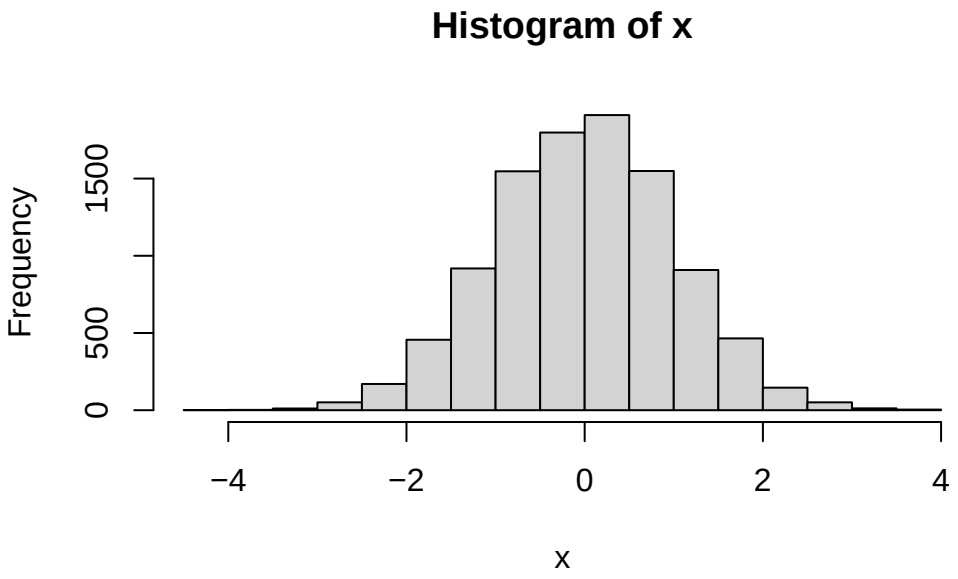
Class 7: Machine Learning 1

Aishwarya Ramesh

K-means clustering

First we will test how this method works in R with some made up data.

```
x <- rnorm(10000)
hist(x)
```

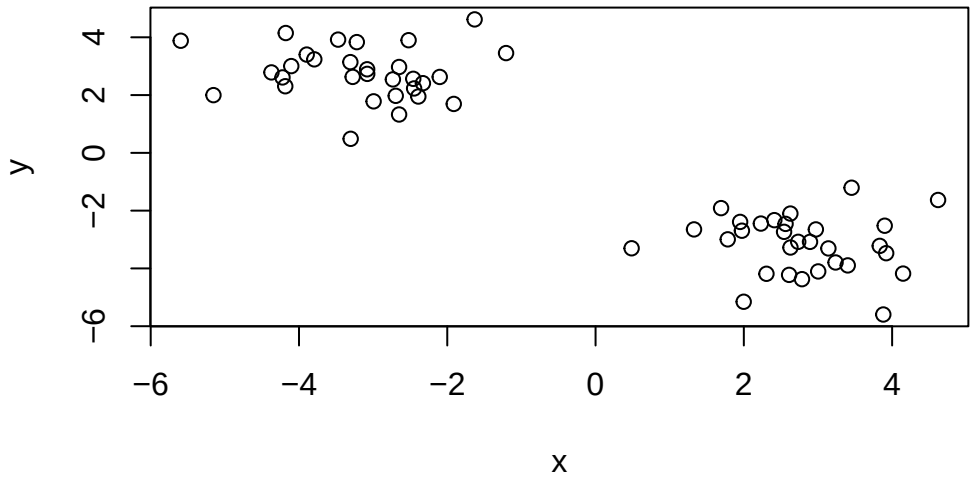


Let's make some numbers centered on -3

```
rev(c('a', 'b', 'c'))
```

```
[1] "c" "b" "a"
```

```r
tmp <- c(rnorm(30,-3), rnorm(30, 3))

x <- cbind(x=tmp, y=rev(tmp))
plot(x)
```



Now let's see how `kmeans()` works with this data.

```r
km <- kmeans(x, centers=2, nstart=20)
km
```

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

Cluster means:
          x         y
1  2.768326 -3.165310
2 -3.165310  2.768326

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] 52.80235 52.80235
 (between_SS / total_SS =  90.9 %)

Available components:

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

```
km$centers
```

```
          x         y
1  2.768326 -3.165310
2 -3.165310  2.768326
```

