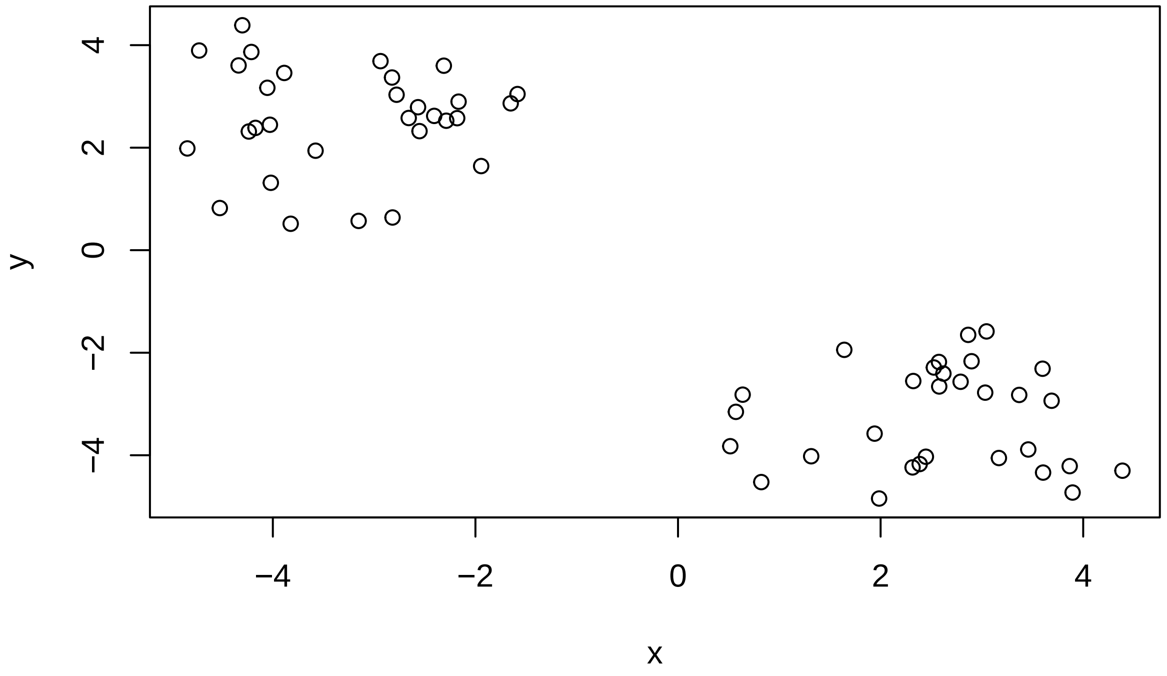
Machine Learning 1

#Clustering Methods

kmeans clustering in R is done with the ‘kmeans()’ function. Here we makeup some data to test and learn with. #rev command reverses the data set #cbind binds vectors by column. rbind binds them by rows

```
tmp <- c(rnorm(30, 3), rnorm(30, -3))
data <- cbind(x = tmp, y=rev(tmp))
plot(data)
```



Run ‘kmeans()’ set K to 2 nstrt 20. The thing with kmeans is you have to tell it how many clusters you want

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

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

```
##           x         y
## 1 -3.252320  2.562103
## 2  2.562103 -3.252320
##
## 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 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] 58.88431 58.88431
##  (between_SS / total_SS =  89.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Q. how many points are in each cluster?

```r
km$size
```

```
## [1] 30 30
```

```r
km$size
```

```
## [1] 30 30
```

Q. What 'component' of your result object details cluster assignmnet/membership?

