Class 7: Machine Learning 1

## Keilyn Duarte (PID A16881868)

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

Let's start by making up 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 = 1))
```
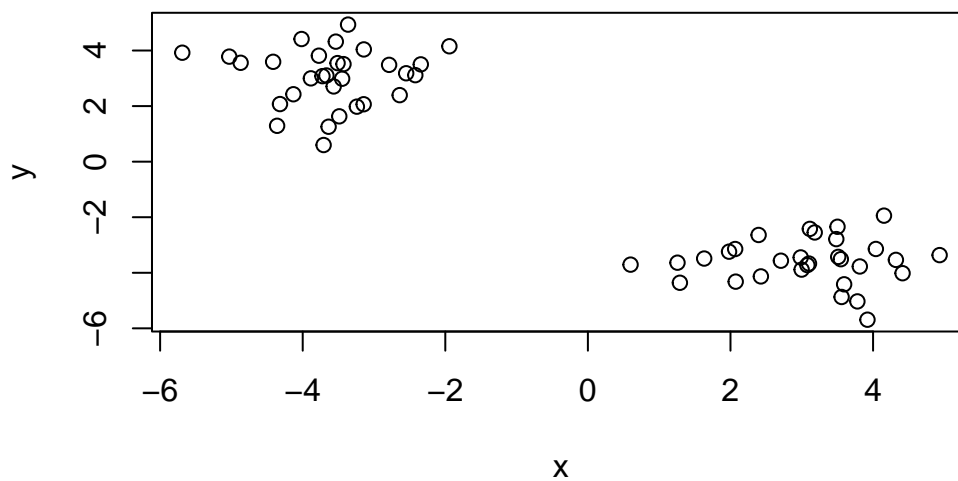
Histogram of rnorm(n = 3000, mean = 1)
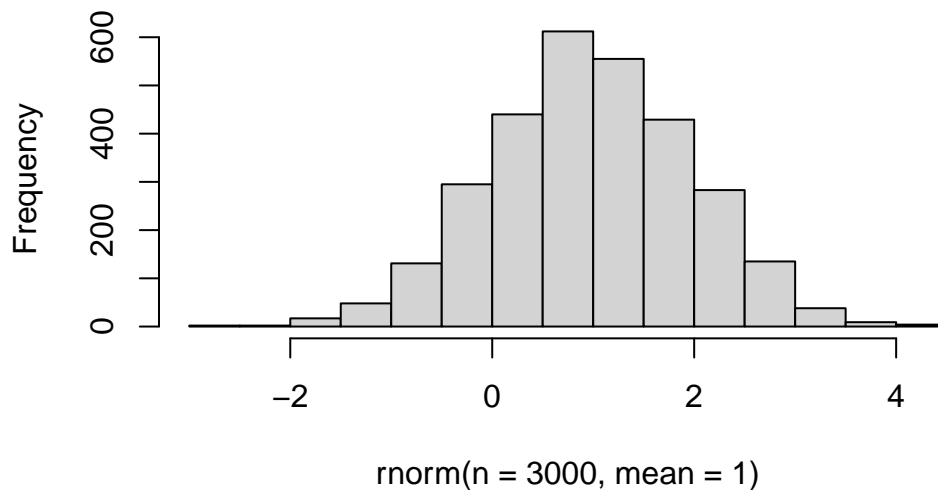
Make data `z` with two "clusters"

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

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

```
             x         y
[1,] -1.942833 4.1529787
[2,] -3.510941 3.5477224
[3,] -2.639522 2.3946226
[4,] -3.667690 3.1013995
[5,] -3.142930 4.0392090
[6,] -3.706699 0.5985832
```

```
plot(z)
```


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 -3.592354   3.048514
2 3.048514 -3.592354

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 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] 50.32824 50.32824
 (between_SS / total_SS = 92.9 %)

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 lies 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 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
```
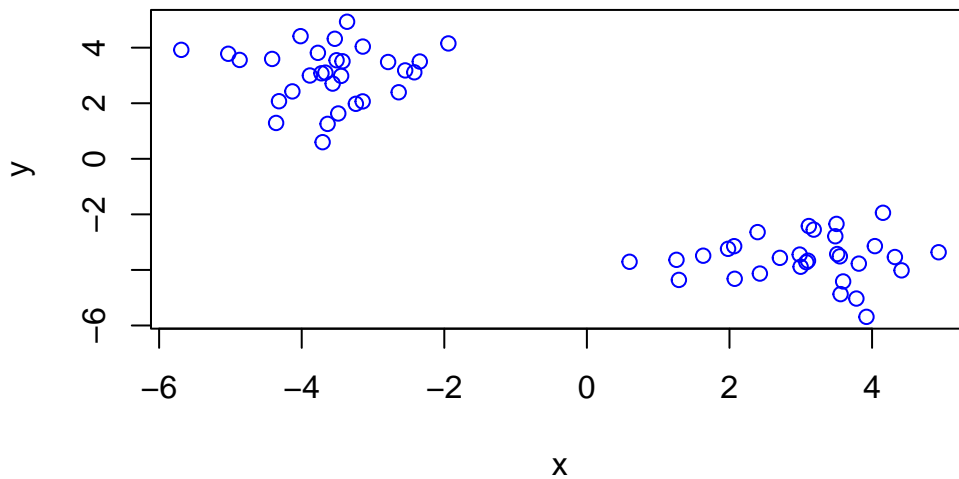
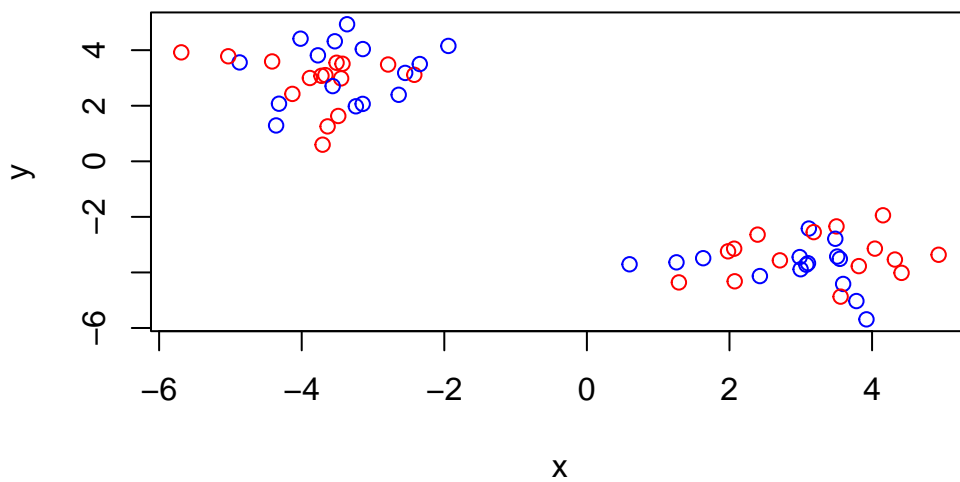Q. Center of each cluster?

```
k$centers
```

```
          x          y
1 -3.592354  3.048514
2  3.048514 -3.592354
```