Q. How many points are in each cluster?

```
km$size
```

```
[1] 30 30
```

Q. What 'component' of your result object details - cluster assignment/membership?
- cluster center

```
km$cluster
```

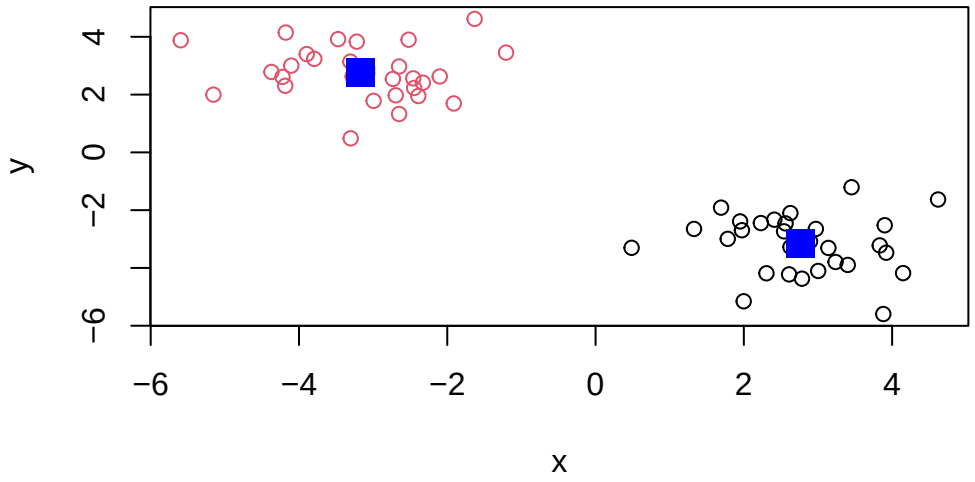
```
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
[39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
km$centers
```

```
          x         y
1  2.768326 -3.165310
2 -3.165310  2.768326
```

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

```r
plot(x, col=km$cluster)
points(km$centers, col='blue', pch=15, cex=2)
```



Hierarchical Clustering

The `hclust()` function in R performs hierarchical clustering.

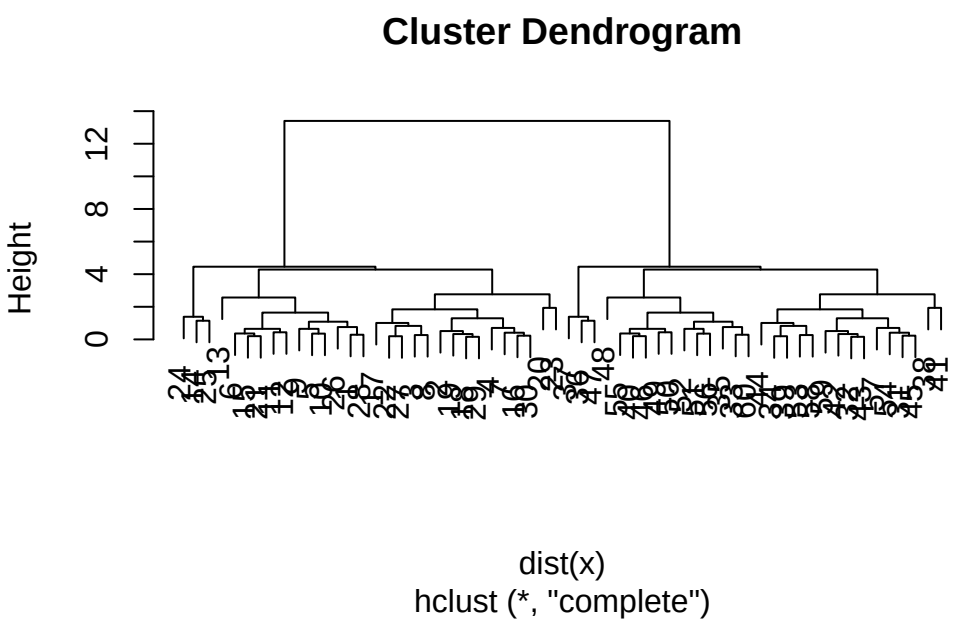
The `hclust()` function requires an input of a distance matrix, which we can get from the `dist()` function.

```r
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 objects

```
plot(hc)
```



Now to get my cluster membership vector I need to "cut" the tree to yield separate "branches" with the "leaves" on each branch being our clusters. To do this, we use the `cutree()` function.

```
cutree(hc, h=9)
```

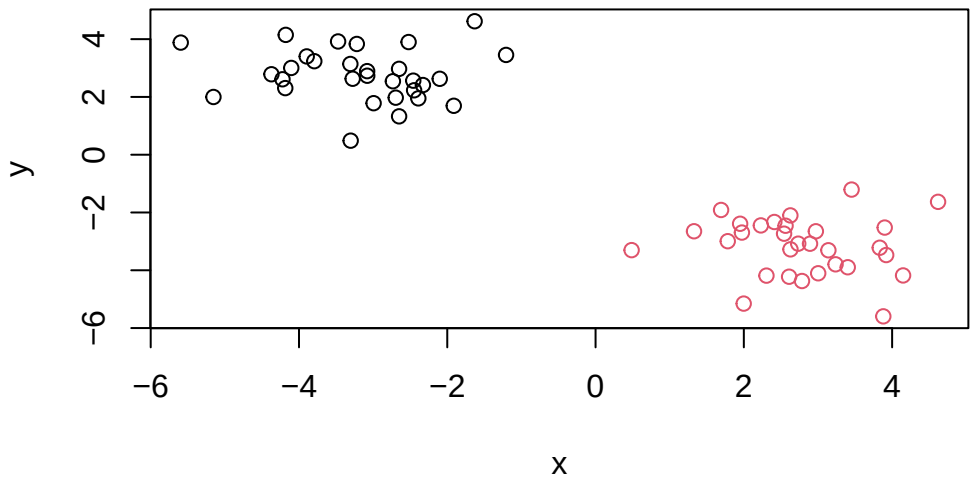
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Use `cutree()` with a k=2.

