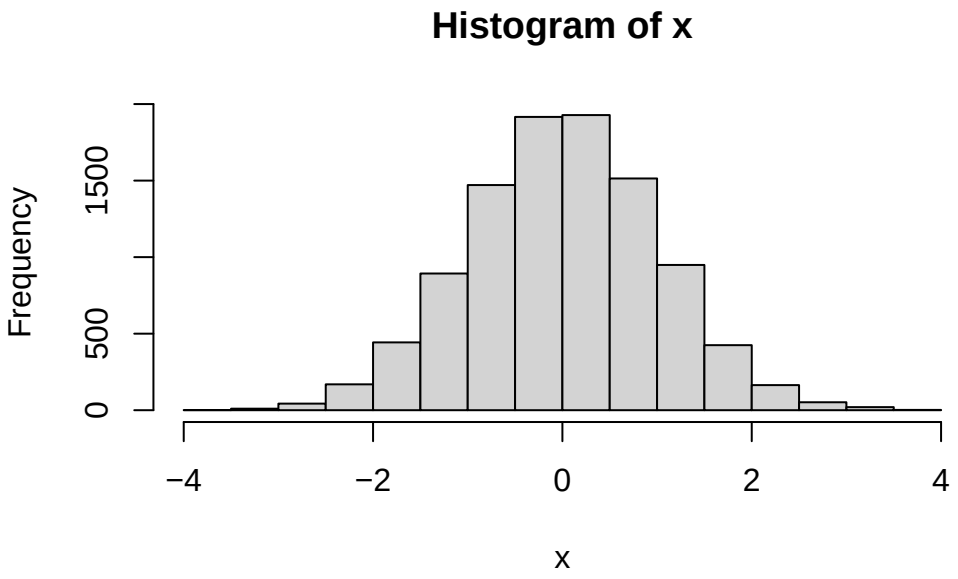
Class 07 Machine Learning 1

Zavier Annis

K-means clustering

First: test how method works in R with made-up data.

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



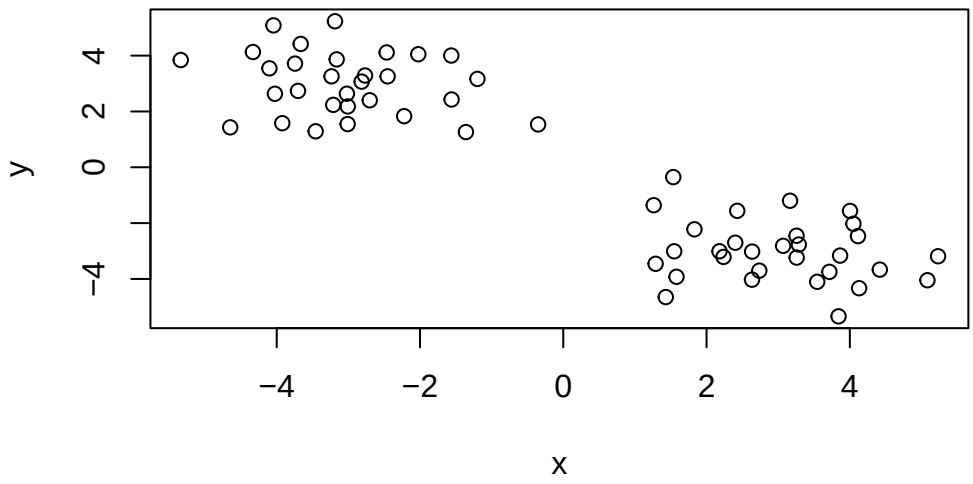
Now let's make numbers with -3 and +3 as their center.

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

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

```
              x          y
 [1,] -4.0258119  2.6341169
 [2,] -4.3322060  4.1292929
 [3,] -3.7450569  3.7146886
 [4,] -3.9223717  1.5802739
 [5,] -0.3512665  1.5349198
 [6,] -1.5591333  2.4281115
 [7,] -1.1984449  3.1654587
 [8,] -3.0084712  2.1804994
 [9,] -5.3399225  3.8423844
[10,] -2.0237174  4.0498262
[11,] -3.0206240  2.6359168
[12,] -3.1632653  3.8639904
[13,] -4.0458243  5.0832481
[14,] -4.1017529  3.5447284
[15,] -1.5632689  4.0028834
[16,] -3.2350189  3.2577820
[17,] -3.1867180  5.2316029
[18,] -3.4563584  1.2887648
[19,] -2.4519707  3.2563228
[20,] -4.6491410  1.4306370
[21,] -2.8142618  3.0685423
[22,] -3.0100259  1.5482585
[23,] -3.6653630  4.4184891
[24,] -2.7675303  3.2874060
[25,] -2.7015777  2.4008692
[26,] -1.3589648  1.2621446
[27,] -3.2108149  2.2365047
[28,] -3.7026611  2.7364774
[29,] -2.4654527  4.1138532
[30,] -2.2212527  1.8310259
[31,]  1.8310259 -2.2212527
[32,]  4.1138532 -2.4654527
[33,]  2.7364774 -3.7026611
[34,]  2.2365047 -3.2108149
[35,]  1.2621446 -1.3589648
[36,]  2.4008692 -2.7015777
```

```
[37,]  3.2874060 -2.7675303
[38,]  4.4184891 -3.6653630
[39,]  1.5482585 -3.0100259
[40,]  3.0685423 -2.8142618
[41,]  1.4306370 -4.6491410
[42,]  3.2563228 -2.4519707
[43,]  1.2887648 -3.4563584
[44,]  5.2316029 -3.1867180
[45,]  3.2577820 -3.2350189
[46,]  4.0028834 -1.5632689
[47,]  3.5447284 -4.1017529
[48,]  5.0832481 -4.0458243
[49,]  3.8639904 -3.1632653
[50,]  2.6359168 -3.0206240
[51,]  4.0498262 -2.0237174
[52,]  3.8423844 -5.3399225
[53,]  2.1804994 -3.0084712
[54,]  3.1654587 -1.1984449
[55,]  2.4281115 -1.5591333
[56,]  1.5349198 -0.3512665
[57,]  1.5802739 -3.9223717
[58,]  3.7146886 -3.7450569
[59,]  4.1292929 -4.3322060
[60,]  2.6341169 -4.0258119
```

```
plot(x)
```