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

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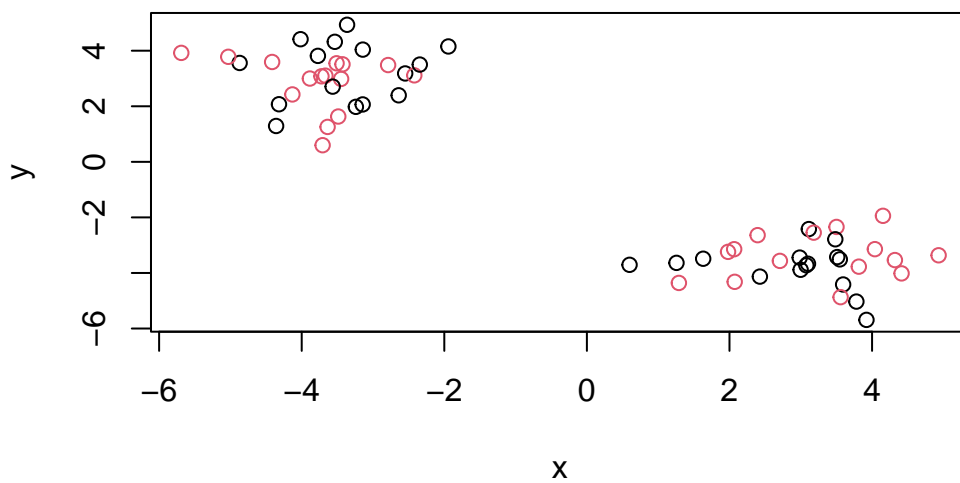

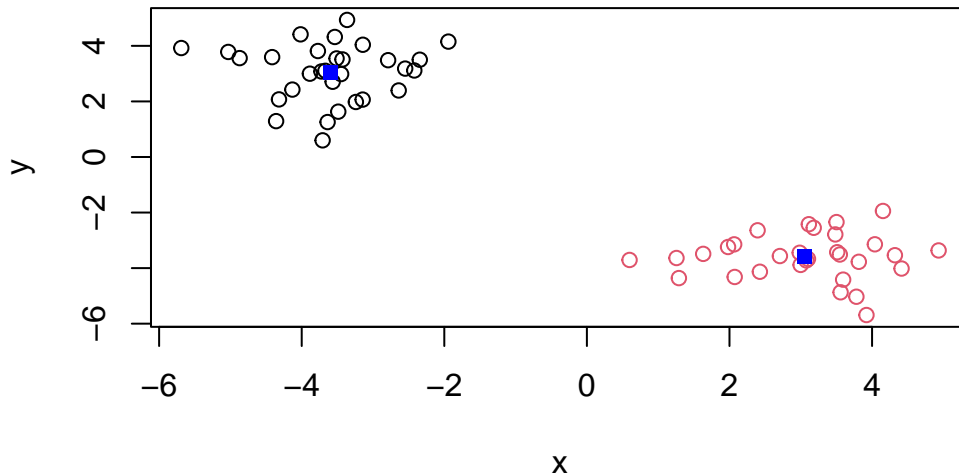```
plot(z, col = c("blue","red"))
```

You can color by number.

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

Plot 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 visualization plot as above (pot of z colored by cluster membership)

```
k4 <- kmeans(z, centers = 4)
k4
```

```
K-means clustering with 4 clusters of sizes 12, 12, 18, 18

Cluster means:
          x         y
1  3.834433 -3.067853
2 -3.067853  3.834433
3 -3.942021  2.524569
4  2.524569 -3.942021

Clustering vector:
 [1] 2 2 3 3 2 3 2 3 3 2 3 3 2 2 3 2 2 3 2 3 2 3 2 3 3 3 3 3 3 3 4 4 4 4 4 4 4 1
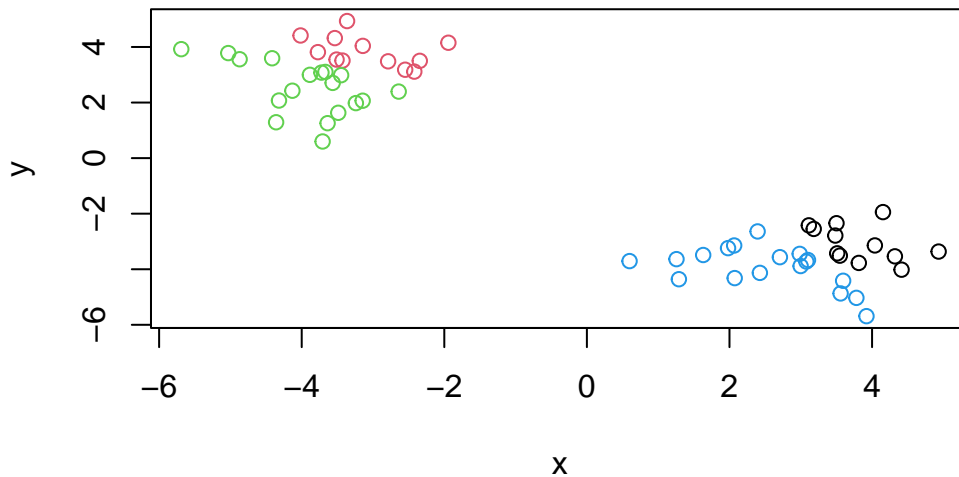```

```
[39] 4 1 4 1 4 1 1 4 1 1 4 4 1 4 4 1 4 1 4 4 1 1

Within cluster sum of squares by cluster:
[1]  7.881512  7.881512 24.591355 24.591355
 (between_SS / total_SS =  95.4 %)

Available components:

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