```
grps <- cutree(hc, k=2)
```

A plot of our data colored by our hclust grps.

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



Principal Component Analysis (PCA)

PCA of UK food data

Data import

```
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
```

```
# dim() function gives rows and columns
```

Checking your data

Looking at the first 6 rows of data:

```
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
```

Fixing error where rownames listed as column:

```
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
```

Checking dimension now – should be 4 columns

```
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?

Answer: The dataframe slicing method is less efficient, as it will cut more columns than we want if it is run multiple times. Therefore, it is preferred to set rownames while reading in the dataset, as is done below.

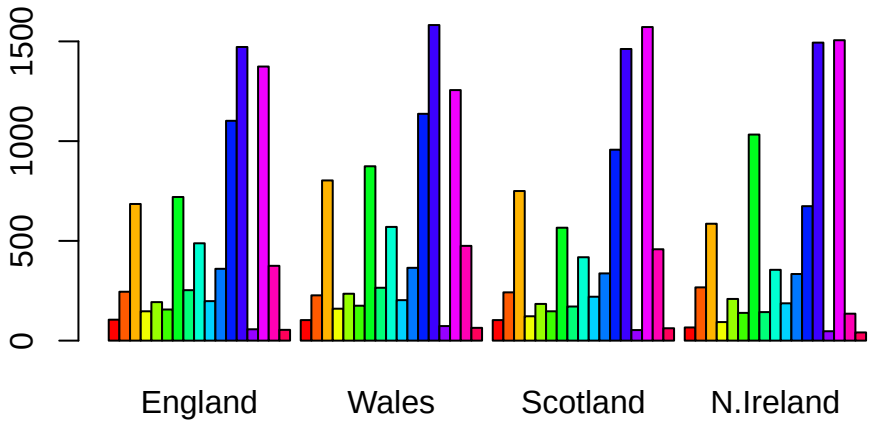
```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names=1)
head(x)
```

```
               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
```

Spotting major differences and trends

Generating a regular barplot:

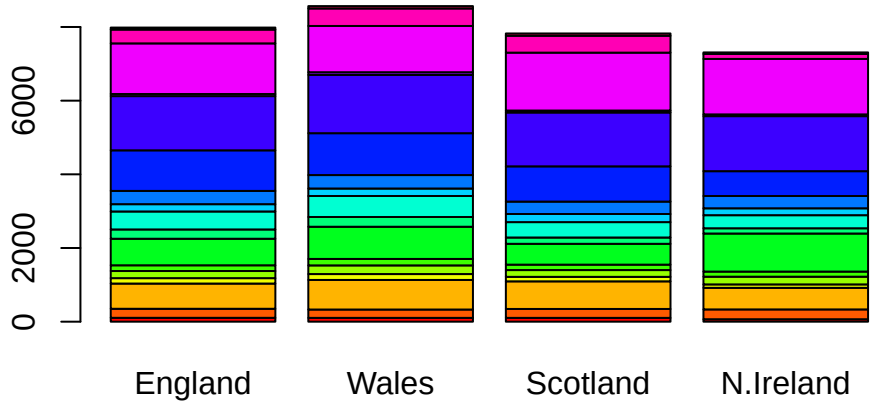
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

We just have to change beside to F.

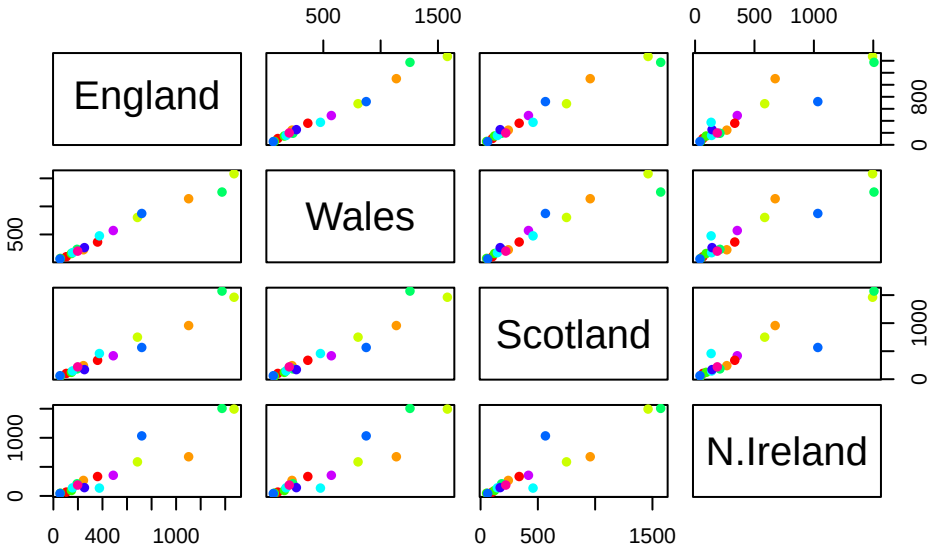
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



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 following code creates pairwise plots for all categories. For example, the top row plots England's food consumption vs Wales, Scotland and N.Ireland in that order. If a point lies on the diagonal, this means that consumption of that food is the same or similar for both countries being plotted.

```
pairs(x, col=rainbow(10), pch=16)
```



Note: log fold change refers to what the log2 of the slope is. For example, if England eats 20 potatoes and Wales eats 10, there's a log fold change of $\log(20/10) = 1$.