Now it's time to try 'kmeans()' with this data!

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

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

Cluster means:
          x         y
1 -3.009942  2.991967
2  2.991967 -3.009942

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 2

Within cluster sum of squares by cluster:
[1] 72.24527 72.24527
 (between_SS / total_SS =  88.2 %)

Available components:
```

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

Q. How many points are in each cluster?

```
km$size
```

```
[1] 30 30
```

A. 30 points in each cluster.

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

```
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 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
```

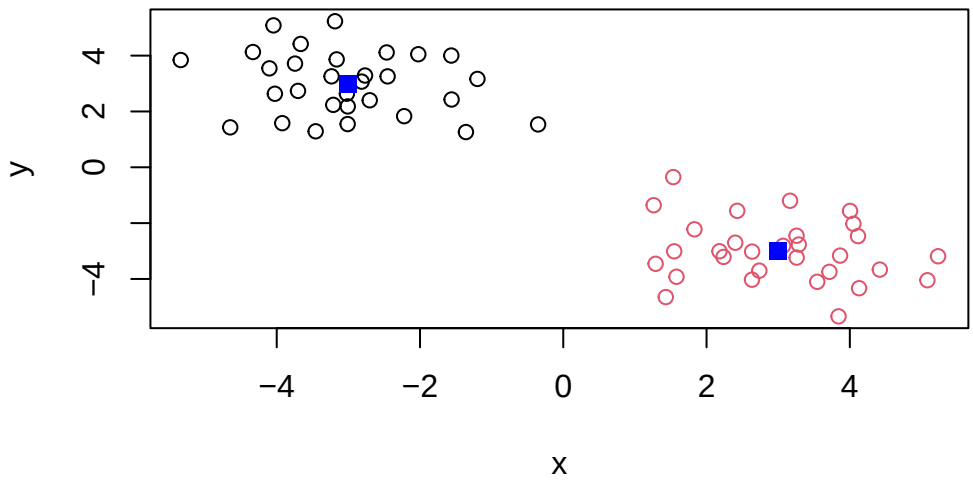
```
km$centers
```

```
          x         y
1 -3.009942  2.991967
2  2.991967 -3.009942
```

A. km*clusterdetailsassignment/membership.km*centers details the cluster center locations.

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

```
plot(x, col=km$cluster)
points(km$centers, col="blue", pch=15, cex=1.2)
```



Hierarchical Clustering

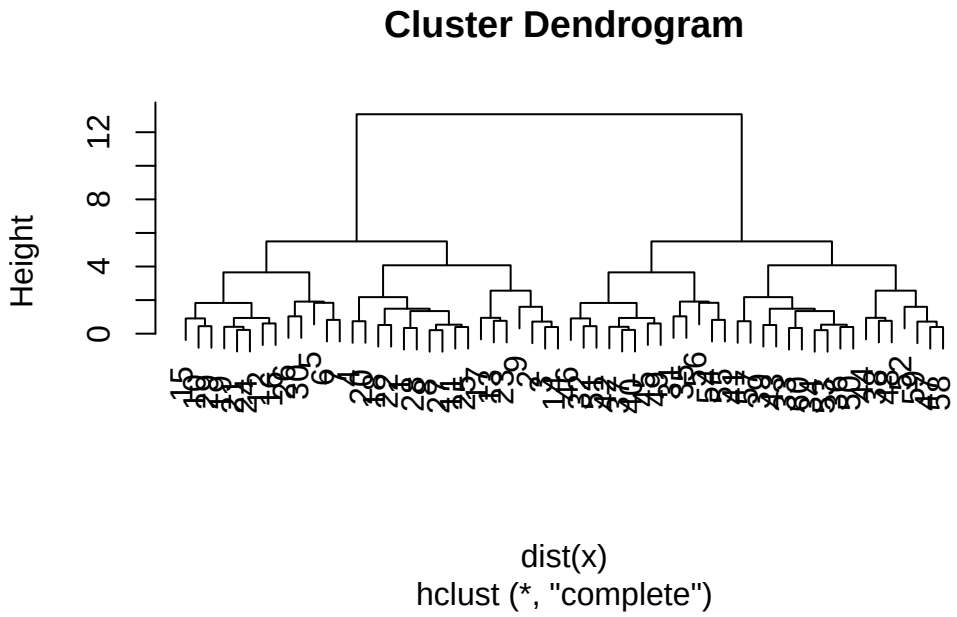
'hclust()' function in R performs hierarchical clustering. It also requires an input distance matrix, which can be obtained from 'dist()' function.