```r
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 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
```

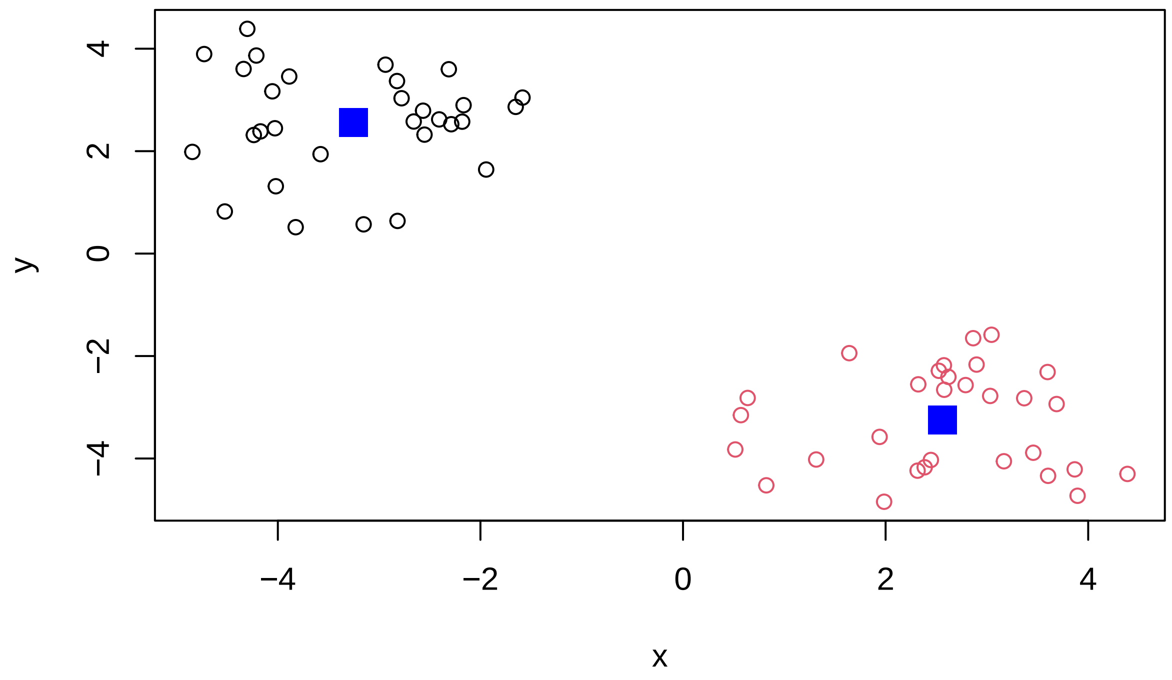
Q. What 'component' of your result object details cluster center?

```r
km$centers
```

```
##           x         y
## 1 -3.252320  2.562103
## 2  2.562103 -3.252320
```

Q. Plot x colored by the kmeans cluster assignmnet and add cluster centers as blue points? #cluster center

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



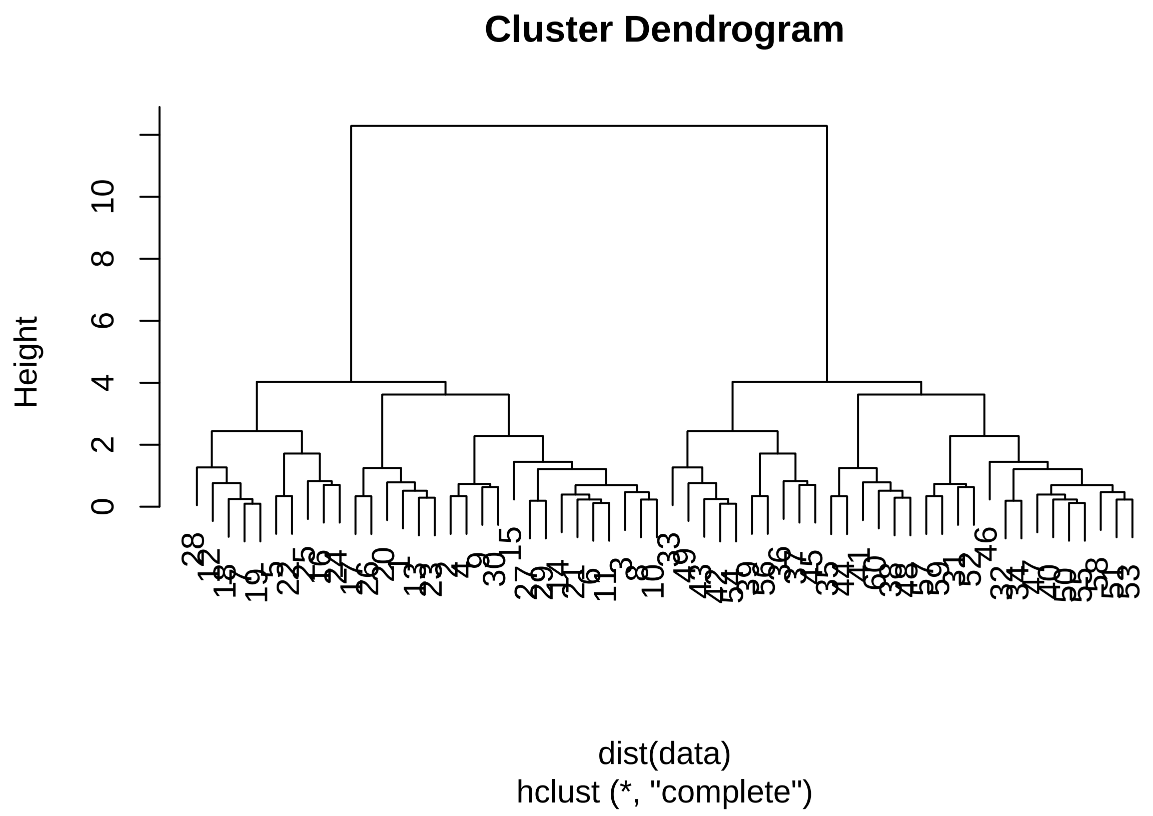
#hclust - hierarchicial clustering We will use the "hclust()" function on the same data as before and see how this method works

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

