Bimm 143: Machine Learning 1

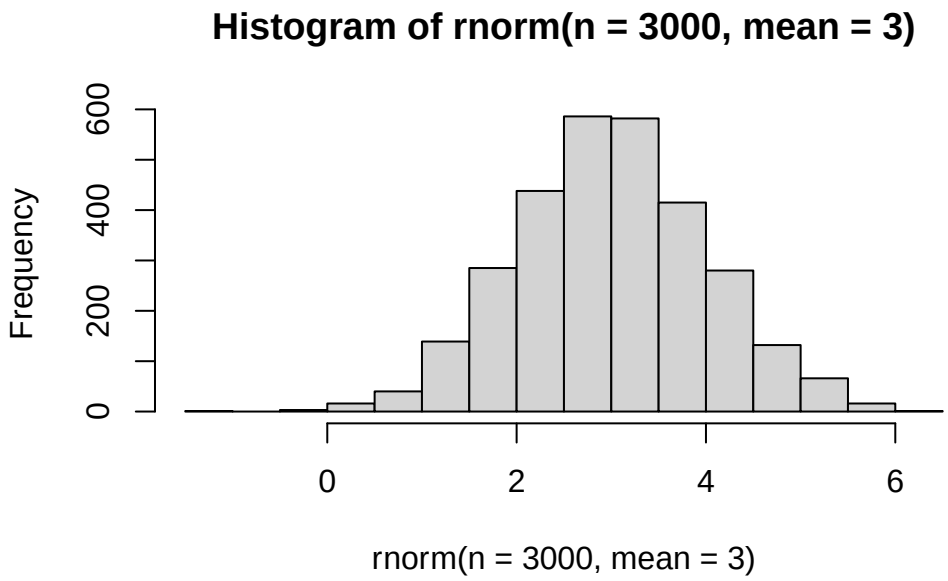
Joseph Elmaghraby (A16788229)

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

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

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

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

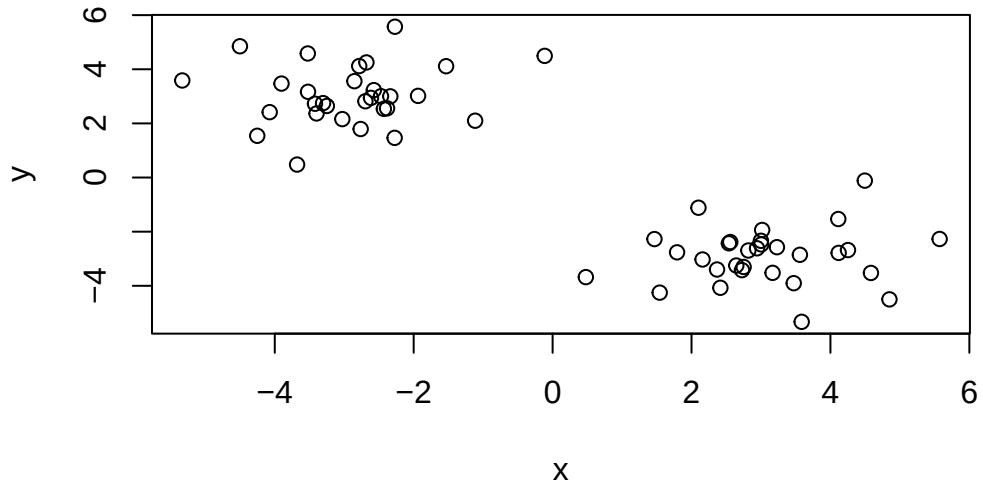


Make data with two "clusters"

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

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

```
             x        y
[1,] -3.524888 4.583250
[2,] -3.521287 3.167277
[3,] -4.252423 1.542165
[4,] -3.398492 2.367284
[5,] -2.680307 4.252441
[6,] -5.331534 3.586049
```

```r
plot(z)
```



How big is `z`

```r
nrow(z)
```

```
[1] 60
```

```
ncol(z)
```

```
[1] 2
```

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         y
1  3.043930 -2.900579
2 -2.900579  3.043930

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] 65.72816 65.72816
 (between_SS / total_SS =  89.0 %)

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

```
k$cluster
```

```
 [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
```

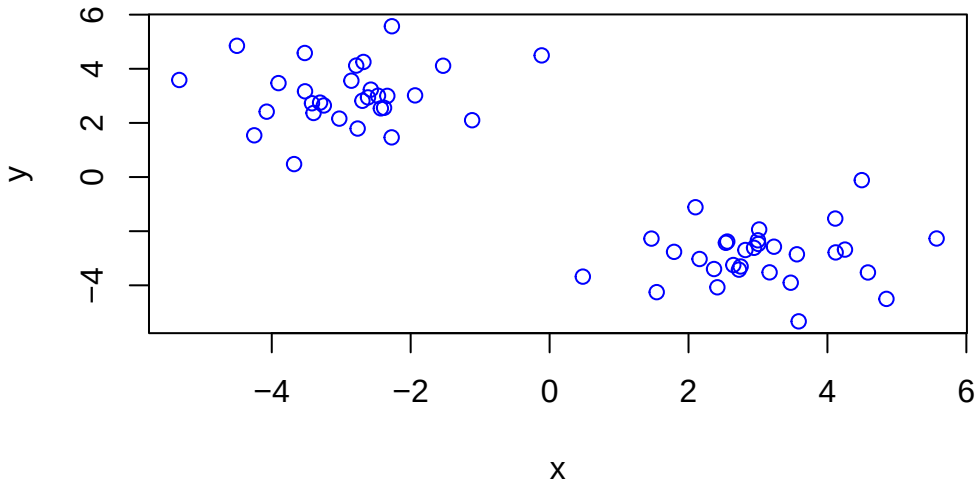
Q. Center of each cluster?

```
k$centers
```

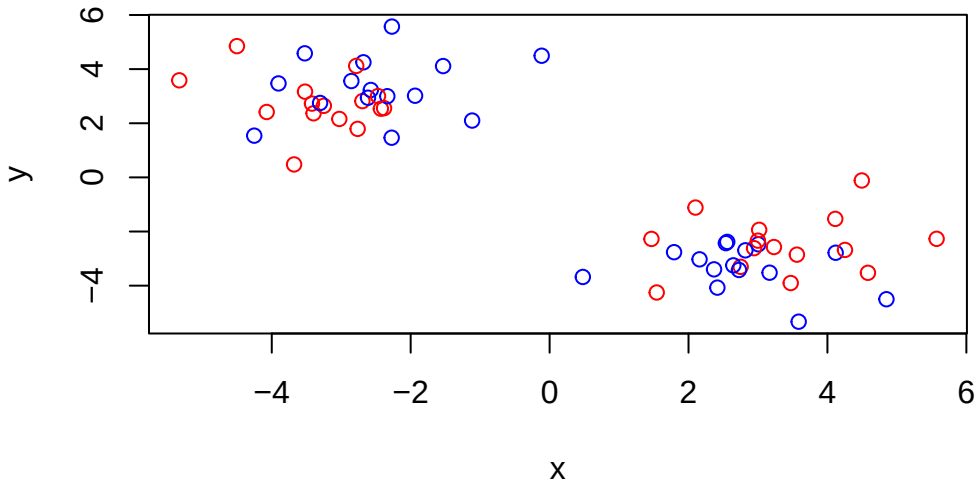
```
          x         y
1  3.043930 -2.900579
2 -2.900579  3.043930
```

Q. Put this result into together and make a little "base R" plot of cluster result.Also add the cluster centers points to this plot.