```
hc <- hclust(dist(x))
hc
```

```
Call:
hclust(d = dist(x))

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

There is also a plot method for hclust objects:

```
plot(hc)
```



Now, to get cluster membership vector, I have to "cut" the tree to get separate "branches" (the "leaves" are the clusters). To do this, use the 'cutree()' function.

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

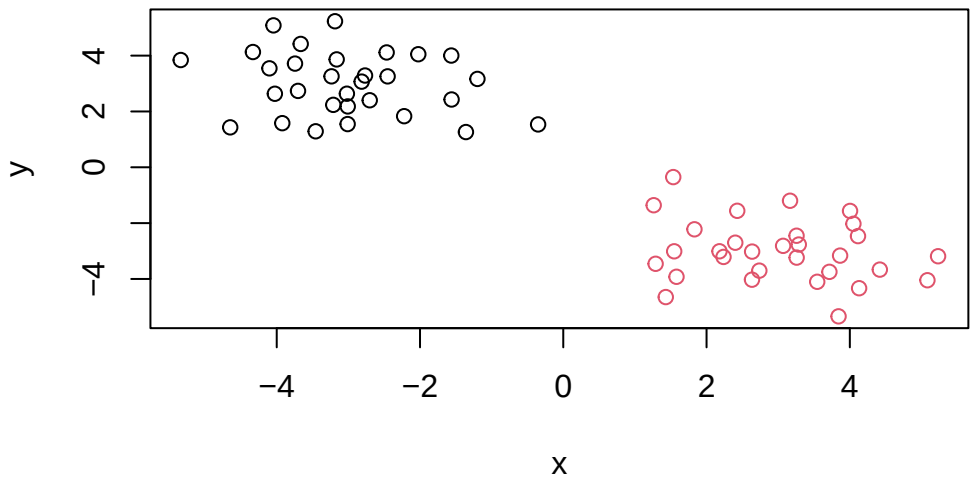
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Use 'cutree()' with a k=2 (instead of finding the height h on the plot) to get 2 groups.

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

Now for a plot colored by hclust groups:

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



Principal Component Analysis (PCA)

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url)
```

Q1.

```
## Complete the following code to find out how many rows and columns are in x?
dim(x)
```

```
[1] 17  5
```

A1. There are 17 rows and 5 columns in x.

Q2.

```
## Preview the 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
```

```
# Note how the minus indexing works
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
```

```
# Check dimensions again:
dim(x)
```

```
[1] 17  4
```

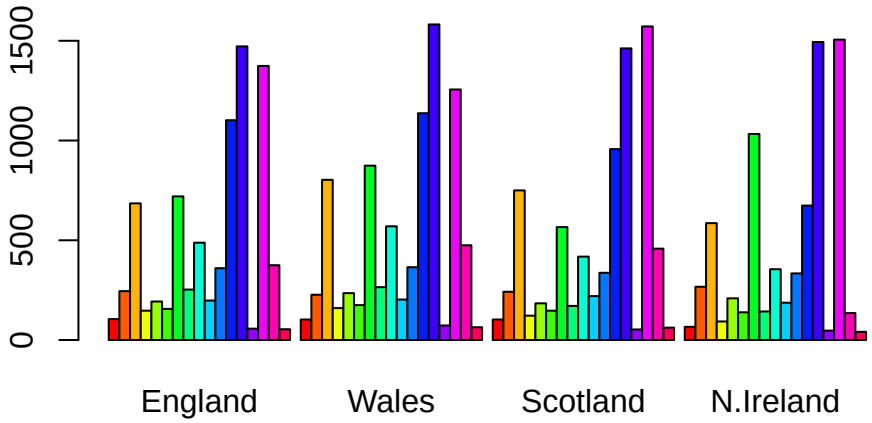
```
# You can also do this:
x <- read.csv(url, row.names=1)
head(x)
```

```
              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
```

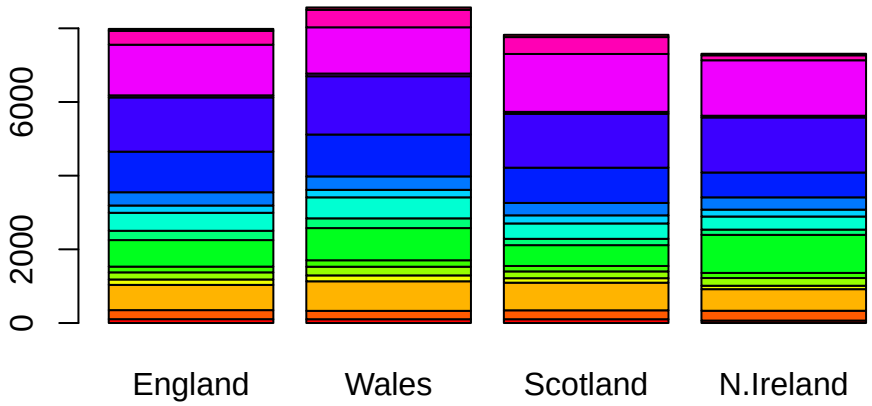
A2. I strongly prefer the second approach because it is far more efficient, only involving one step. Under circumstances where the code might be run more than once or rendered later, the second method is much more robust becuase the first method will continue to remove columns that contain valuable data.

Q3.

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



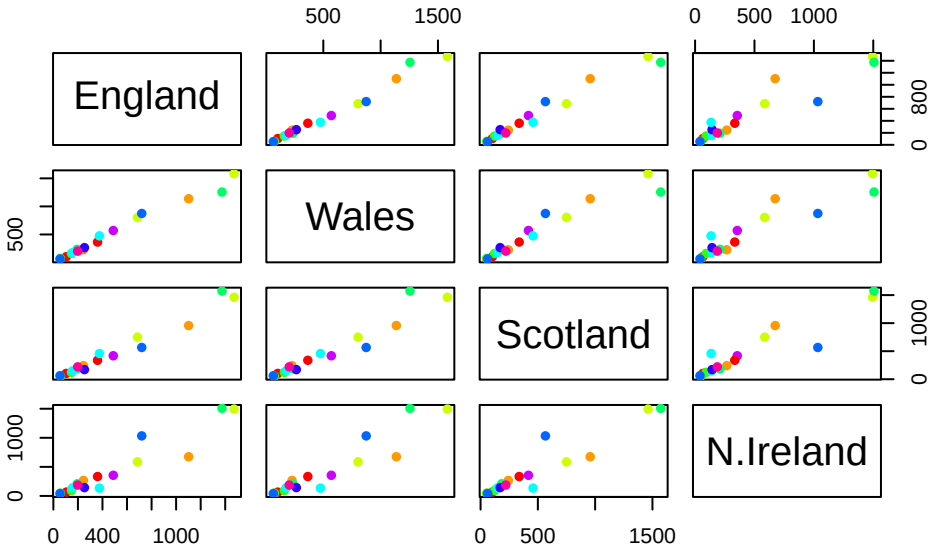