```
##
## Call:
## hclust(d = dist(data))
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 60
```

hclust has a plot method

```
plot(hc)
```

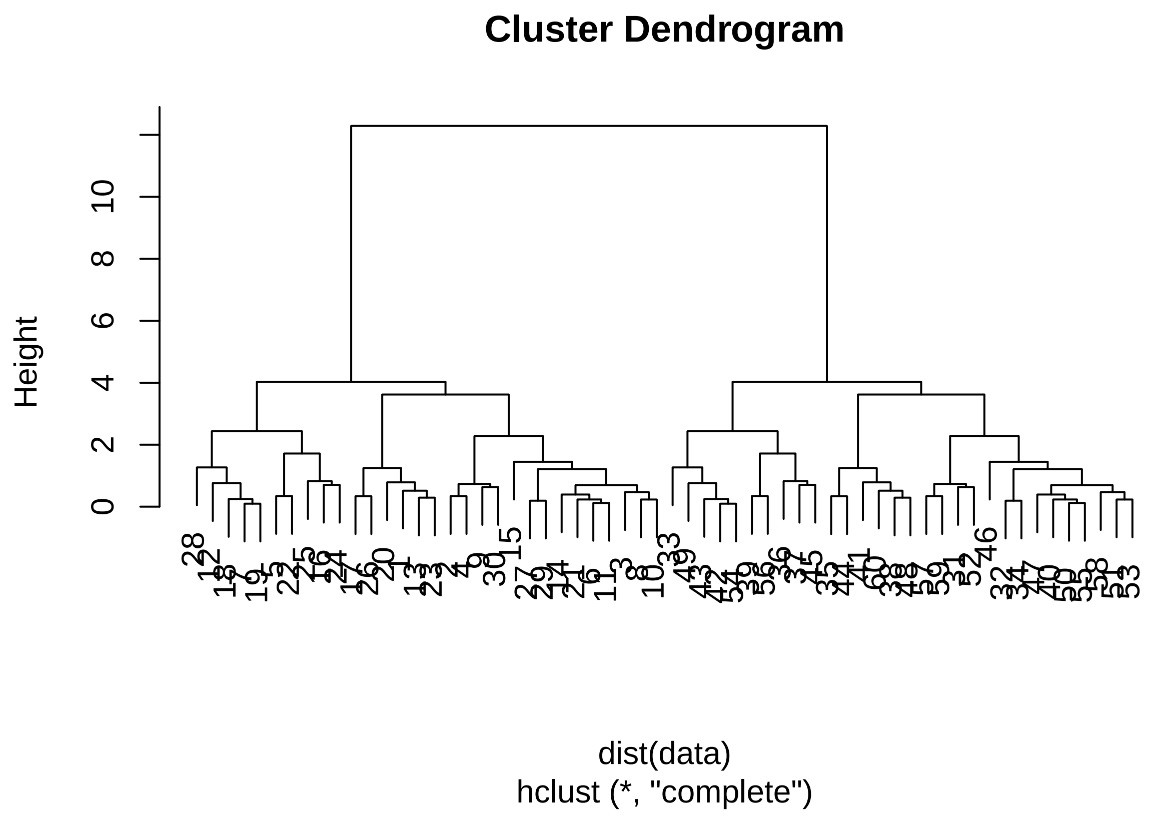


To find our membership vector we need to "cut" the tree and for this we use the 'cutree()' function and tell it the height to cut at.