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

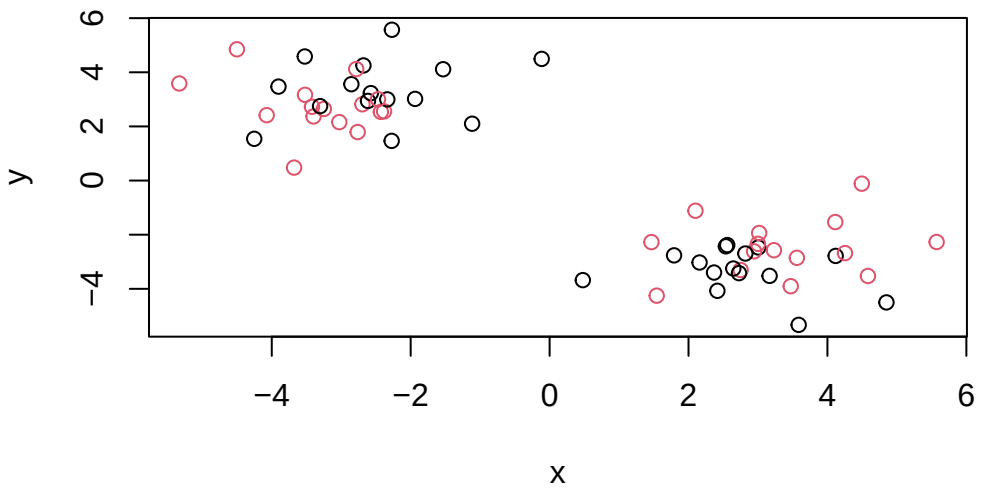



```
plot(z, col=c("blue", "red"))
```



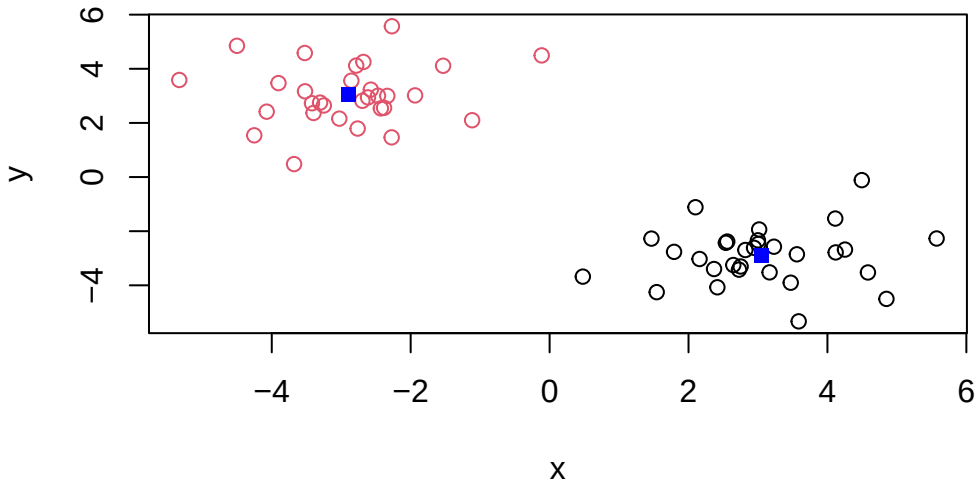
You can color by number.

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



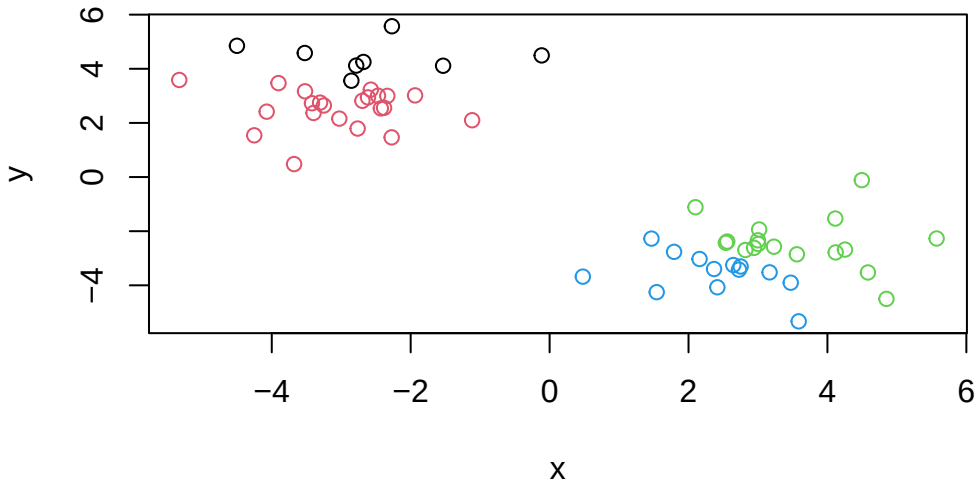
Plot by cluster memebership.

```
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 reslut visualization plot as above (plot of z cluster memebership).

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



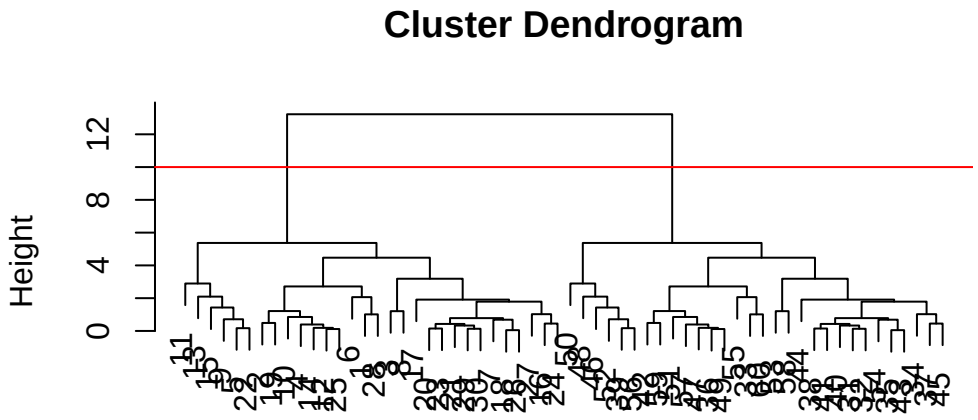
##Hierarchical Clustering

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

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