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

While this is somewhat useful, it takes work to dig into details to find out what is different between countries.

PCA to the rescue

Principal Component Analysis (PCA) can help us when we have lots of things that are being measured i.e. many dimensions in a dataset.

The main PCA function in base R is called `prcomp()`.

The `prcomp()` function wants as input the transpose of our food matrix/table/data.frame.

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

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 4.189e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

PC1 captured 67.44% of the total variance in the dataset, as is indicated by Proportion of Variance.

Cumulative proportion indicates how much of the variance would be captured if you used this PC and also all before it.

Above results shows that PCA captures 67% of the total variance in the original data in one PC and 96.5% in two PCs.

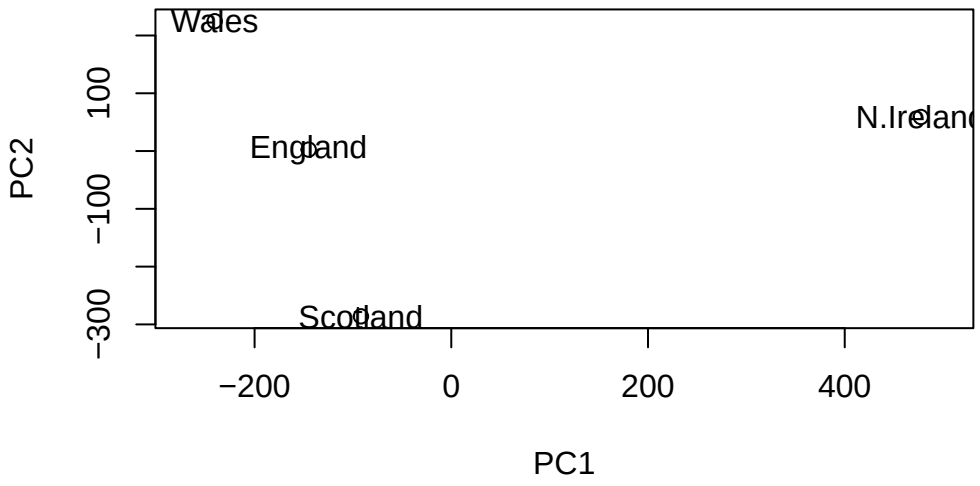
```
head(pca$x)
```

```
                PC1         PC2         PC3           PC4
England   -144.99315    2.532999 -105.768945  2.842865e-14
Wales     -240.52915  224.646925   56.475555  7.804382e-13
Scotland   -91.86934 -286.081786   44.415495 -9.614462e-13
N.Ireland  477.39164   58.901862    4.877895  1.448078e-13
```