```
cutree(hc, h=7)
```

```
##  [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
```

```
plot(hc)
```



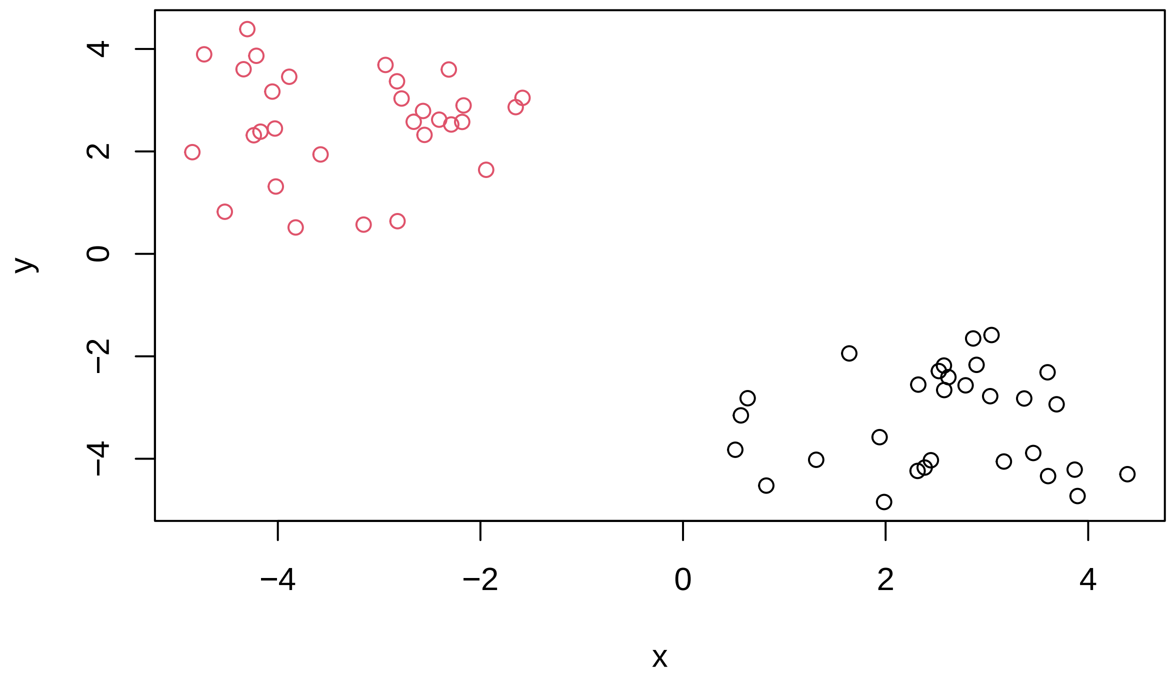
```
abline
```

```
## function (a = NULL, b = NULL, h = NULL, v = NULL, reg = NULL,
##     coef = NULL, untf = FALSE, ...)
## {
##     int_abline <- function(a, b, h, v, untf, col = par("col"),
##         lty = par("lty"), lwd = par("lwd"), ...) .External.graphics(C_abline,
##         a, b, h, v, untf, col, lty, lwd, ...)
##     if (!is.null(reg)) {
##         if (!is.null(a))
##             warning("'a' is overridden by 'reg'")
##         a <- reg
##     }
##     if (is.object(a) || is.list(a)) {
##         p <- length(coefa <- as.vector(coef(a)))
##         if (p > 2)
##             warning(gettextf("only using the first two of %d regression coefficients",
##                 p), domain = NA)
##         islm <- inherits(a, "lm")
##         noInt <- if (islm)
##             !as.logical(attr(stats::terms(a), "intercept"))
##         else p == 1
##         if (noInt) {
##             a <- 0
##             b <- coefa[1L]
##         }
##         else {
##             a <- coefa[1L]
##             b <- if (p >= 2)
##                 coefa[2L]
##             else 0
##         }
##     }
##     if (!is.null(coef)) {
##         if (!is.null(a))
##             warning("'a' and 'b' are overridden by 'coef'")
##         a <- coef[1L]
##         b <- coef[2L]
##     }
##     int_abline(a = a, b = b, h = h, v = v, untf = untf, ...)
##     invisible()
## }
## <bytecode: 0x7fc15b0ba758>
## <environment: namespace:graphics>
```

We can also use 'cutree()' and state the number of k clusters we want…

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

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



#princical component analysis (PCA)

PCA is a useful analysis method when you have lots of dimensions in your data…

##PCA of UK food data

Import the data from a CSV file

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

How many row and cols?