```r
plot(z, col = k4$cluster)
```



```r
k4$totss
```

```
[1] 1423.69
```

Hierarchical Clustering

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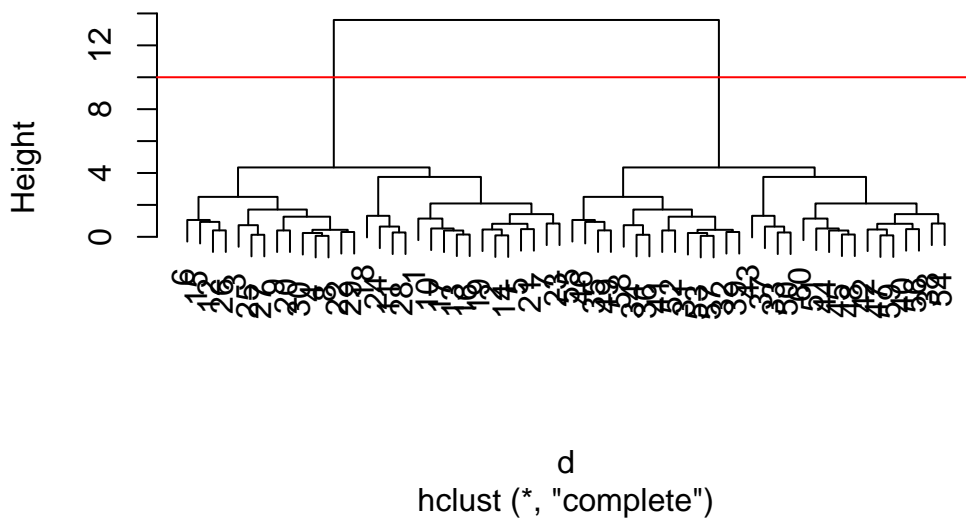
```
d <- dist(z)
hc <- hclust(d)
hc
```

```
Call:
hclust(d = d)

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

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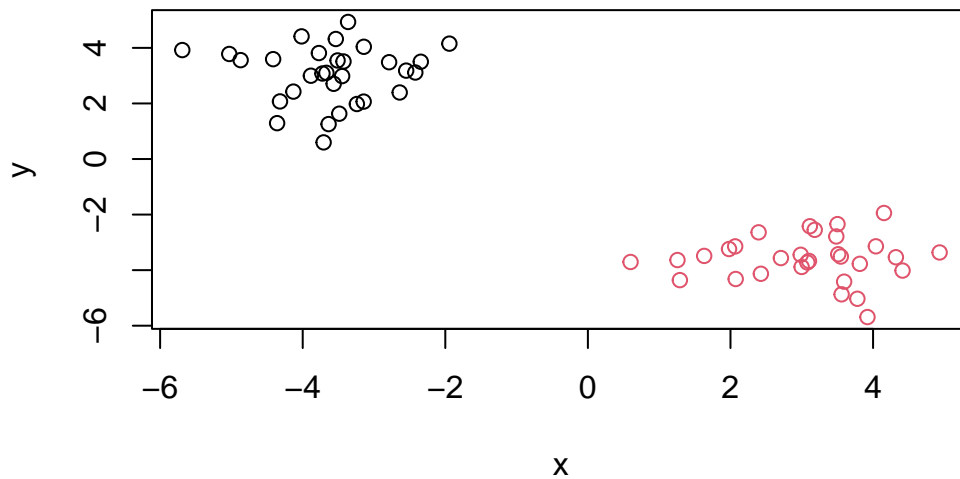```

## Cluster Dendrogram



d
hclust (*, "complete")

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

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

```r
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.

Data import

```r
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

9

```
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 question?

```
nrow(x)
```

```
[1] 17
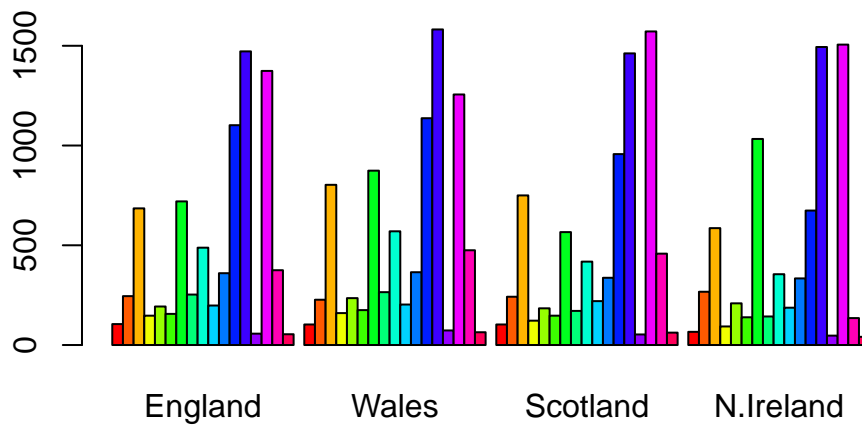```

```
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?

I prefer editing the data import code because it's more simple than the other option, which adds more lines of code overall.
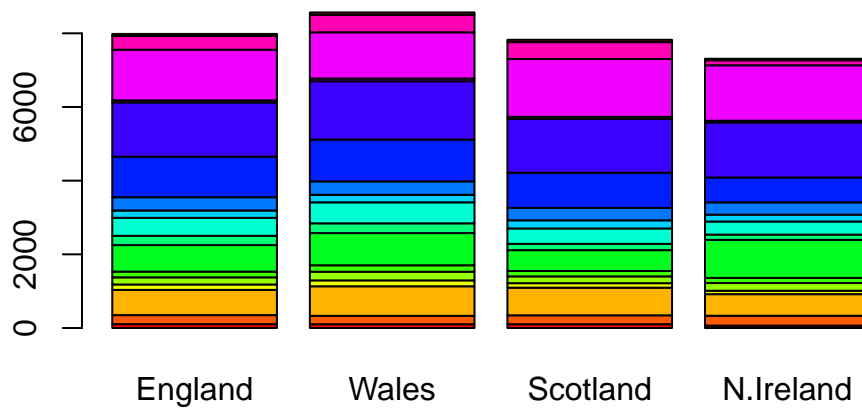
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```

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

Setting the beside argument to F results in this plot, where the different foods are stacked on top of eachother.