```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



A3. Changing the 'beside' argument in 'barplot()' function to FALSE instead of TRUE results in the stacked bar plot instead of the dodge/beside plot.

Q5.

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



A5. The code generates a figure that plots points that compare two countries (in each smaller plot) for a given category. One country (the one parallel to the horizontal border) is on the x-axis and the other is on the y-axis. If a given point lies on the diagonal for a given plot, that indicates that the countries have similar values (in g/week consumed) for that category. This is somewhat useful but it takes substantial work to understand the details and find out what is different between the countries.

Q6. / A6. Per the dataset, it appears that N. Ireland consumes substantially more fresh potatoes than the other countries and substantially less fresh fruit than the other countries.

PCA to the rescue! Main function is called 'prcomp()'.

Q7.

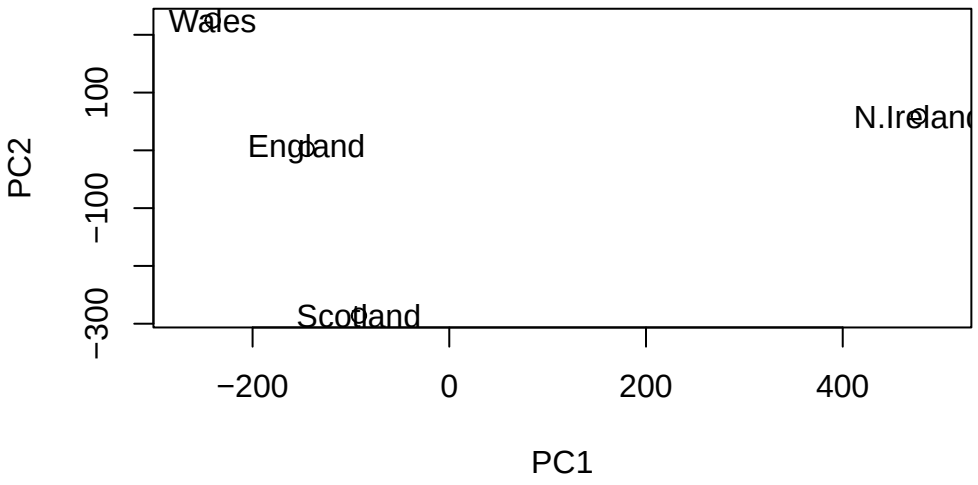
```
pca <- prcomp(t(x))
summary(pca)
```

```
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
```

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

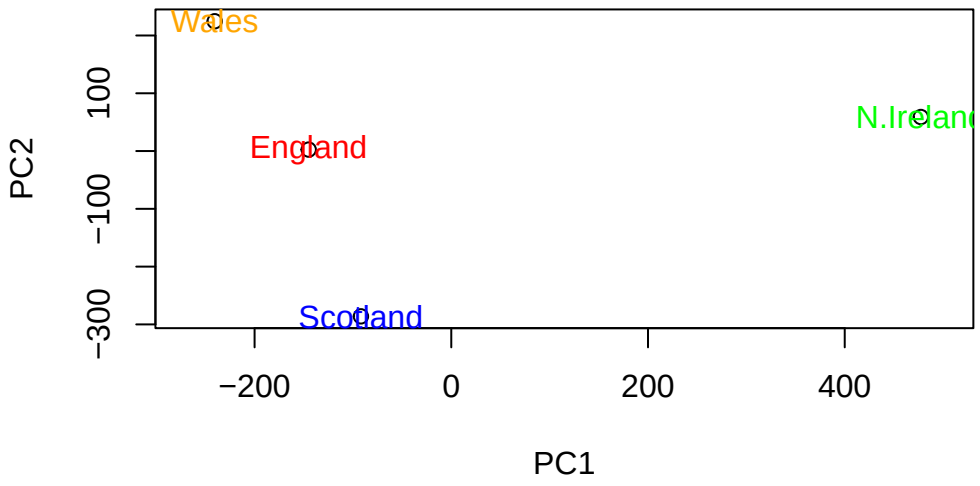
```
# Plot PC1 vs PC2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x))
```



A7. The blanks in the code are filled as ‘,1’ and ‘,2’ to include the first and second columns in the pca dataset, corresponding to the data from PC1 and PC2.