```
Call:
hclust(d = d)

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

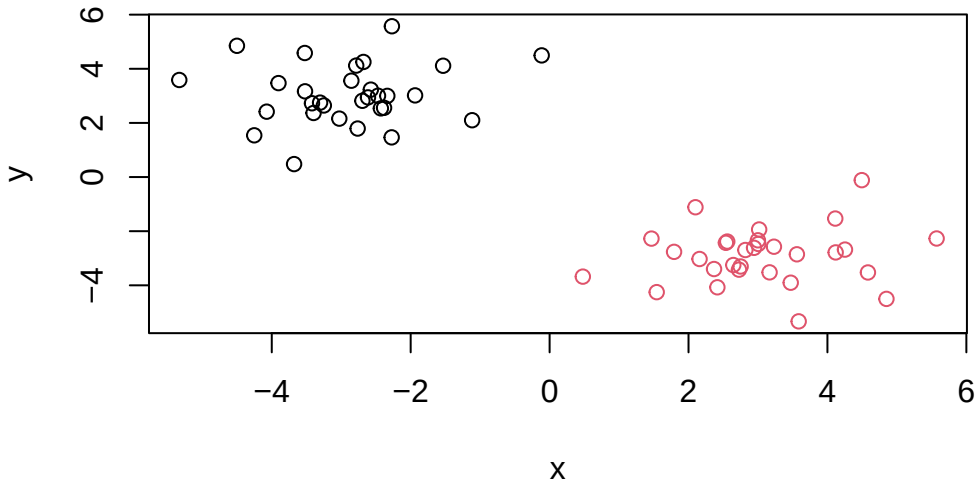
```
plot(hc)
abline(h=10, col="red")
```



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

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

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



##Hands on with Principal Component Analysis (PCA)

Let's examine some silly 17-dimensional data detailing food 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?

```
nrow(x)
```

```
[1] 17
```

```
ncol(x)
```

```
[1] 4
```

```
dim(x)
```

```
[1] 17  4
```

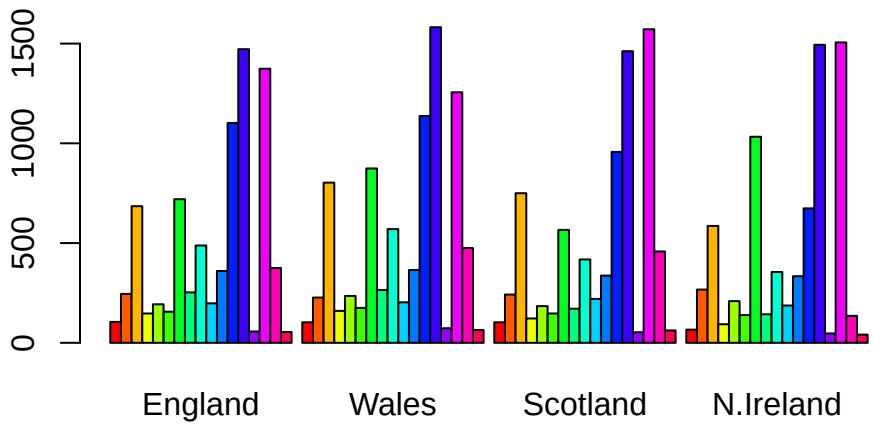
```
x <- read.csv(url, row.names=1)
head(x)
```

```
              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
```

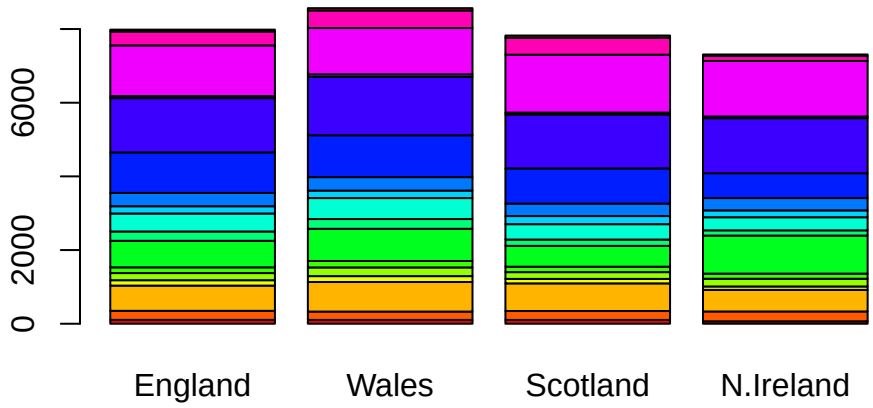
Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

Answer: Approach 2 is better because it sets the row name correct from the start.

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




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

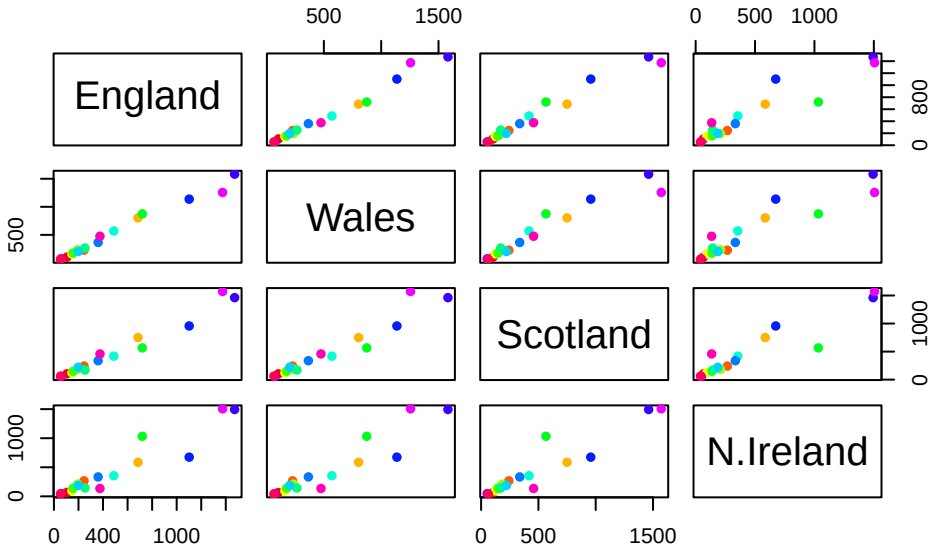


Q3: Changing what optional argument in the above barplot() function results in the following plot?