```
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?

```
pairs(x, col=rainbow(nrow(x)), pch=16)
```
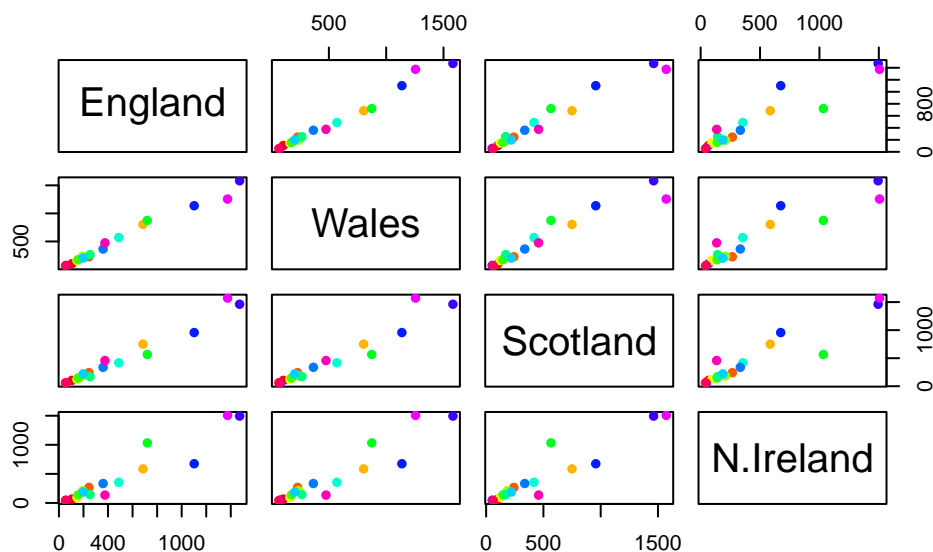
Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set?

People in N. Ireland eat more potatoes and less fresh fruit than other countries in the UK.

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 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
```

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 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. 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], pch=16, col = c("orange","red","blue","darkgreen"), xlab="PC1", yl
```

14
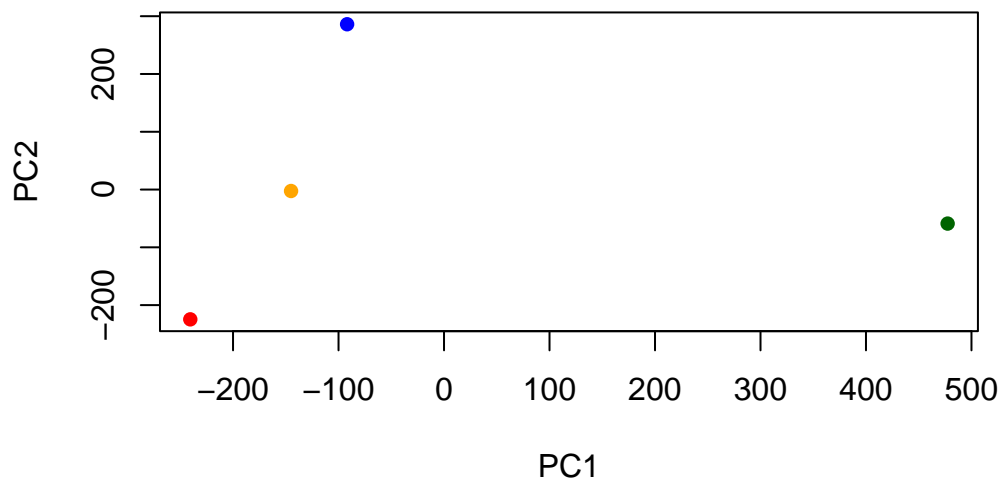
We can look at the so-called PC "loadings" result object to see how the original foods contribute to our new PCs (i.e. how the original variables contribute to our new, better PC variables).