```
dim(x)
```

```
## [1] 17  5
```

#remove first col as this is not a col just the rows but it is seeing it as a col

```
x[,1]
```

```
##  [1] "Cheese"               "Carcass_meat "        "Other_meat "
##  [4] "Fish"                 "Fats_and_oils "       "Sugars"
##  [7] "Fresh_potatoes "      "Fresh_Veg "           "Other_Veg "
## [10] "Processed_potatoes "  "Processed_Veg "       "Fresh_fruit "
## [13] "Cereals "             "Beverages"            "Soft_drinks "
## [16] "Alcoholic_drinks "    "Confectionery "
```

```
rownames(x) <- x[,1]
x <- x[,-1]
x
```

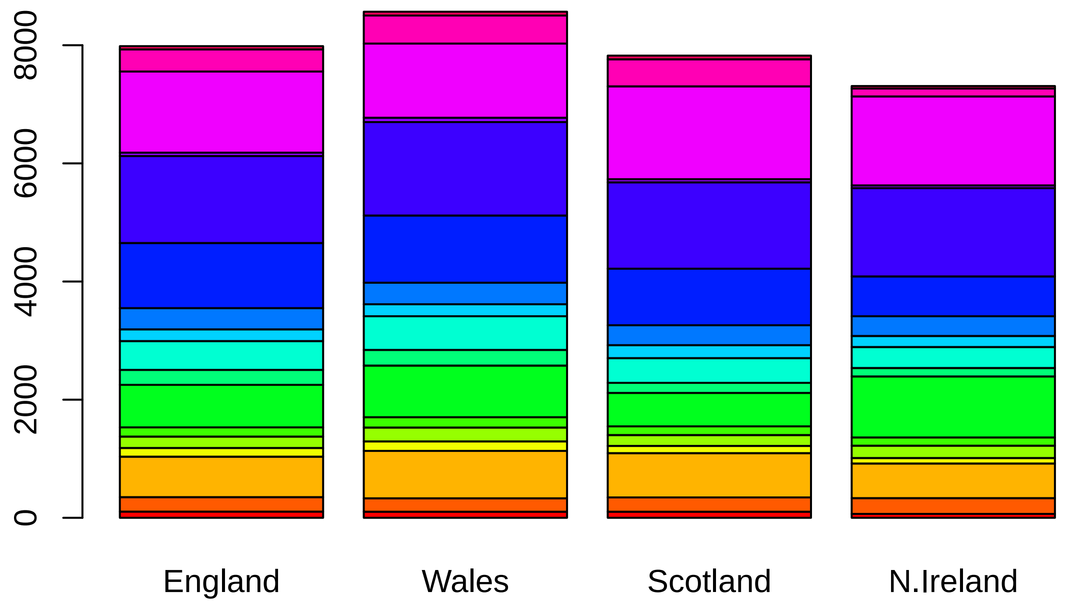
```
##                      England Wales Scotland N.Ireland
## Cheese                   105   103      103        66
## Carcass_meat             245   227      242       267
## Other_meat               685   803      750       586
## Fish                     147   160      122        93
## Fats_and_oils            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
```

#below is a better way to do this. above way - you will lose a col evertime you run it

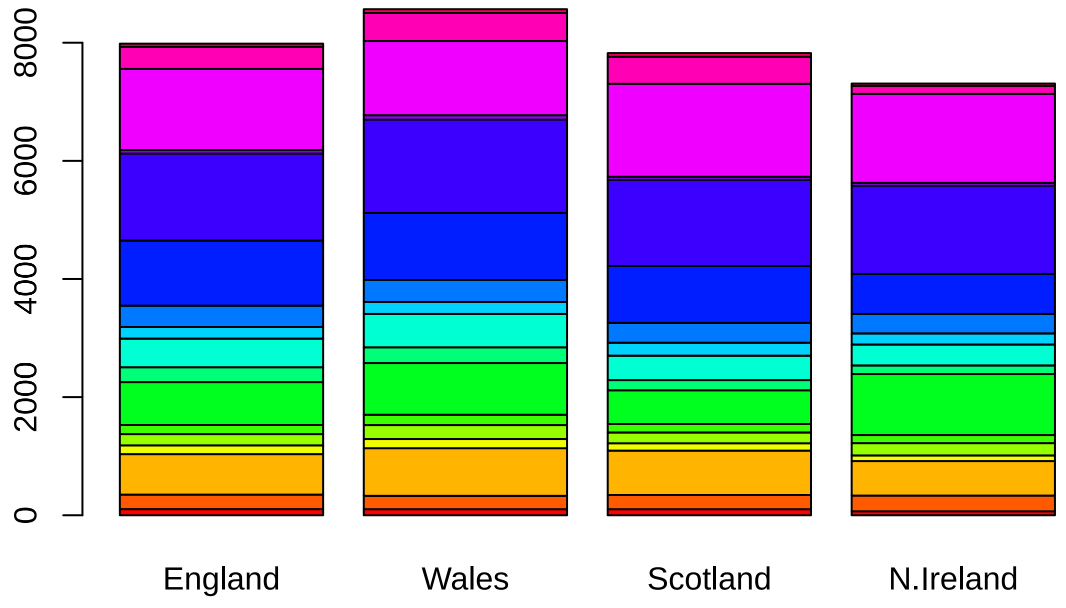
```r
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names = 1)
x
```