Answer: Changing the the argument `beside()` to False, allows the data to lay on top of each other instead of 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(nrow(x)), pch=16)
```



Answer: The more uniform the diagonal means that the countries are more alike in the statistics. The less uniform the diagonal the less alike.

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

Answer: N. Ireland consumes significantly less alcoholic beverages, fresh fruits, and other meats. They consume more fresh potatoes.

Looking at these types of "pairwise plots" can be helpful but it does not scale well and kind of sucks! There must be a better way....

PCA to the rescue!

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

Let's see what is in our PCA result object `pca`

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

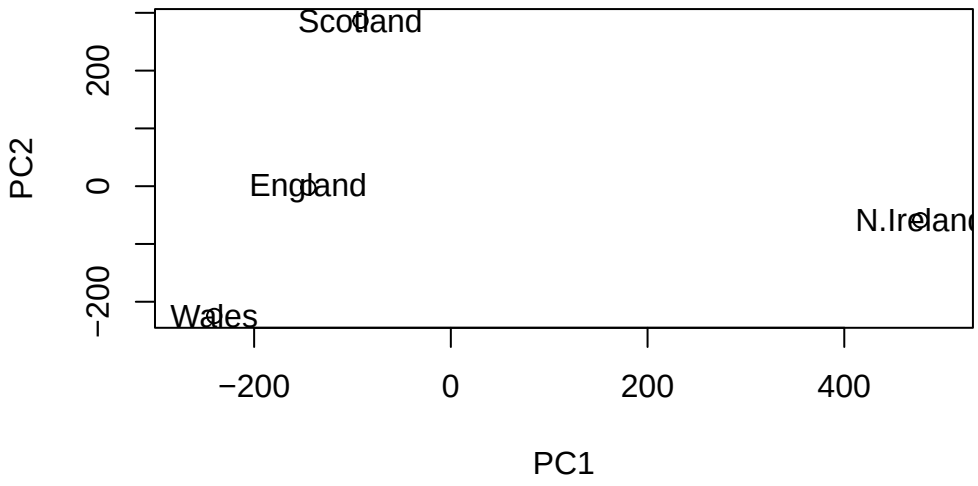
The `pca$x` result object is where we will focus first as this details how the countries are related to each other in terms of our new "axis" (a.k.a "PCs", "eigenvector", etc.)

```
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
```

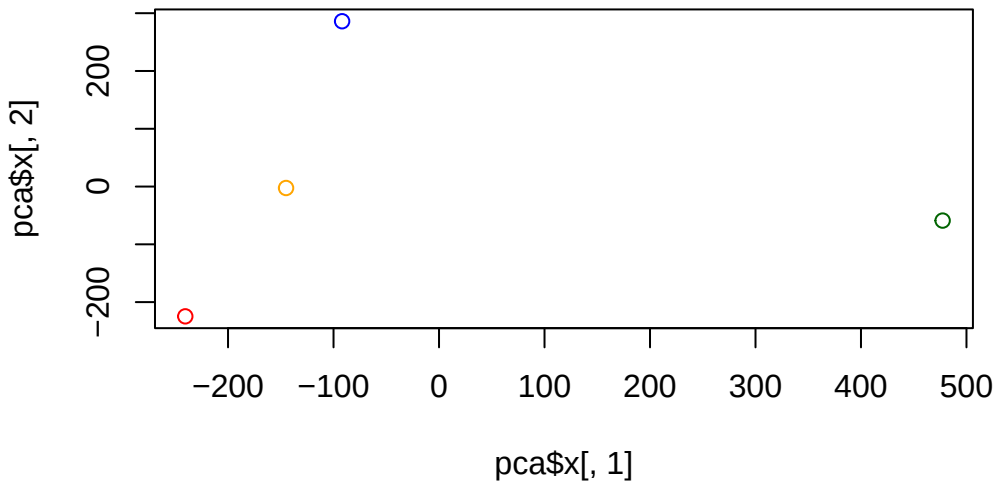
> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

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



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], col=c("orange","red", "blue", "darkgreen"))
```

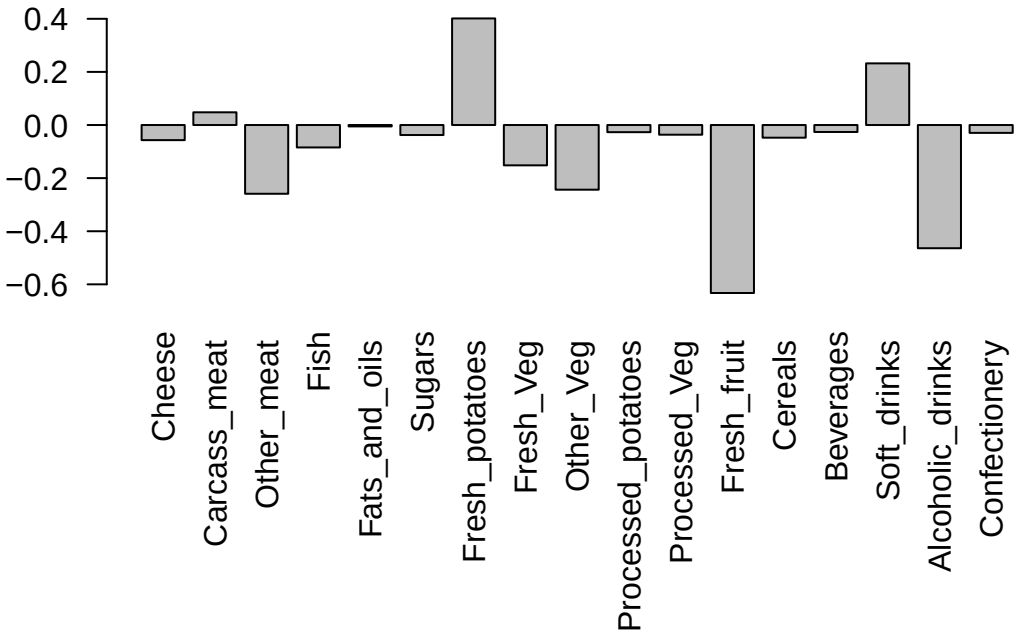


We can look at the so-called PC "loadings" result object to see how the original foods contribute to our new PCs (i.e. how the original variables contribute to our new better PC variables)

```
pca$rotation[,1]
```

```
          Cheese       Carcass_meat         Other_meat               Fish
-0.056955380         0.047927628       -0.258916658       -0.084414983
Fats_and_oils              Sugars     Fresh_potatoes          Fresh_Veg
-0.005193623       -0.037620983         0.401402060       -0.151849942
   Other_Veg  Processed_potatoes       Processed_Veg         Fresh_fruit
-0.243593729       -0.026886233       -0.036488269       -0.632640898
     Cereals           Beverages       Soft_drinks    Alcoholic_drinks
-0.047702858       -0.026187756         0.232244140       -0.463968168
Confectionery
-0.029650201
```