Let's plot our main results

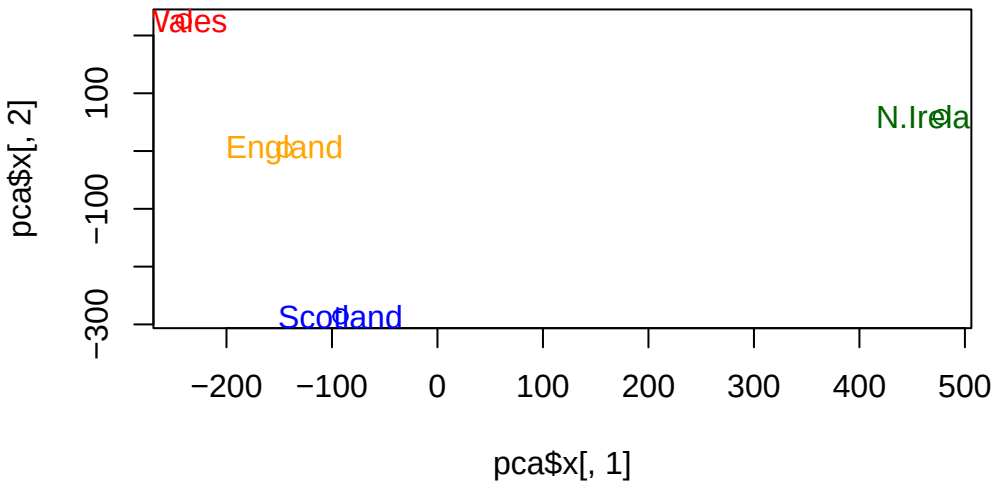
> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
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'))
text(pca$x[,1], pca$x[,2], colnames(x), col=c('orange', 'red', 'blue', 'darkgreen'))
```



Digging deeper: variable loadings

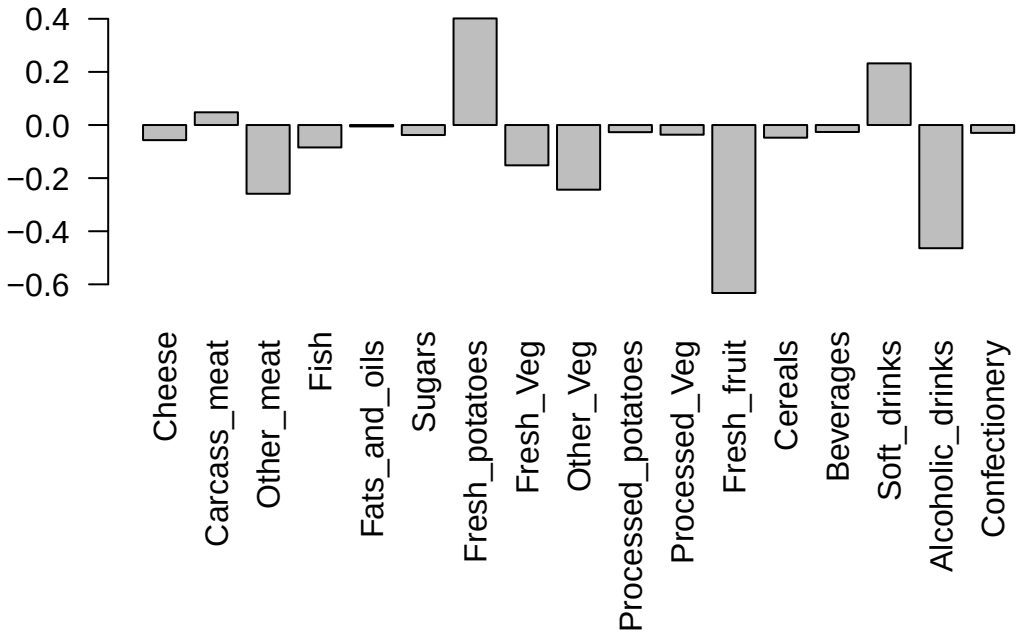
```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

```
## or the second row here...
z <- summary(pca)
z$importance
```

```
                             PC1       PC2      PC3           PC4
Standard deviation     324.15019 212.74780 73.87622 4.188568e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

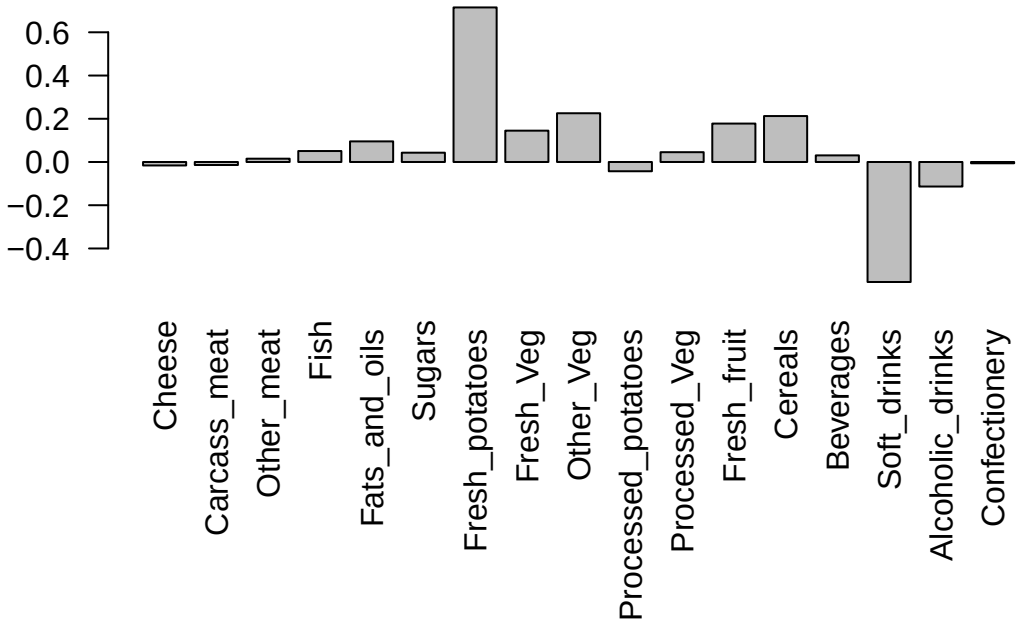
```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



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

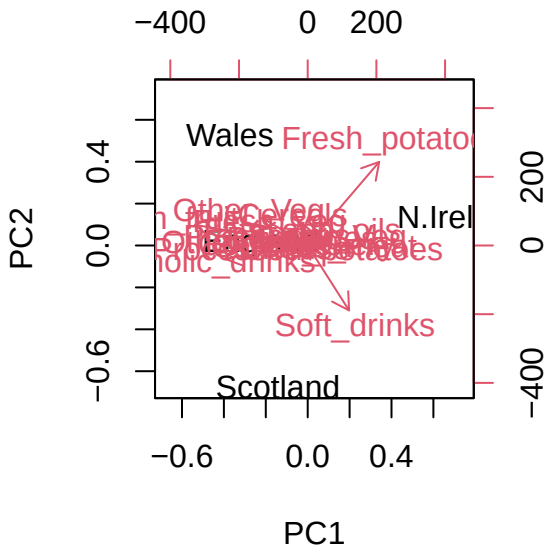
Fresh potatoes and soft drinks feature prominently on the plot for PC2. PC2 is essentially the axis with the second most variance. The bars in the below plot essentially describe the difference between countries on the PC2 axis, and which foods are most different between countries on that axis. For example, fresh potatoes and soft drinks are different between Scotland and Wales, which are stratified along the PC2 axis.