Q8.

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



A8. Colors are added using the 'col()' function with a vector argument that includes the four colors of the flag: col=c("red", "orange", "blue", "green")

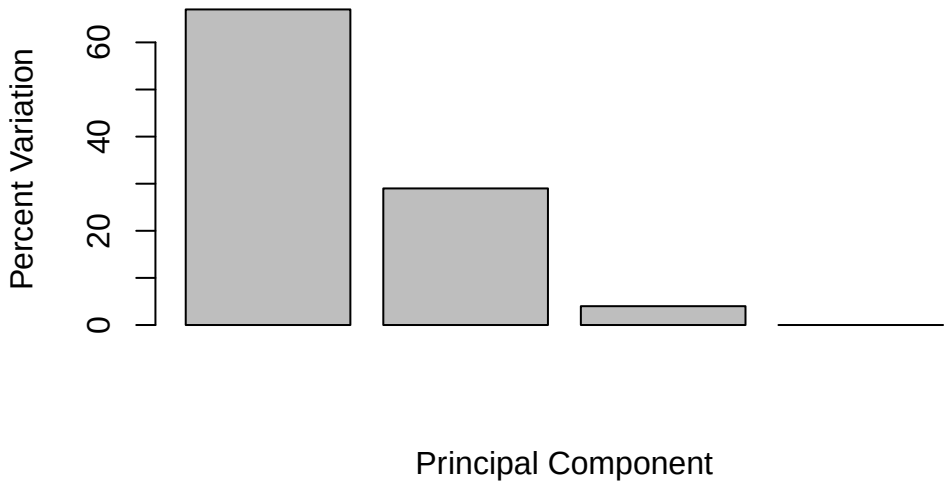
```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

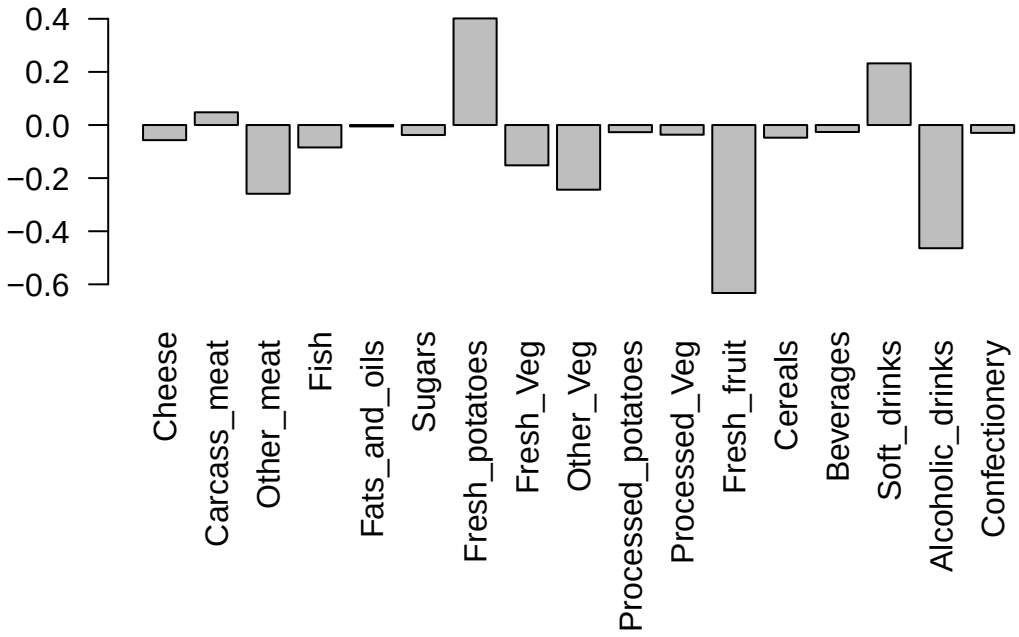
```
## or the second row here...
z <- summary(pca)
z$importance
```

```
                             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
```

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

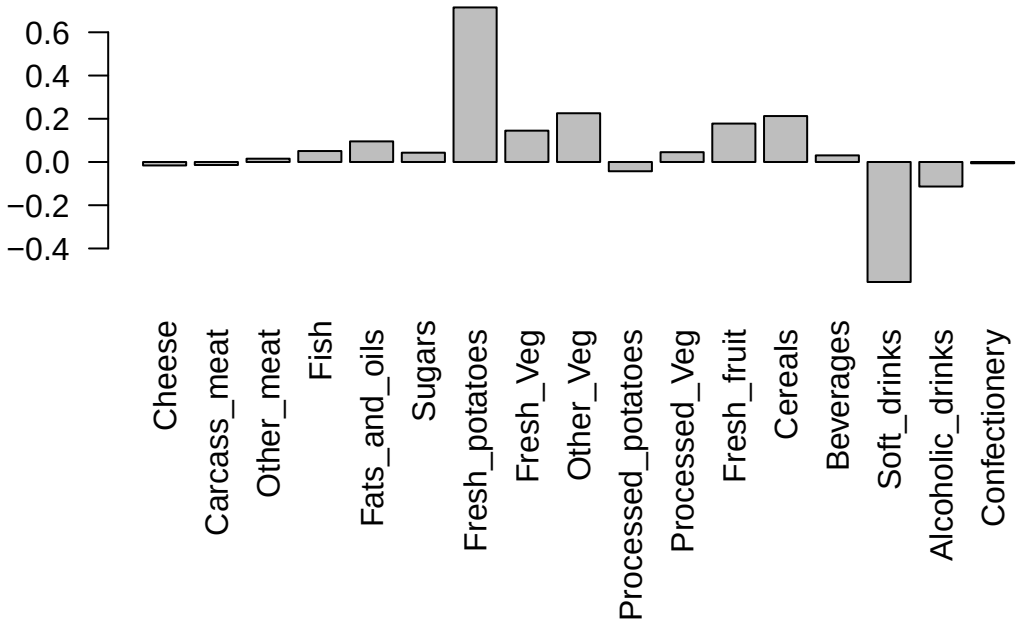



