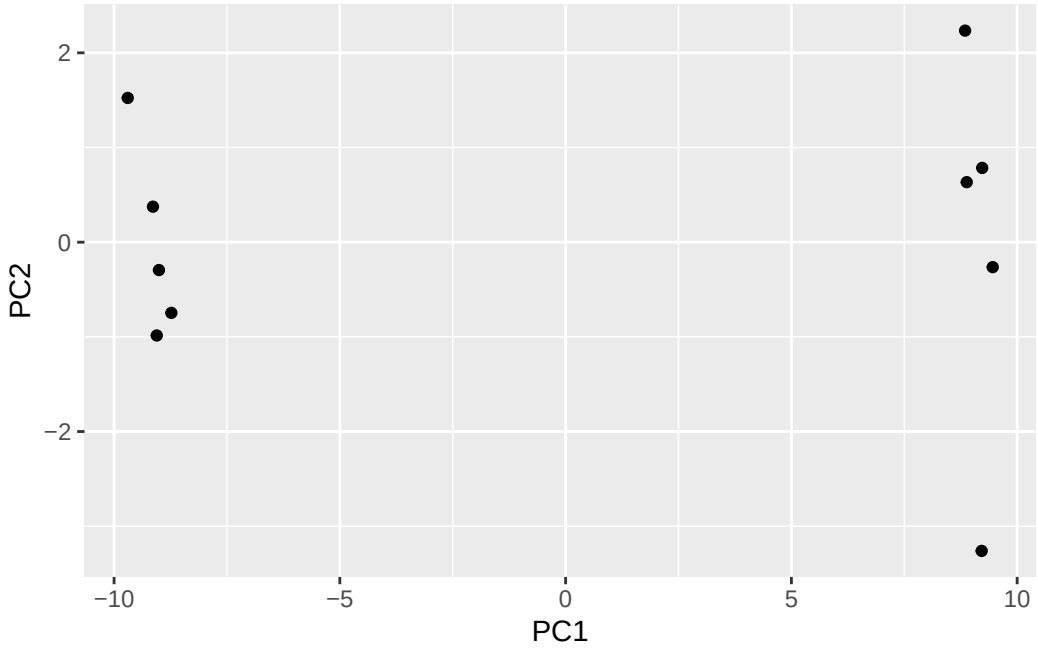

```
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

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

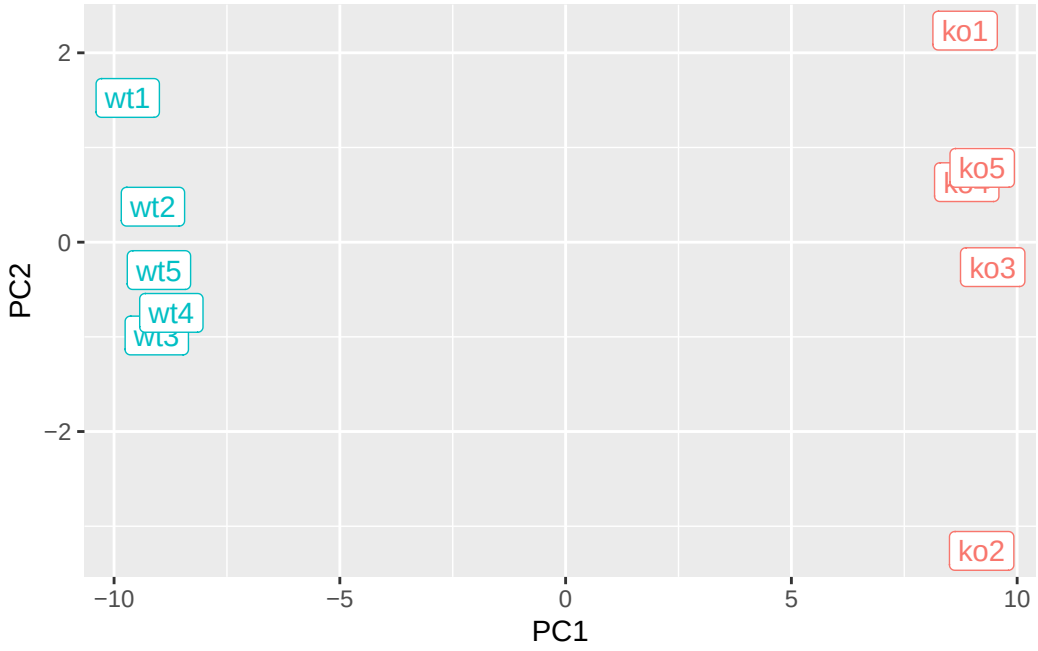
```
library(ggplot2)

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

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

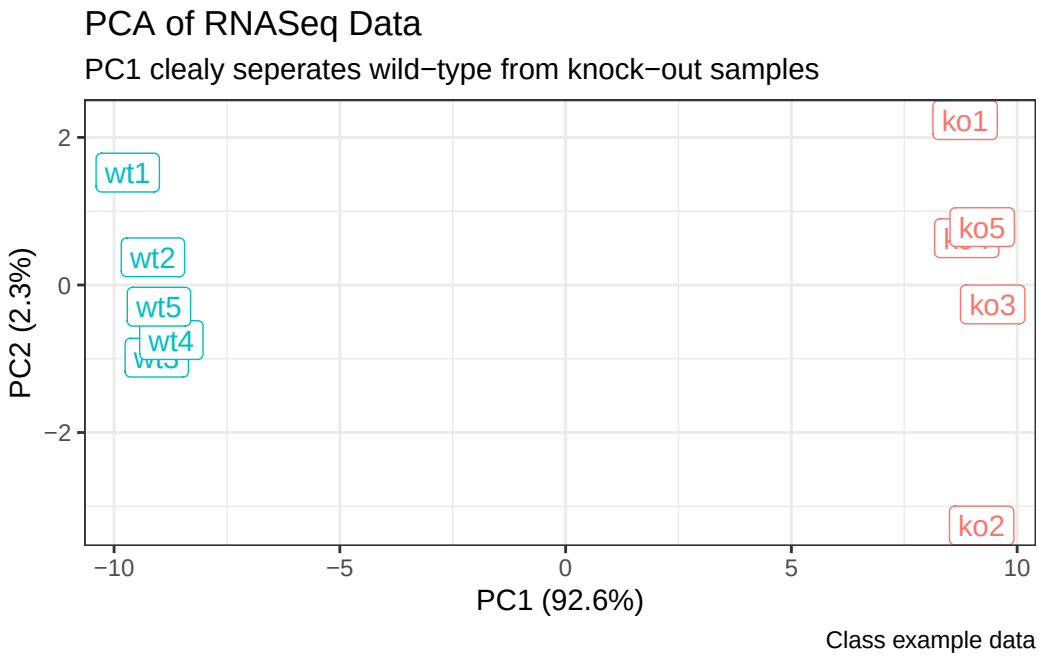



```
df$samples <- colnames(rna.data)
df$condition <- substr(colnames(rna.data),1,2)
```

```
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