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



Creating biplot

```
biplot(pca)
```



PCA of RNA-seq data

First, reading in the data and checking the first 6 rows.

```
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?

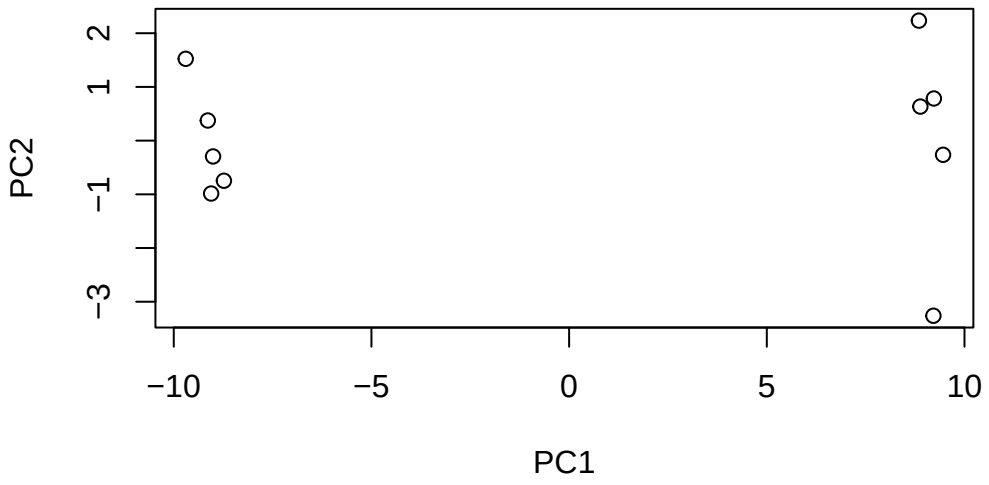
There are 100 genes and 10 samples for each gene in the rna data.

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

```
[1] 100  10
```

Doing PCA on this data and getting a plot for the results:

```
## 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")
```



Now, getting a summary of different PC axes and their variances.

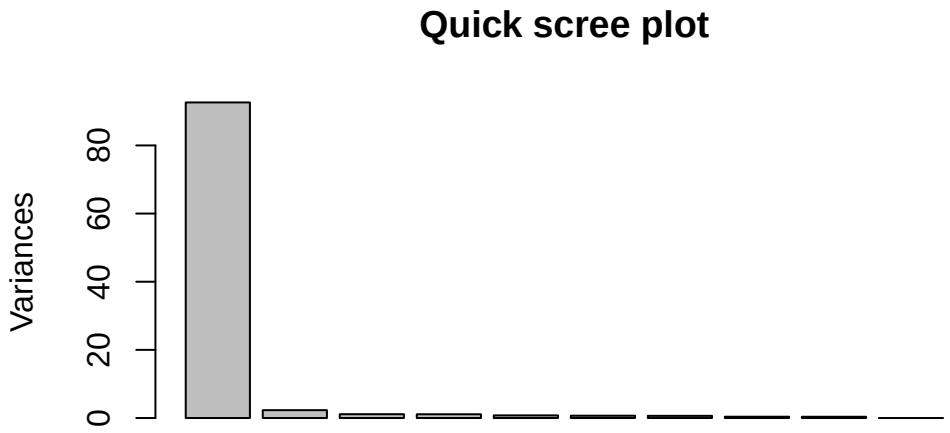
```
summary(pca)
```