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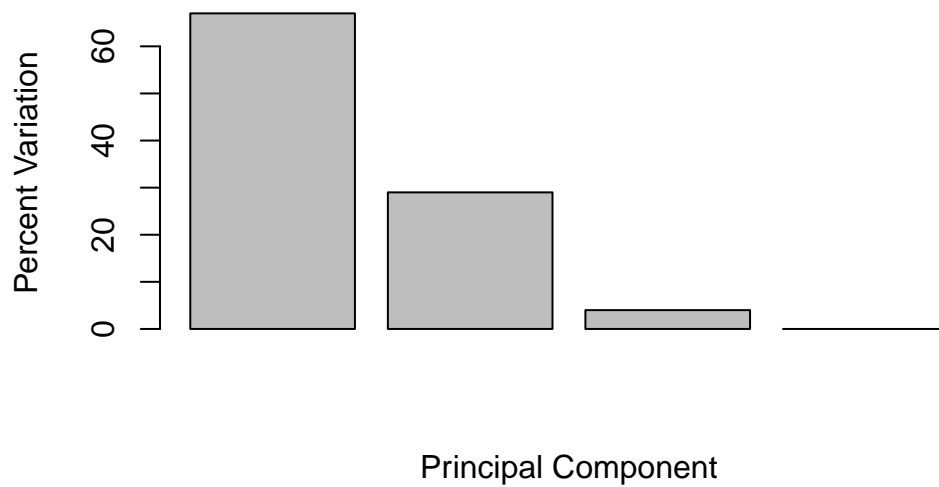
```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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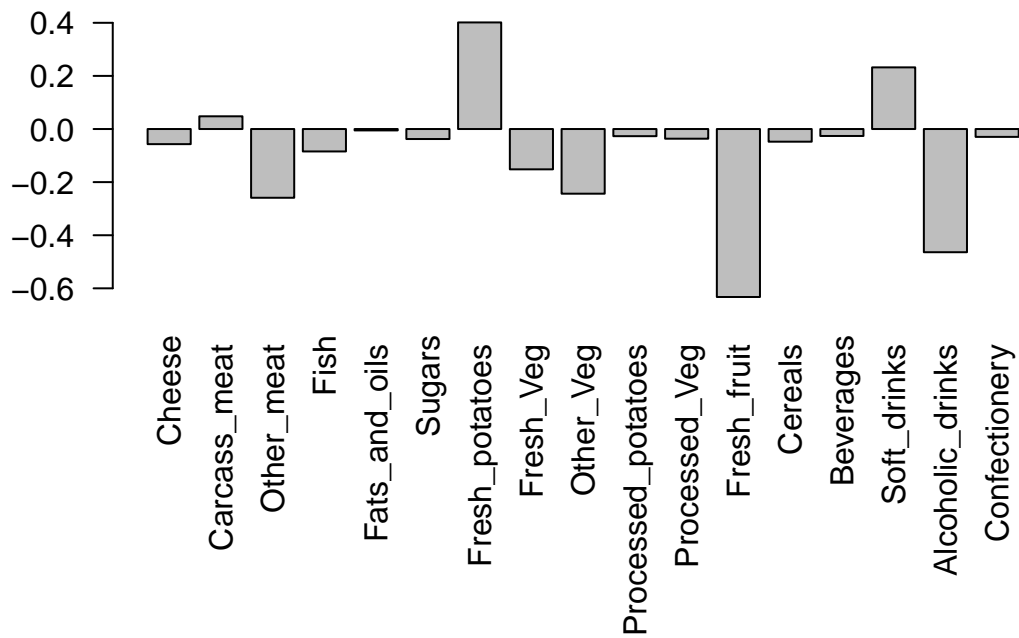
```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```
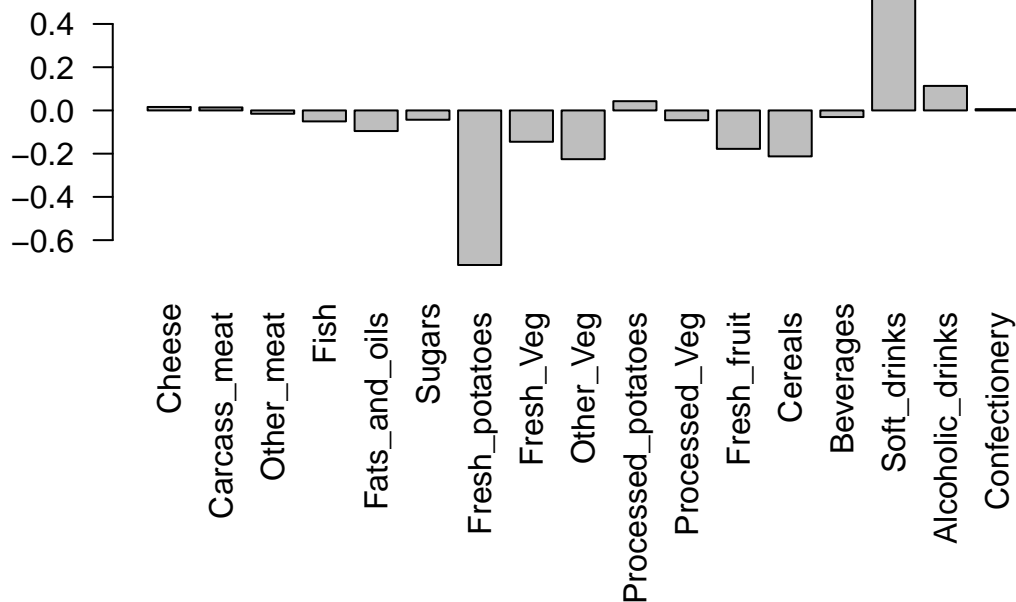

Digging deeper (variable loadings)

```
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))
barplot( pca$rotation[,2], las=2 )
```

PC2 gives us information from an additional 29% of the data that contribute to N. Ireland's position on the PCA plot