```
##                     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
## 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
```

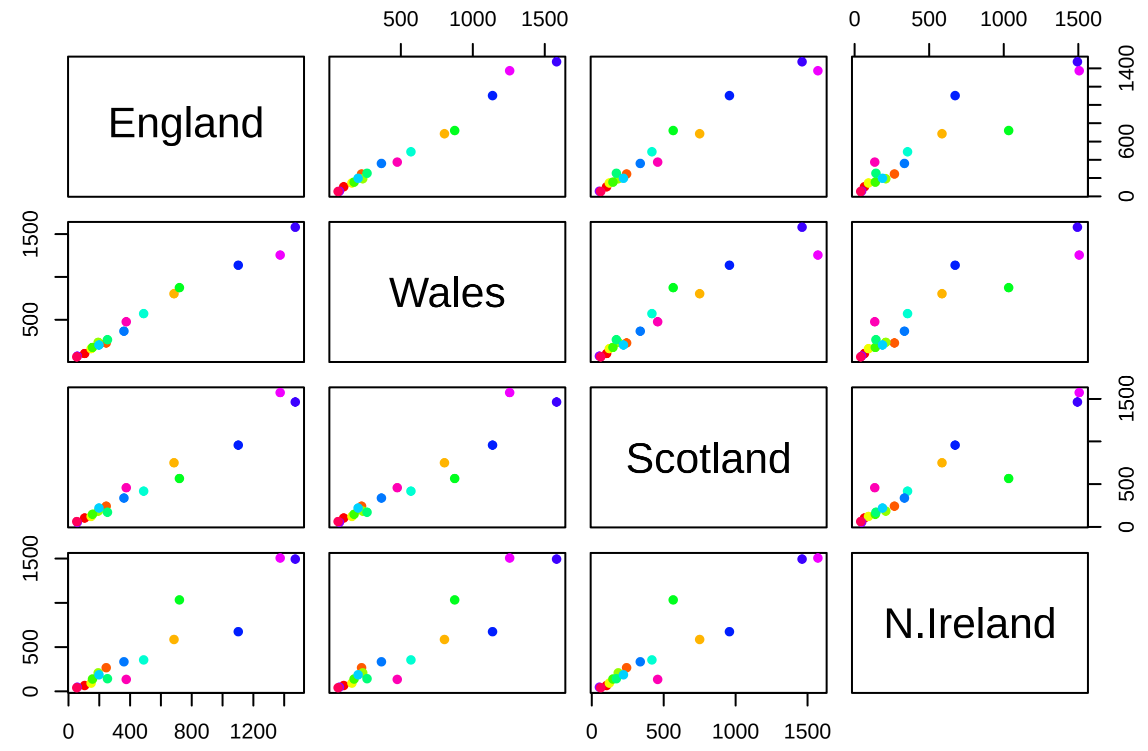
```r
barplot(as.matrix(x), col=rainbow(17))
```