```
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 3.348e-15
```

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

It seems like PC1 is accounting for 92.62% of variance in the data. To verify, can create a barplot for Proportion of Variance for each PC.

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



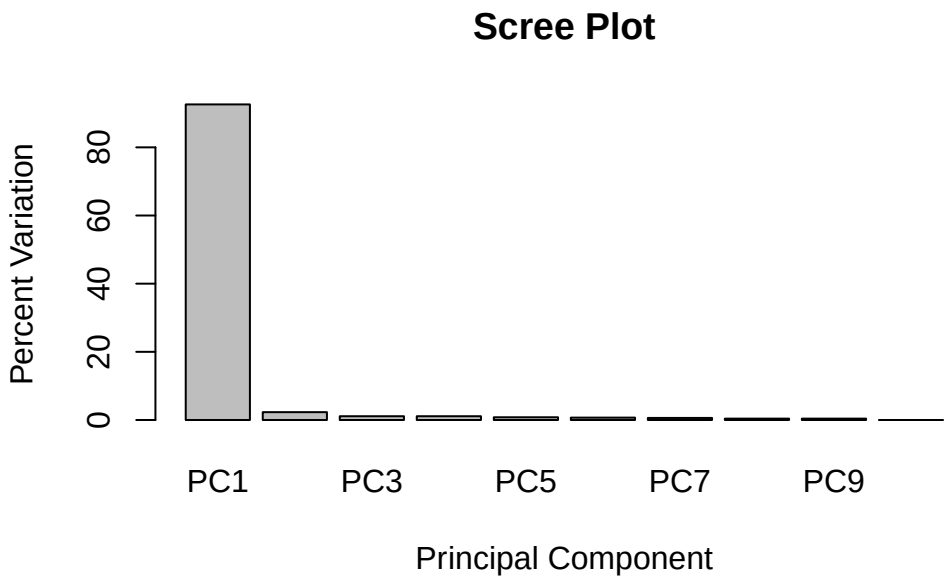
Trying to make scree plot ourselves, and investigating output object of the prcomp function.

```
## 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
```

Using this to generate plot:

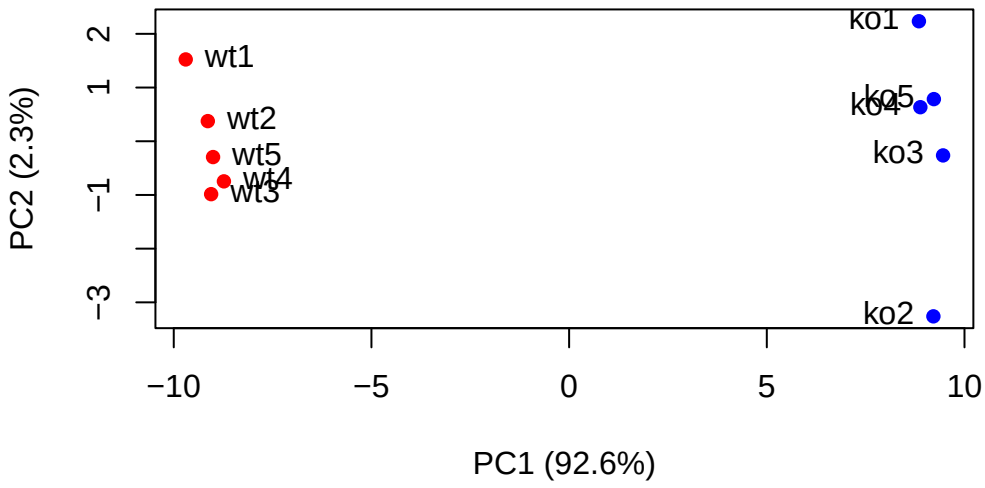
```
barplot(pca.var.per, main="Scree Plot",
        names.arg = paste0("PC", 1:10),
        xlab="Principal Component", ylab="Percent Variation")
```



Again, we see that PC1 contains all the variation.

Now, making main PCA plot more attractive.

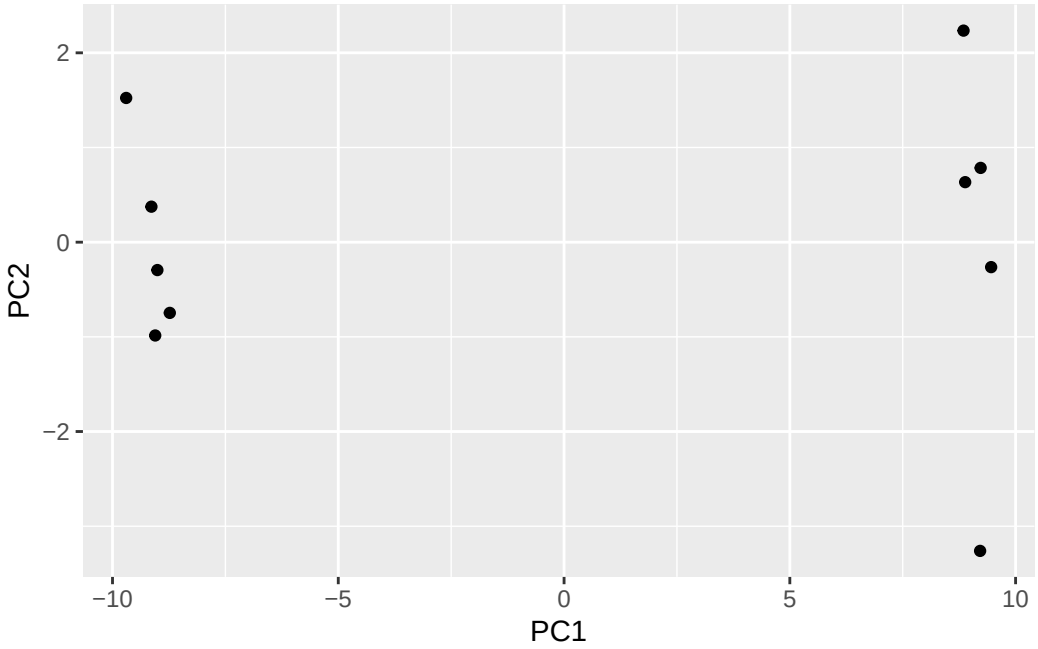
```
# A vector of colors for wt and ko samples
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