Using ggplot for these figures

```
library(ggplot2)

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

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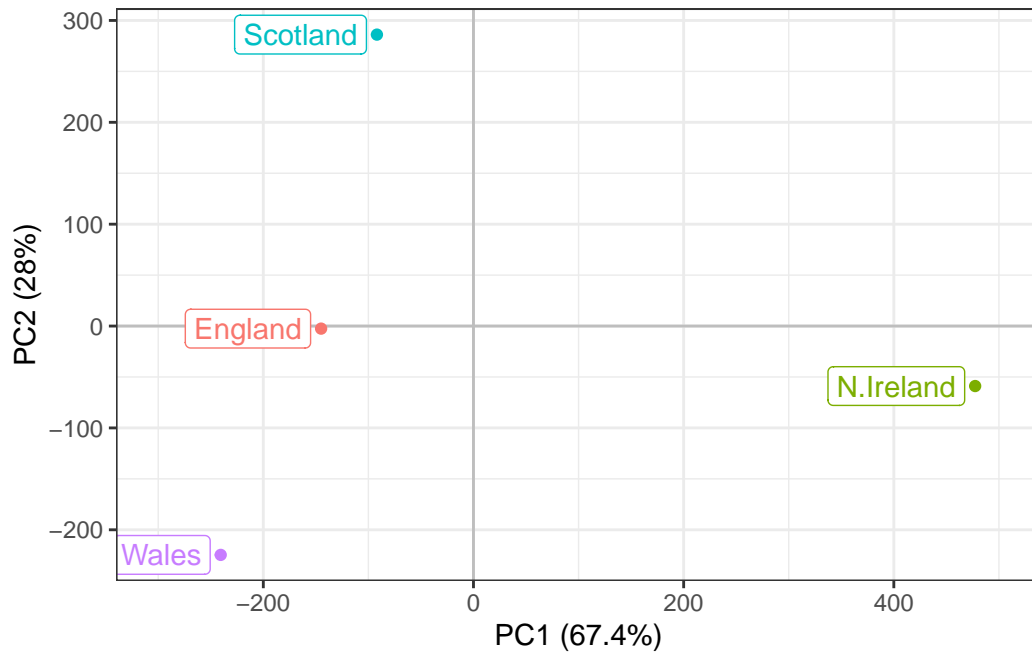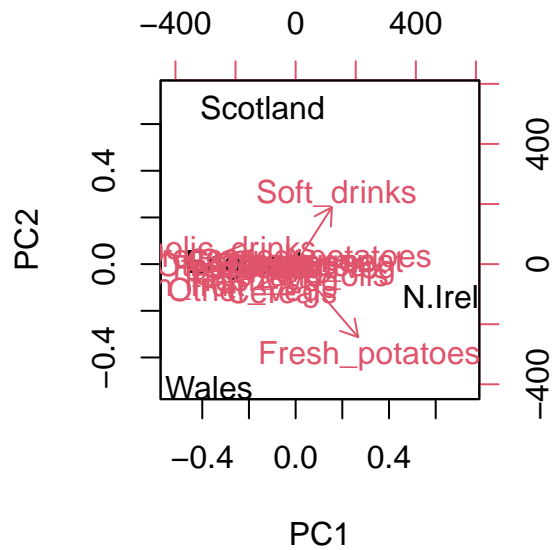
```r
ld <- as.data.frame(pca$rotation)
ld_lab <- tibble::rownames_to_column(ld, "Food")

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()
```
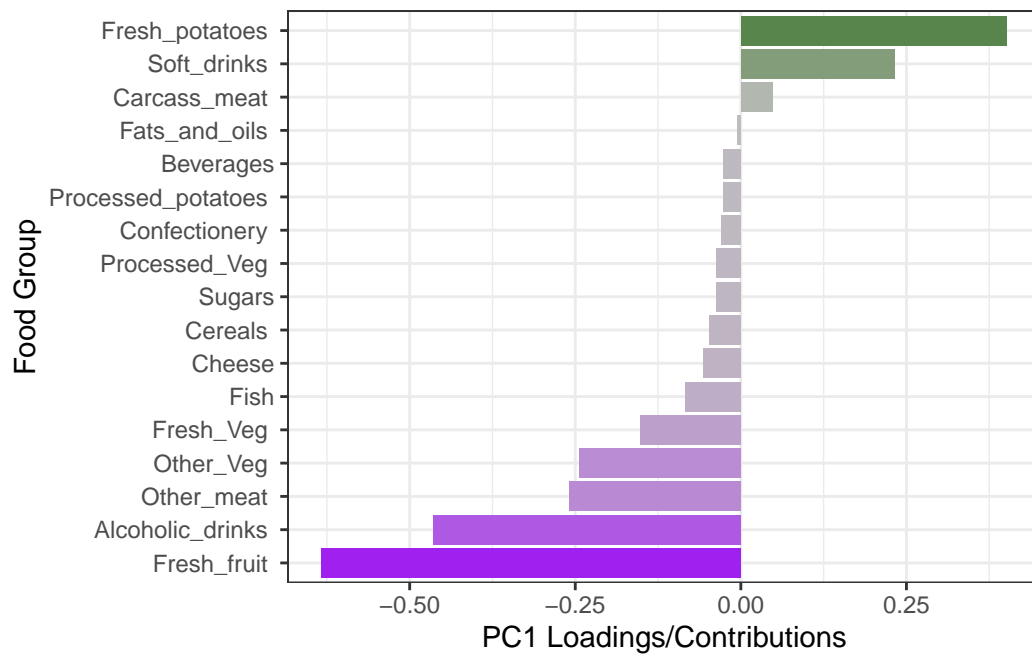
Biplots

```
biplot(pca)
```

PCA of RNA-seq data

```r
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?
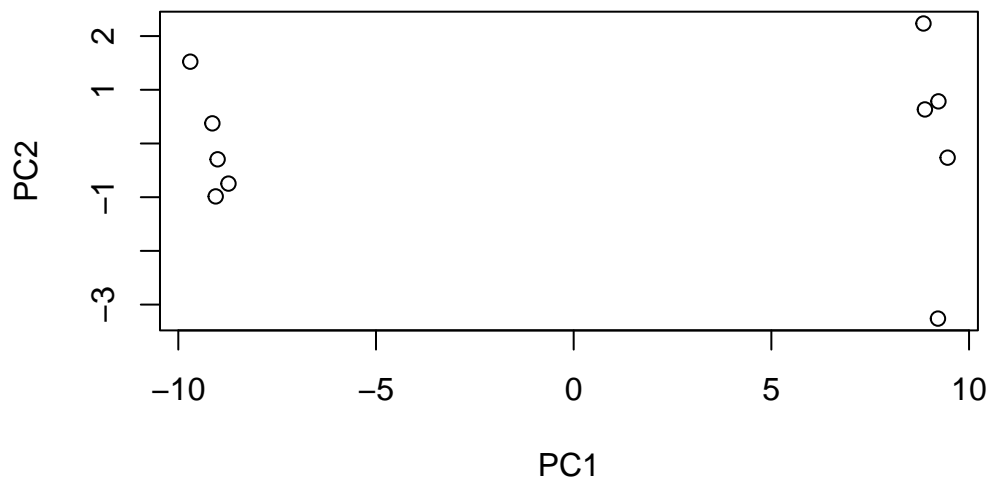
```r
ncol(rna.data)
```

```
[1] 10
```

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

```
[1] 100
```

There are 100 genes (# of rows) and 10 samples (# of columns) in this data set.