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

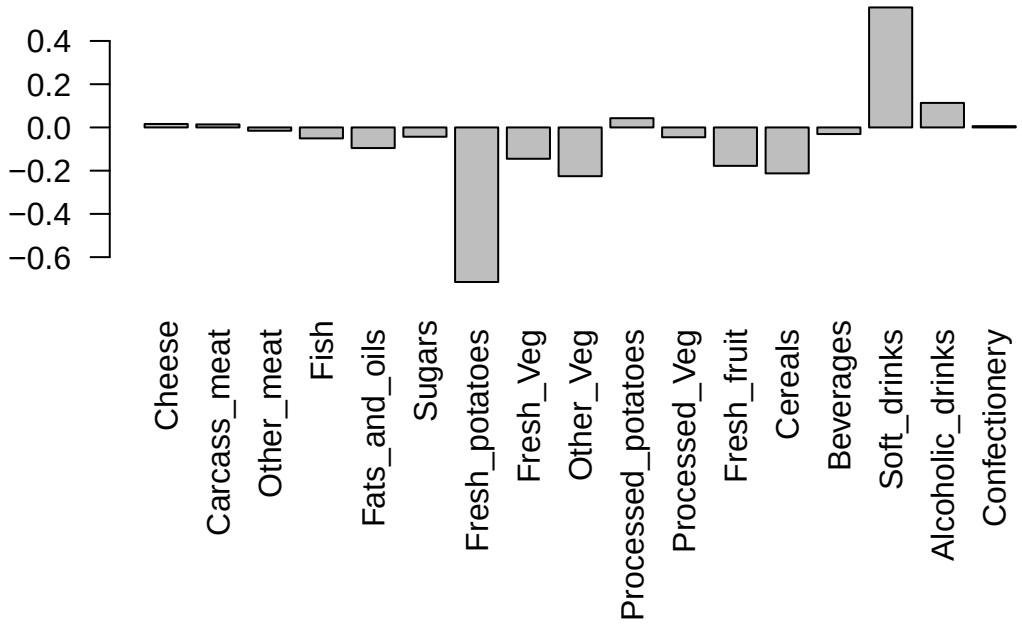


Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about?

```
pca$rotation[,2]
```

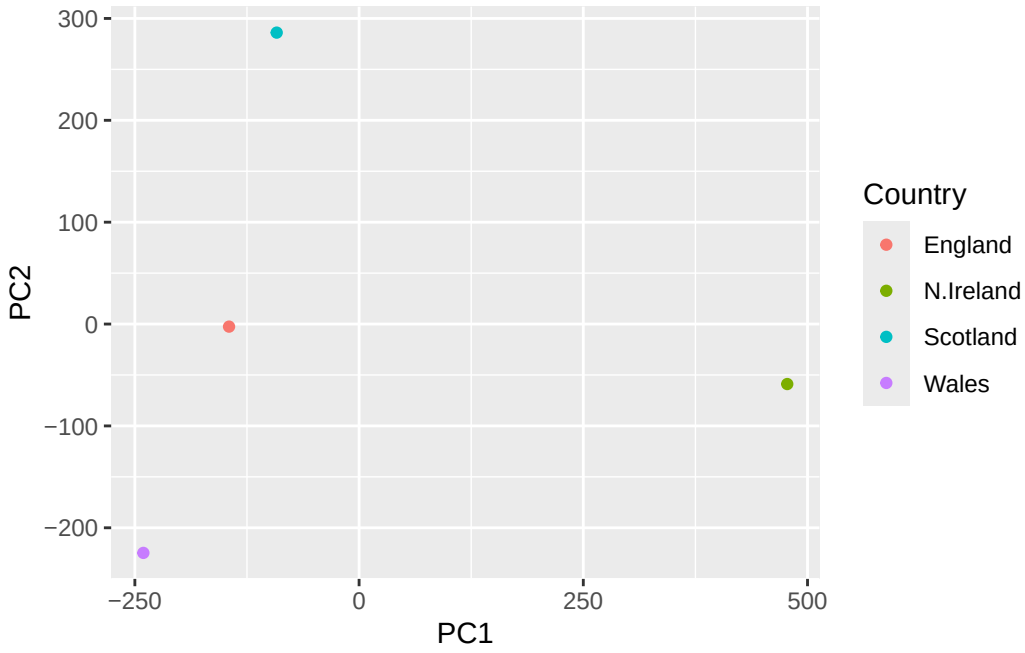
```
             Cheese       Carcass_meat         Other_meat               Fish
       0.016012850        0.013915823       -0.015331138       -0.050754947
    Fats_and_oils             Sugars    Fresh_potatoes          Fresh_Veg
      -0.095388656       -0.043021699       -0.715017078       -0.144900268
        Other_Veg Processed_potatoes      Processed_Veg        Fresh_fruit
      -0.225450923        0.042850761       -0.045451802       -0.177740743
           Cereals          Beverages       Soft_drinks   Alcoholic_drinks
      -0.212599678       -0.030560542        0.555124311        0.113536523
    Confectionery
       0.005949921
```

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

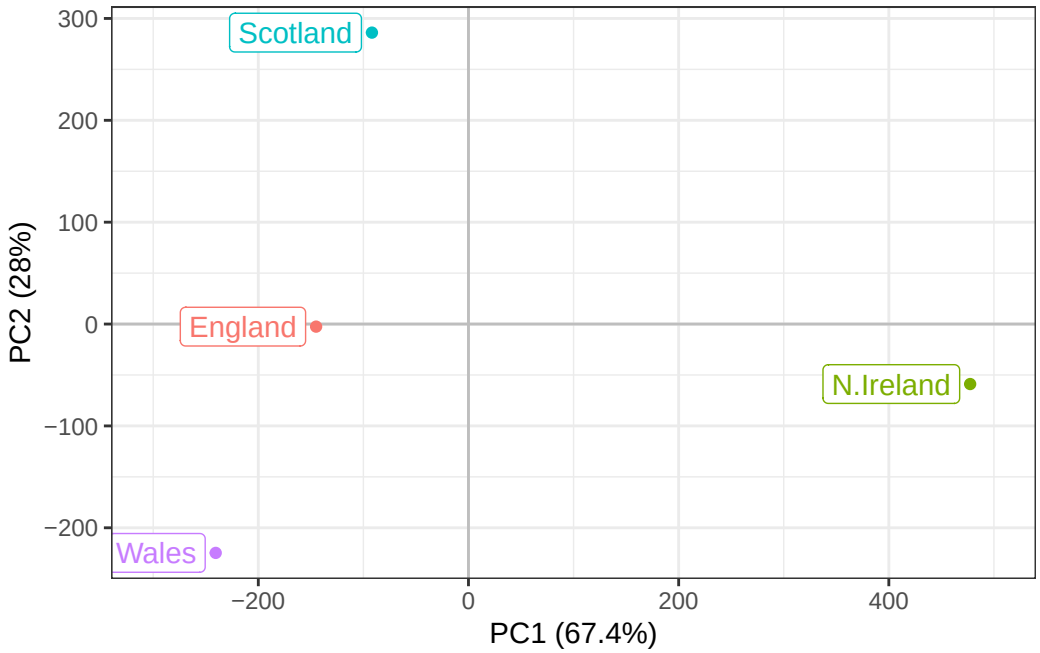


Answer: Fresh Potatoes and Soft drinks feature prominently.PC2 is telling you which values are most positive and most negative.