```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



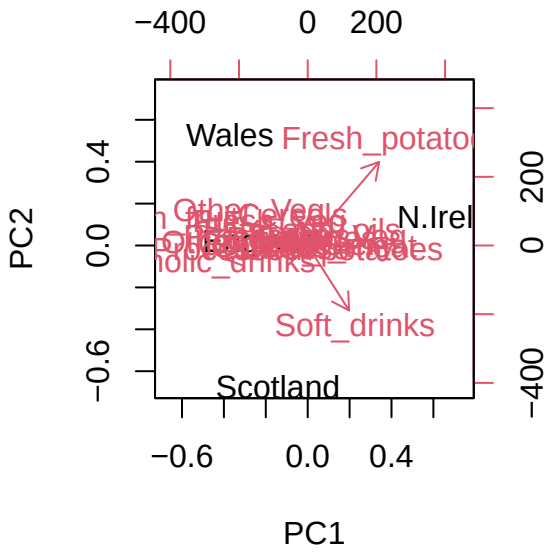
Q9.

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



A9. The two food groups that feature predominantly are fresh potatoes and soft drinks. PC2 mainly tells us about variation in fresh potato and soft drink consumption in N. Ireland (compared to the rest of the UK) not already accounted for by PC1.

```
## The inbuilt biplot() can be useful for small datasets
biplot(pca)
```




```
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.

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

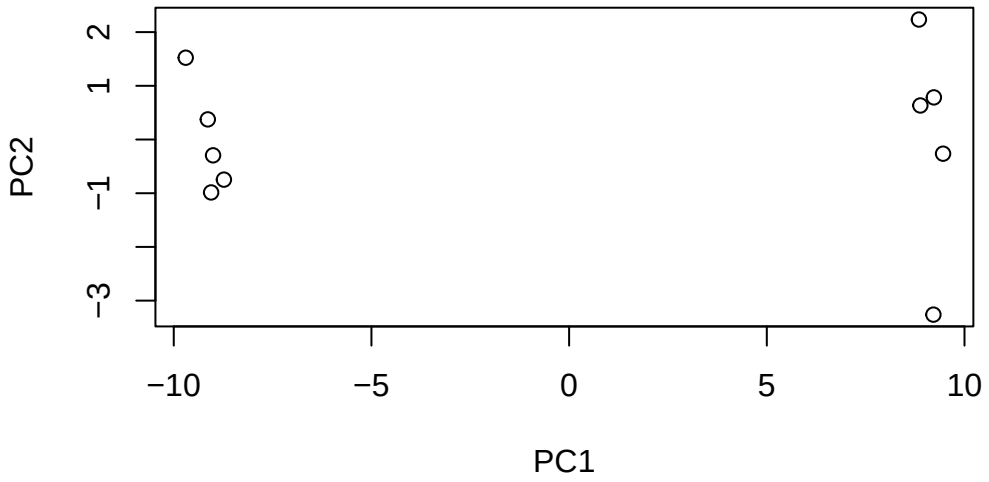
```
[1] 100
```

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

```
[1] 10
```

A10. There are 100 genes and 10 samples in this data set (for a grand total of 1000 individual data points).

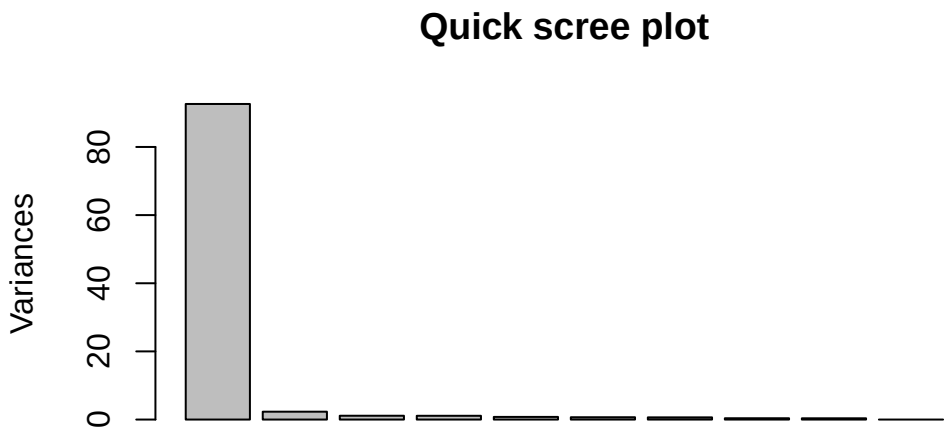
```
## Again we have to take the transpose of our data
pca <- prcomp(t(rna.data), scale=TRUE)