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

## 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
```

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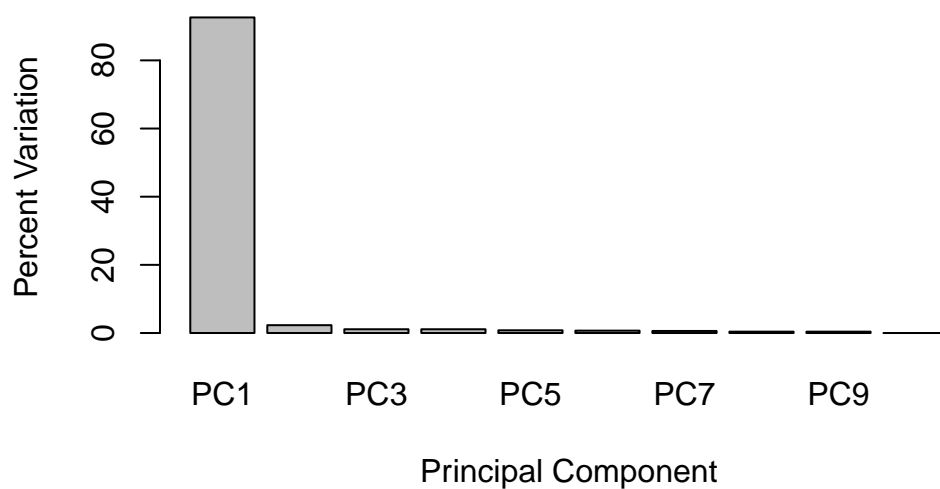
Quick scree plot
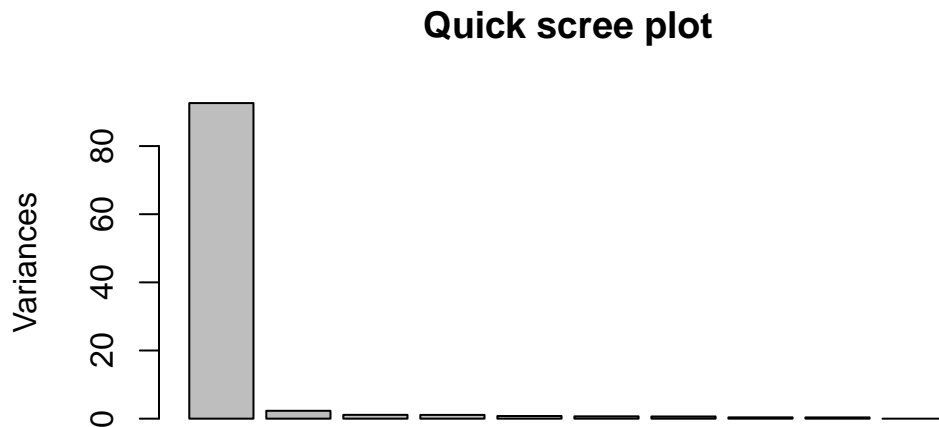

```r
## Variance per PC
pca.var <- pca$sdev^2

## Percent variance
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
```

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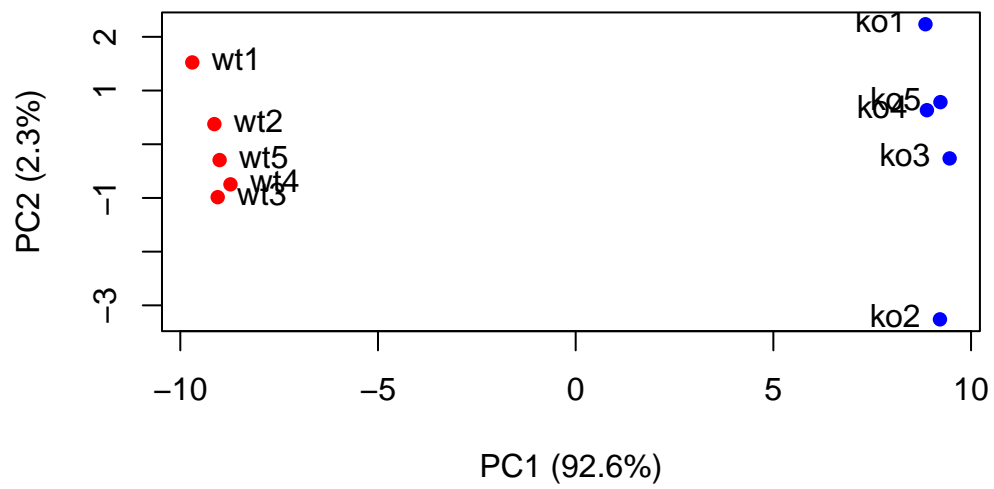
Scree Plot