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)))
```



Using ggplot

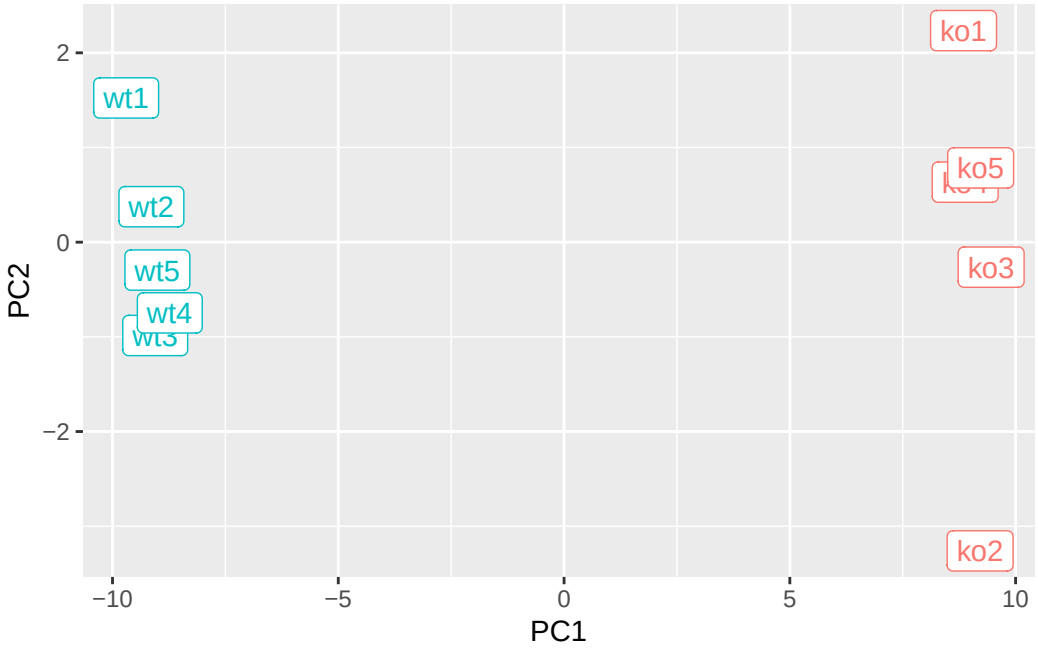
First, we must create dataframe representing PCA data and then plot.

```
library(ggplot2)

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

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



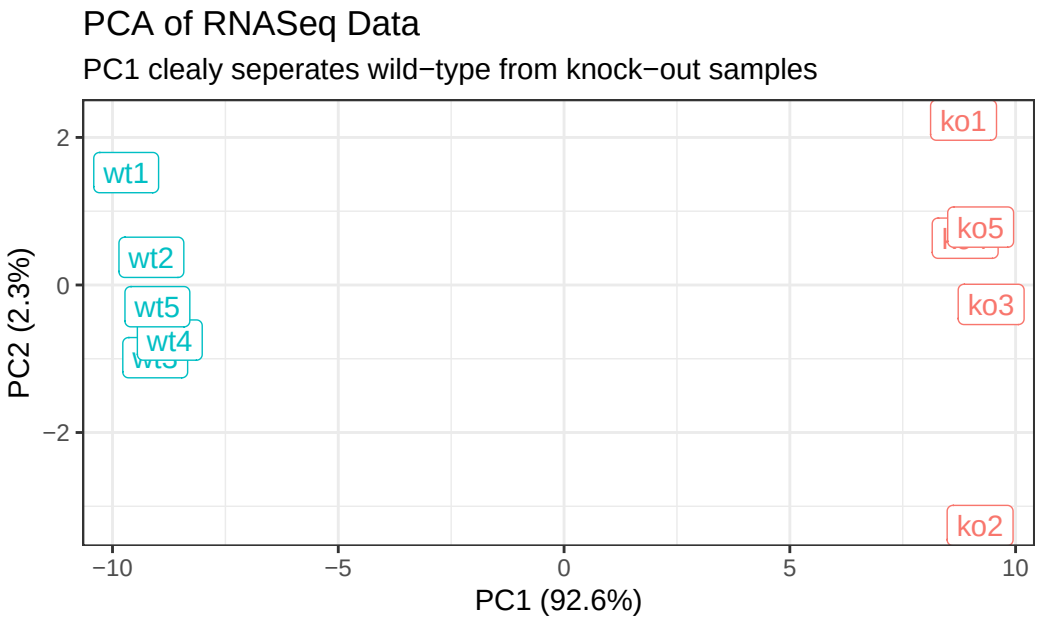
Adding aesthetic conditions, and labels of wt vs ko:

```
# 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
```



Adding titles and labels:

```
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()
```



Optional: gene loadings

Finding top 10 genes that contribute to PC1 in either direction.

```
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"
```