## 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 3.348e-15
Proportion of Variance 0.00385 0.00364 0.000e+00
Cumulative Proportion  0.99636 1.00000 1.000e+00
```

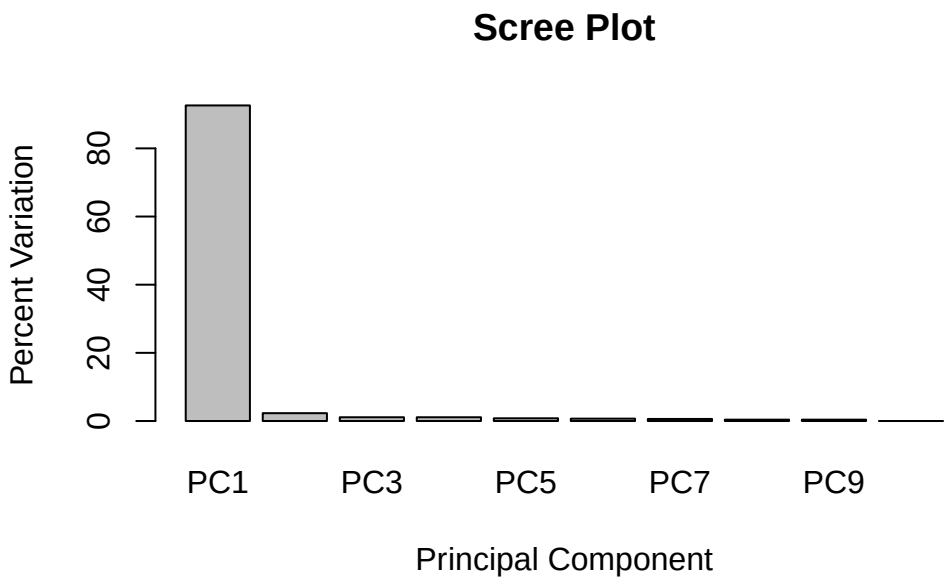
```r
plot(pca, main="Quick scree plot")
```



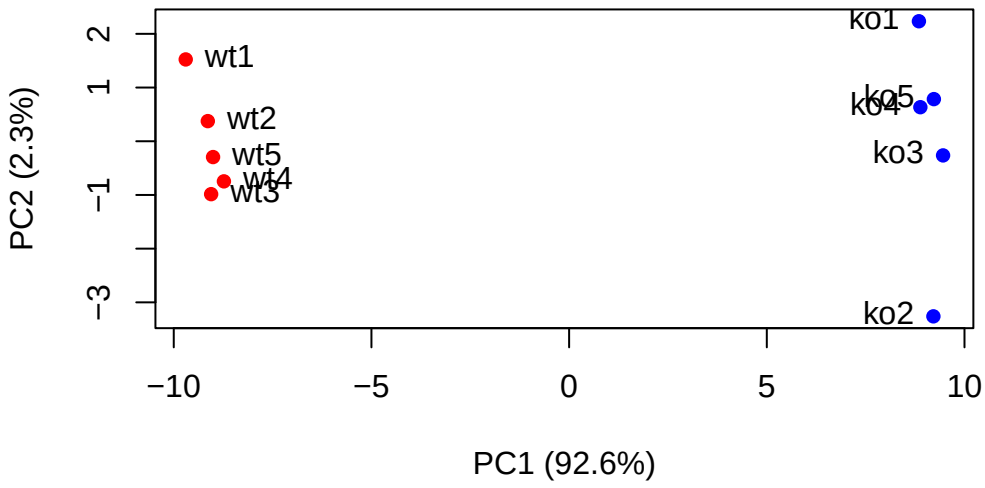