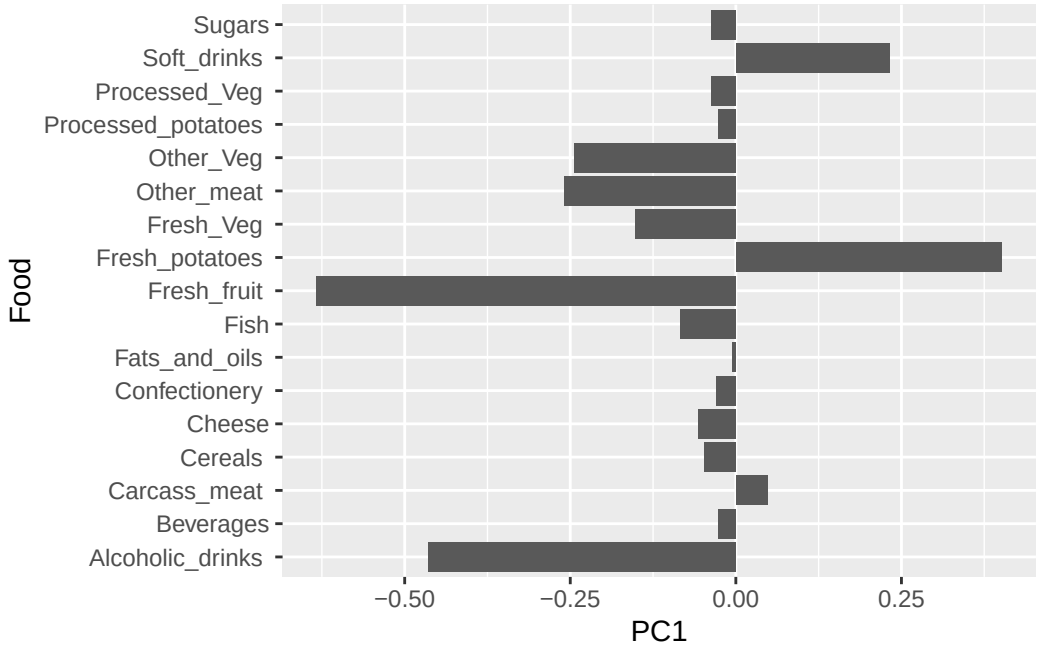
```
library(ggplot2)

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()
```



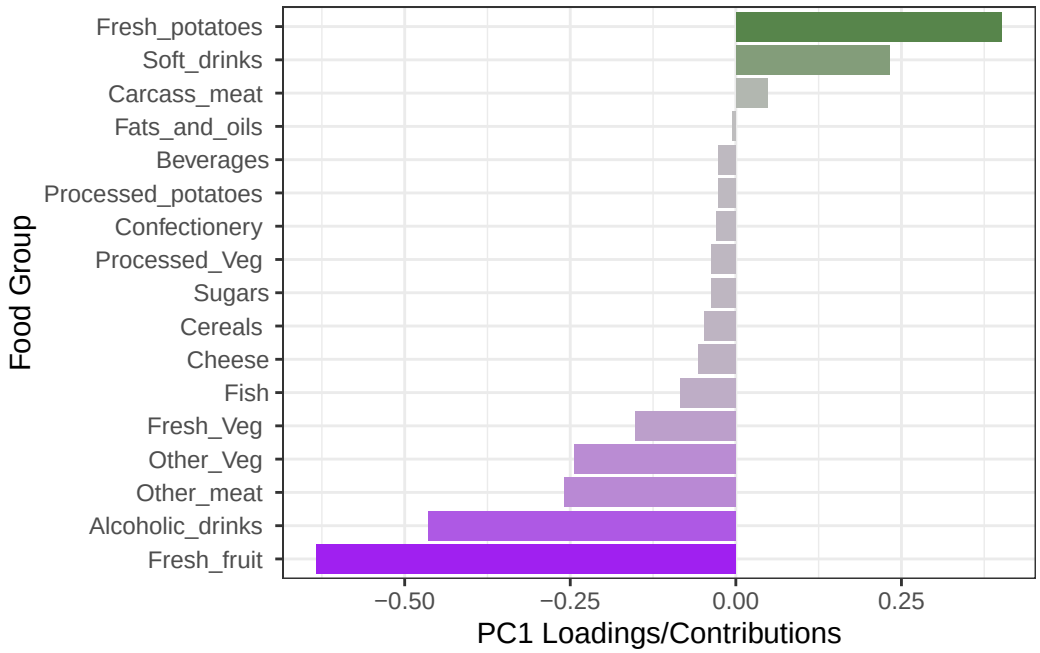

```
ggplot(df_lab) +
  aes(PC1, PC2, col=Country, label=Country) +
  geom_hline(yintercept = 0, col="gray") +
  geom_vline(xintercept = 0, col="gray") +
  geom_point(show.legend = FALSE) +
  geom_label(hjust=1, nudge_x = -10, show.legend = FALSE) +
  expand_limits(x = c(-300,500)) +
  xlab("PC1 (67.4%)")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



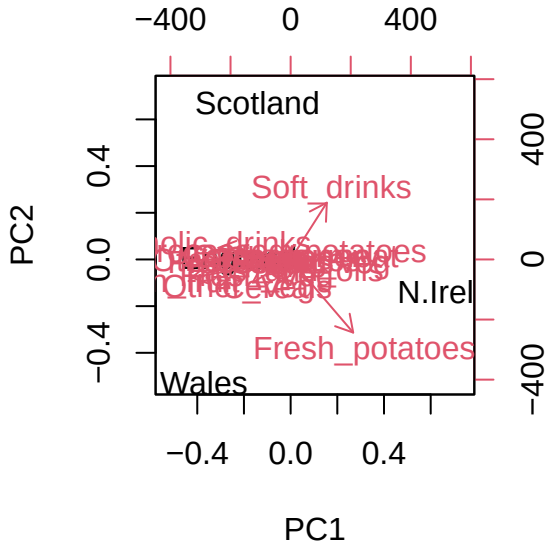

```
ld <- as.data.frame(pca$rotation)
ld_lab <- tibble::rownames_to_column(ld, "Food")