```r
## Vector for 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)))
```
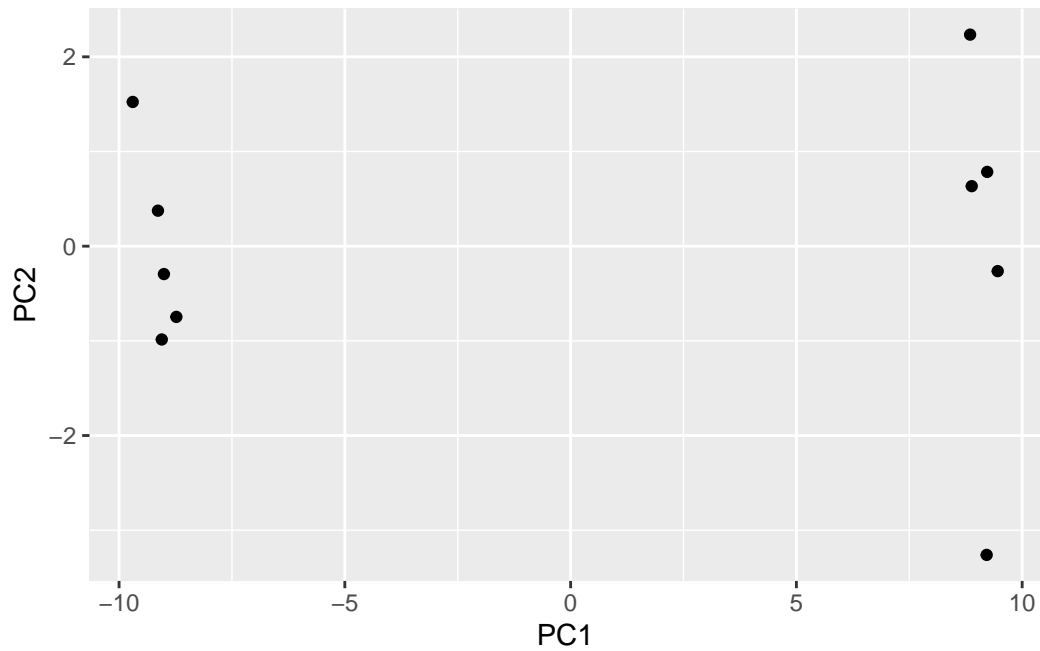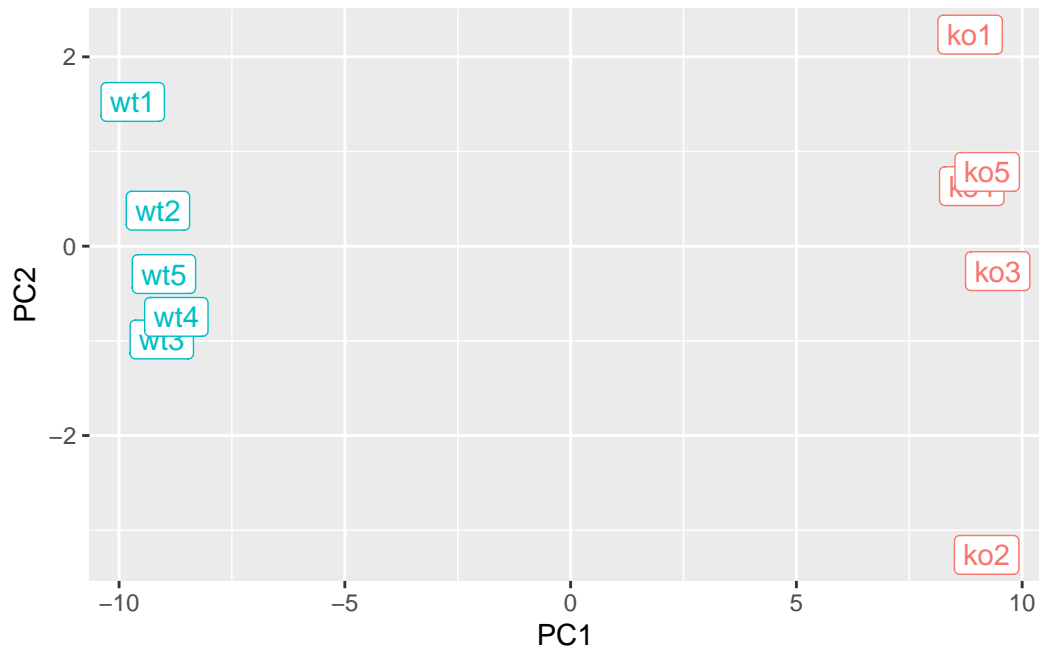
```
library(ggplot2)

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

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

```
# Add a column column for 'wt' and 'ko' "condition"
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
```
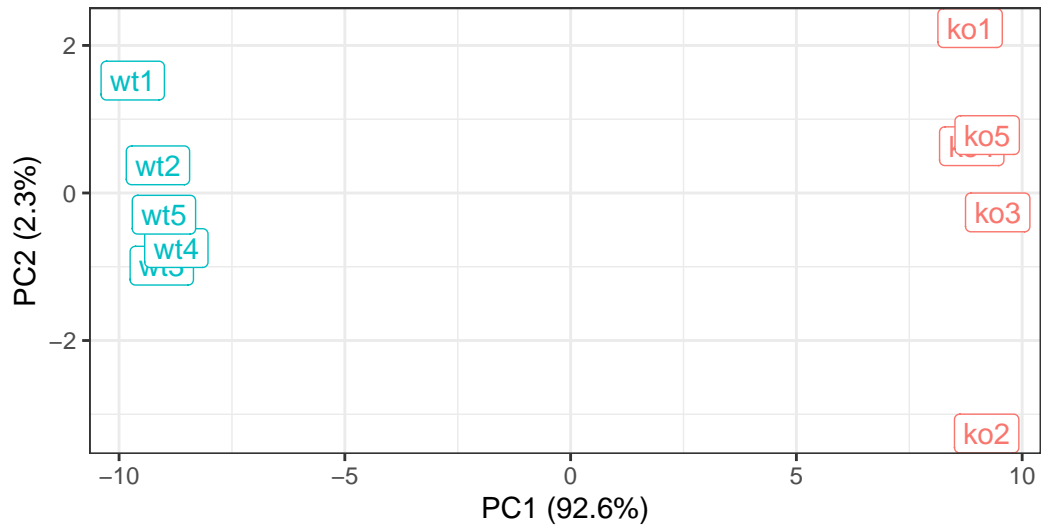
```
# A more polished version
p + labs(title = "PCA of RNASeq Data",
      subtitle = "PC1 clealy seperates wild-type from knock-out samples",
      x = paste0("PC1 (", pca.var.per[1], "%)"),
      y = paste0("PC2 (", pca.var.per[2], "%)"),
      caption = "Class example data") +
   theme_bw()
```

PCA of RNASeq Data

PC1 clealy seperates wild–type from knock–out samples

Class example data