```r
## Variance captured per PC
pca.var <- pca$sdev^2

## 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
```

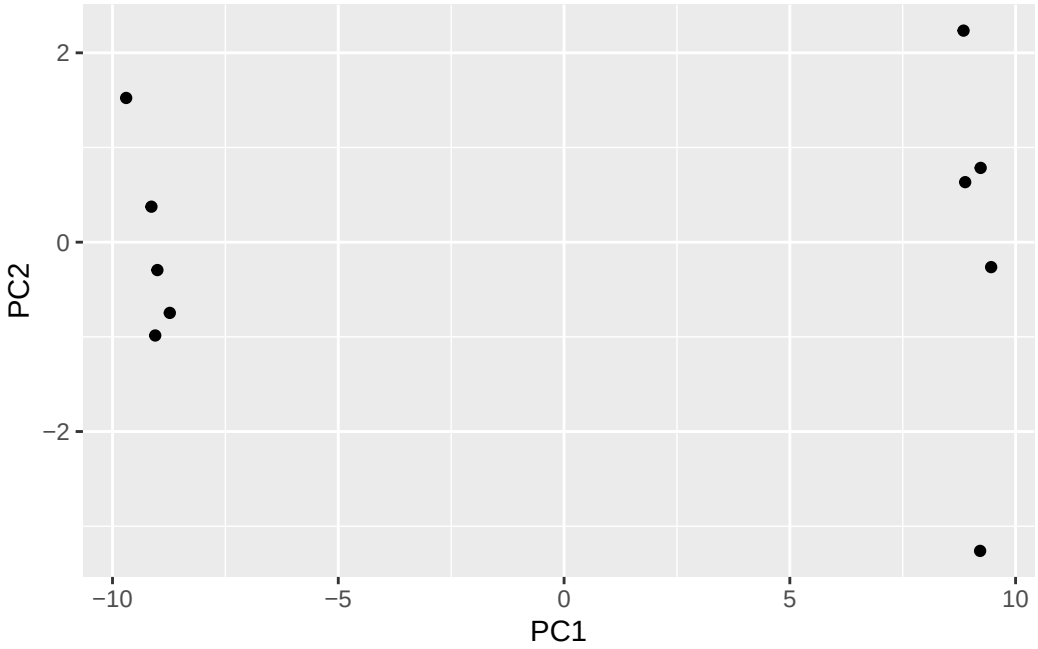
```r
barplot(pca.var.per, main="Scree Plot",
        names.arg = paste0("PC", 1:10),
        xlab="Principal Component", ylab="Percent Variation")
```



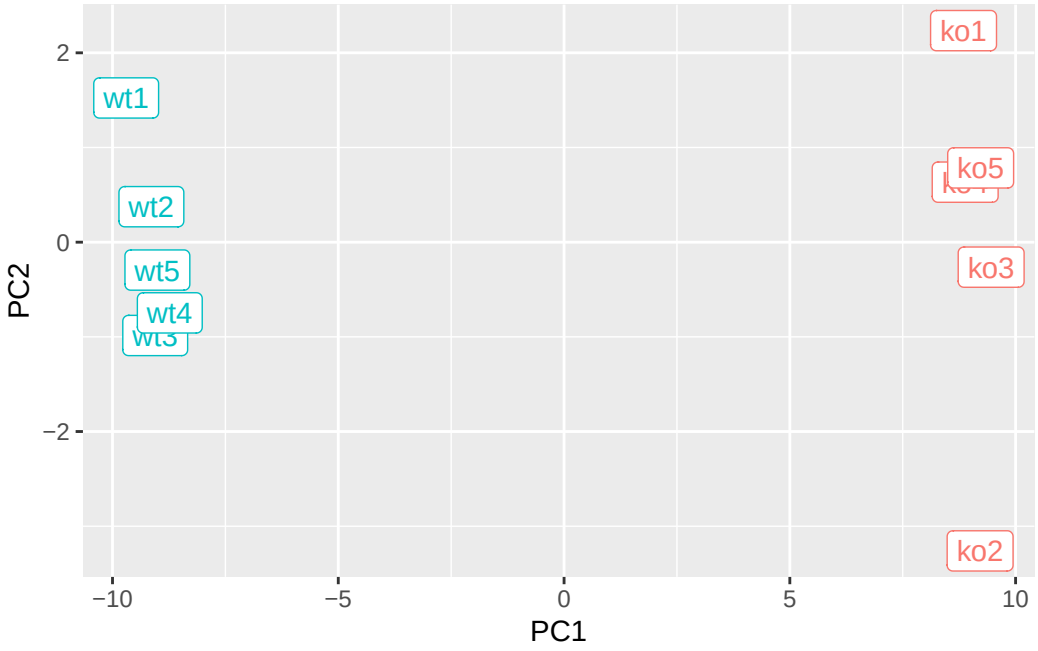

```
## A vector of colors for wt and ko samples
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

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)))
```



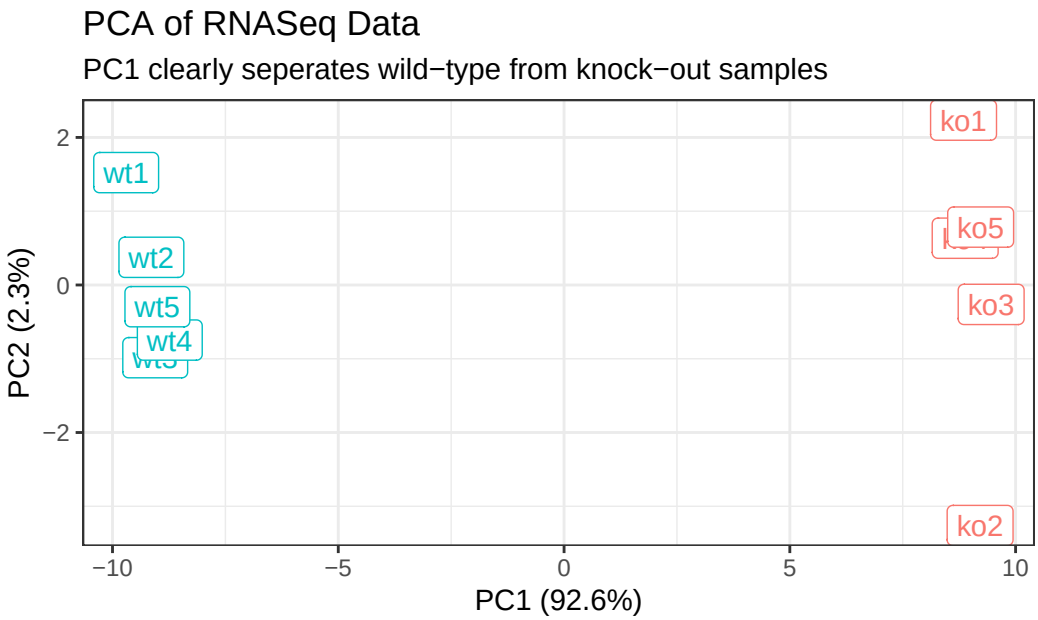

```
library(ggplot2)

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