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()
```




```
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
```

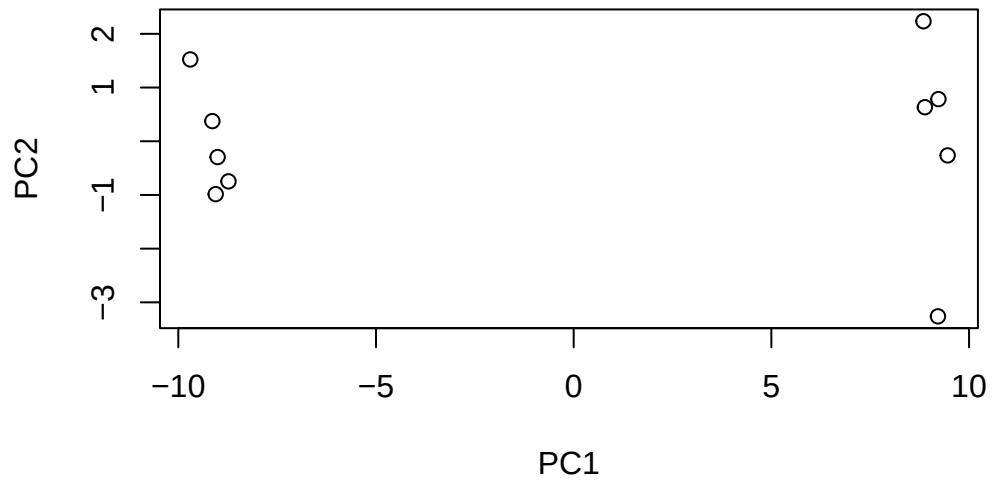
```
dim(rna.data)
```

```
[1] 100  10
```

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

Answer: 100 genes, and 10 samples.

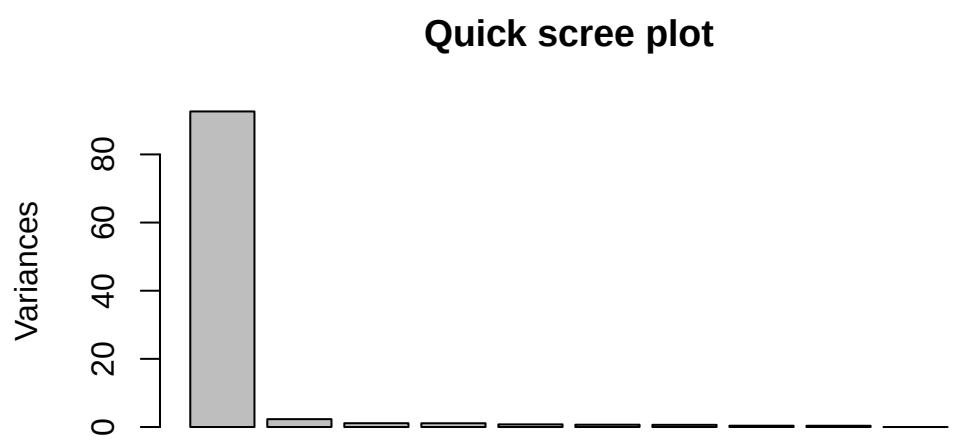
```
## Again we have to take the transpose of our data
pca <- prcomp(t(rna.data), scale=TRUE)

## 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 3.345e-15
Proportion of Variance 0.00385 0.00364 0.000e+00
Cumulative Proportion  0.99636 1.00000 1.000e+00
```

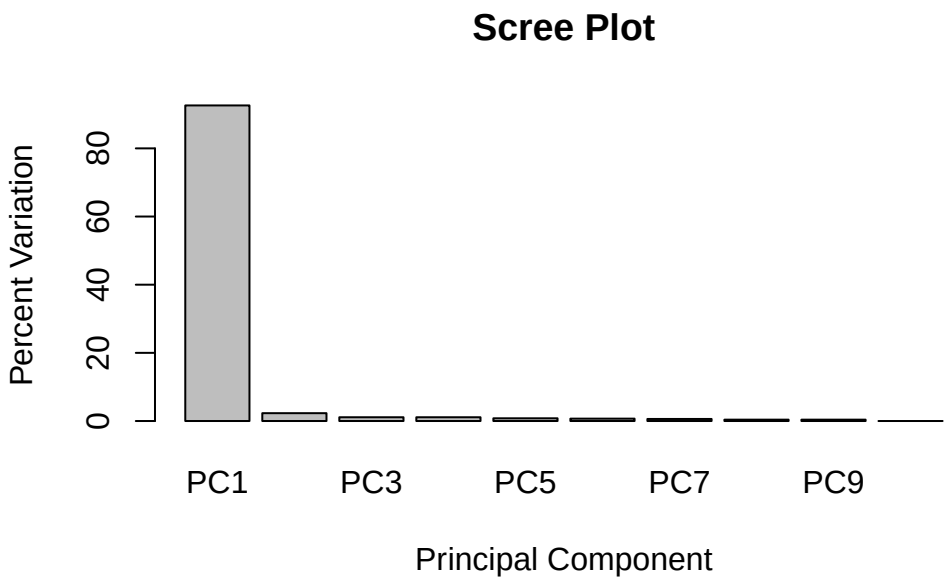
```
plot(pca, main="Quick scree plot")
```



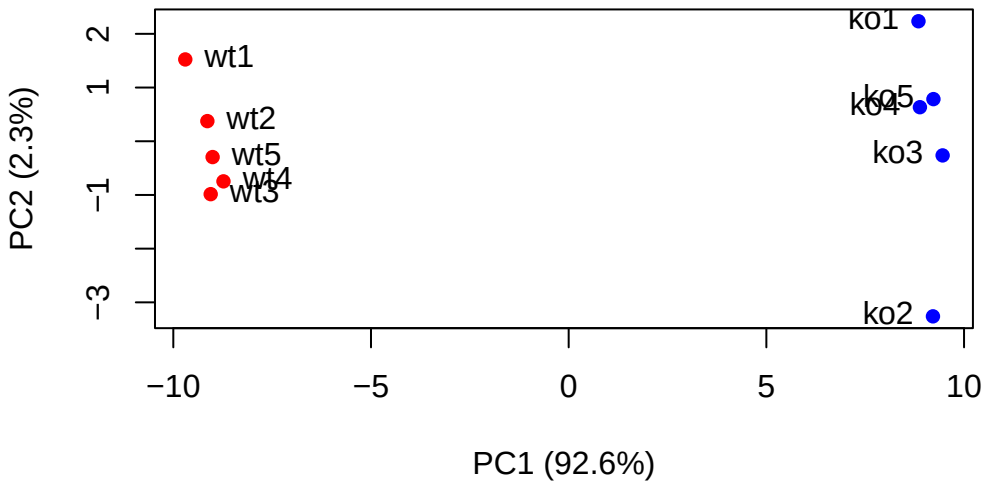