```
barplot(as.matrix(x), col=rainbow(17))
```



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



##PCA to the rescue

Here we will use the base R function for PCA, which is called 'prcomp()' First transpose the data with "t" because 'prcomp()' wants the the rows. It says this in the help page

```
t(x)
```

```
##            Cheese Carcass_meat  Other_meat  Fish Fats_and_oils  Sugars
## England       105          245         685   147           193     156
## Wales         103          227         803   160           235     175
## Scotland      103          242         750   122           184     147
## N.Ireland      66          267         586    93           209     139
##            Fresh_potatoes  Fresh_Veg  Other_Veg  Processed_potatoes
## England               720        253        488                 198
## Wales                 874        265        570                 203
## Scotland              566        171        418                 220
## N.Ireland            1033        143        355                 187
##            Processed_Veg  Fresh_fruit  Cereals  Beverages Soft_drinks
## England              360         1102     1472         57        1374
## Wales                365         1137     1582         73        1256
## Scotland             337          957     1462         53        1572
## N.Ireland            334          674     1494         47        1506
##            Alcoholic_drinks  Confectionery
## England                 375             54
## Wales                   475             64
## Scotland                458             62
## N.Ireland               135             41
```

```
prcomp(x)
```

```
## Standard deviations (1, .., p=4):
## [1] 919.13914 132.06254  88.58981  28.62021
##
## Rotation (n x k) = (4 x 4):
##                 PC1        PC2         PC3         PC4
## England   0.4901572 -0.2808420  0.07019969  0.82215915
## Wales     0.4981704 -0.3475970  0.64010493 -0.47039166
## Scotland  0.5042356 -0.2482197 -0.76259165 -0.32029268
## N.Ireland 0.5072659  0.8594714  0.06157655 -0.01409337
```

```
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
```

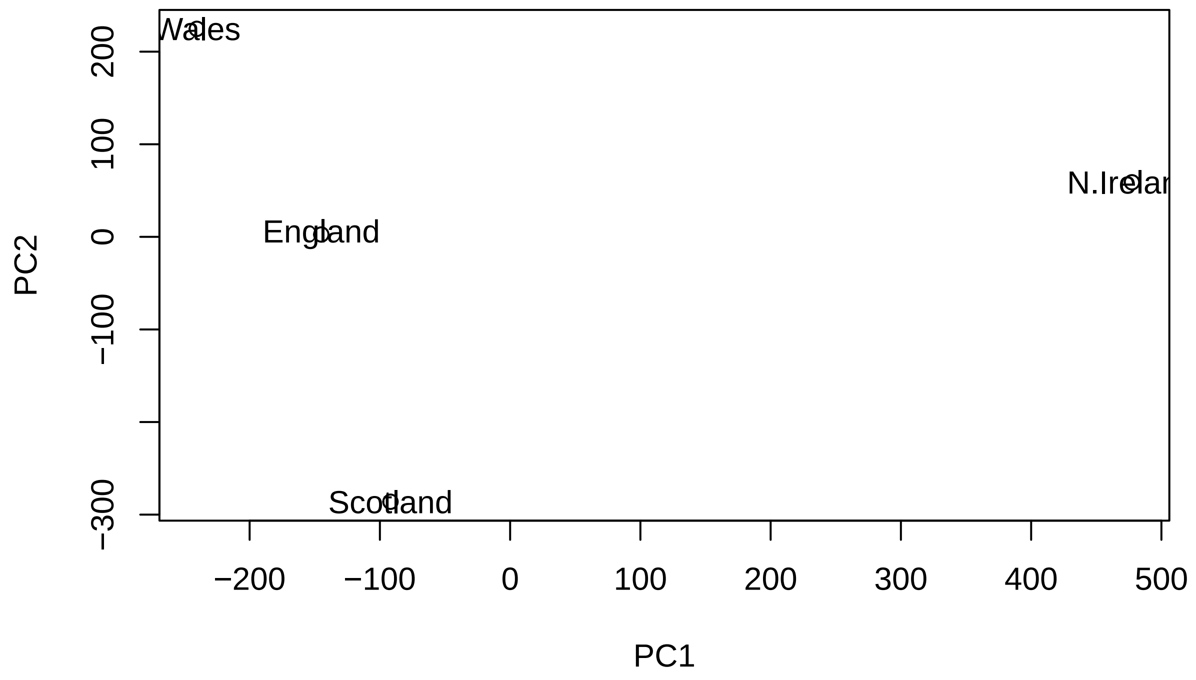
We want score plot (a.k.a PCA plot). Basically of PC1 vs PC2