```
## Variance captured per PC
pca.var <- pca$sdev^2

## 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
```

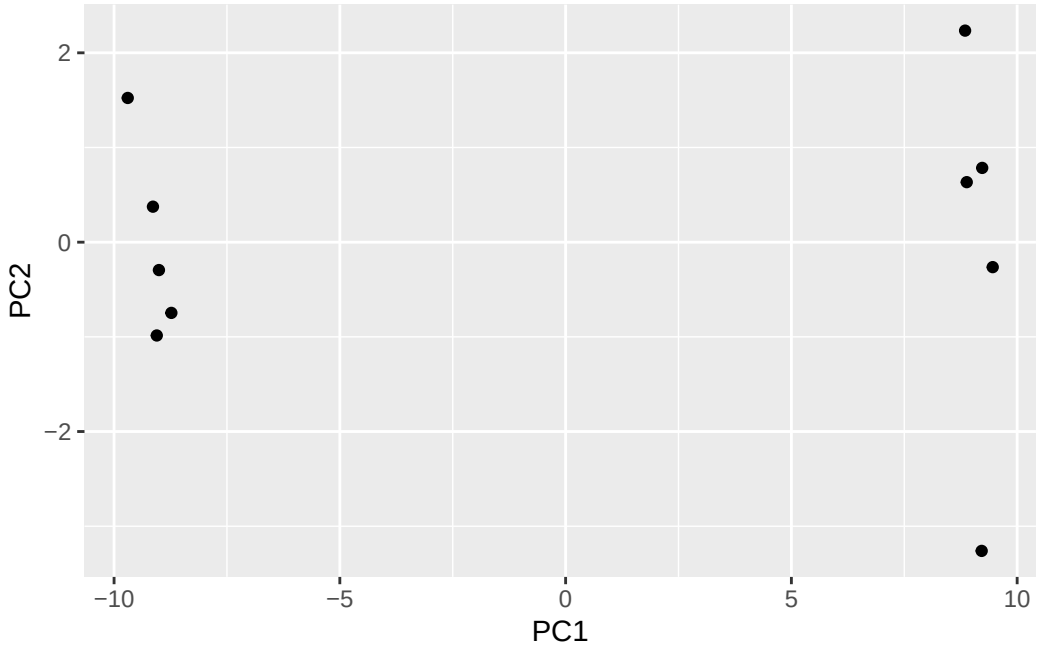
```
barplot(pca.var.per, main="Scree Plot",
        names.arg = paste0("PC", 1:10),
        xlab="Principal Component", ylab="Percent Variation")
```



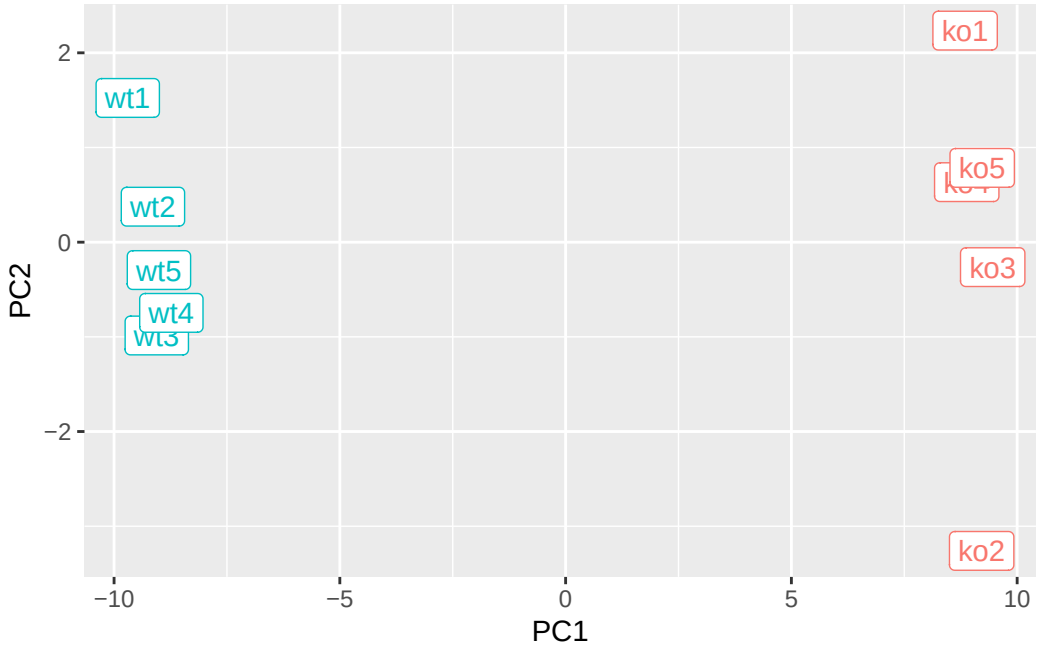

```
## A vector of colors for wt and ko samples
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

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)))
```



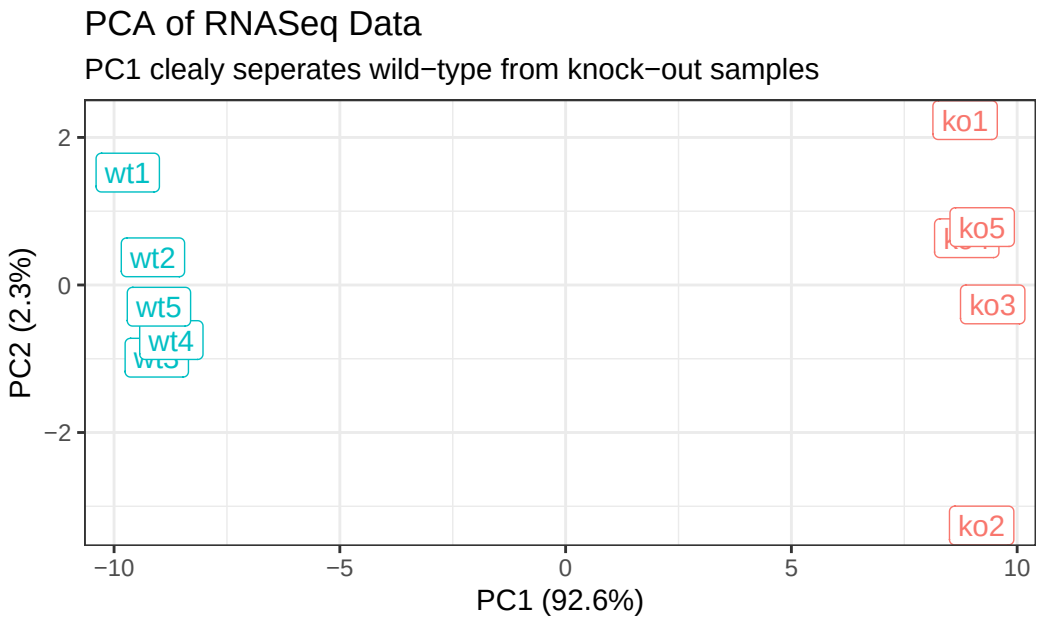

```
library(ggplot2)

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()
```




```
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"
```