```
attributes(pca)
```

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

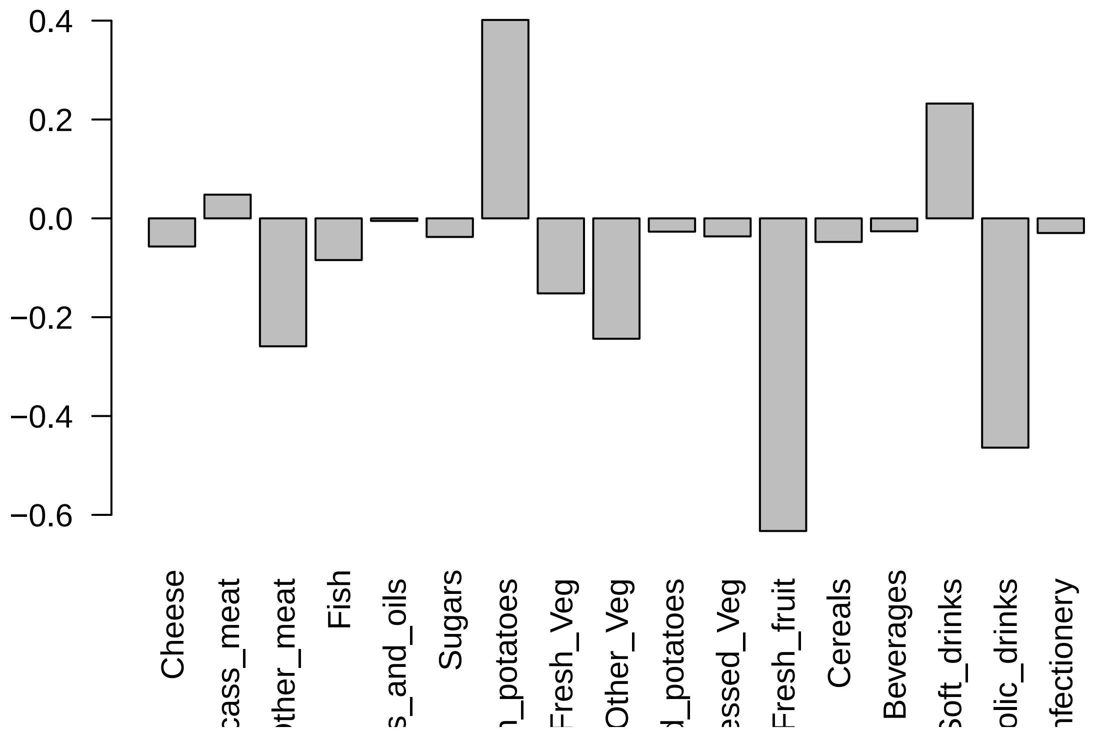
We are after the pca$x component for this plot

```
plot(pca$x[,1:2])
text(pca$x[,1:2], labels = colnames(x))
```



We can also examine the PCA "loadings", which tell us how much original variables contribute to each new PCA...

```
barplot(pca$rotation[,1], las=2)
```



RNASEQ

```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
```

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

```
## [1] 100
```

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

```
## [1] 10
```

```
colnames(rna.data)
```

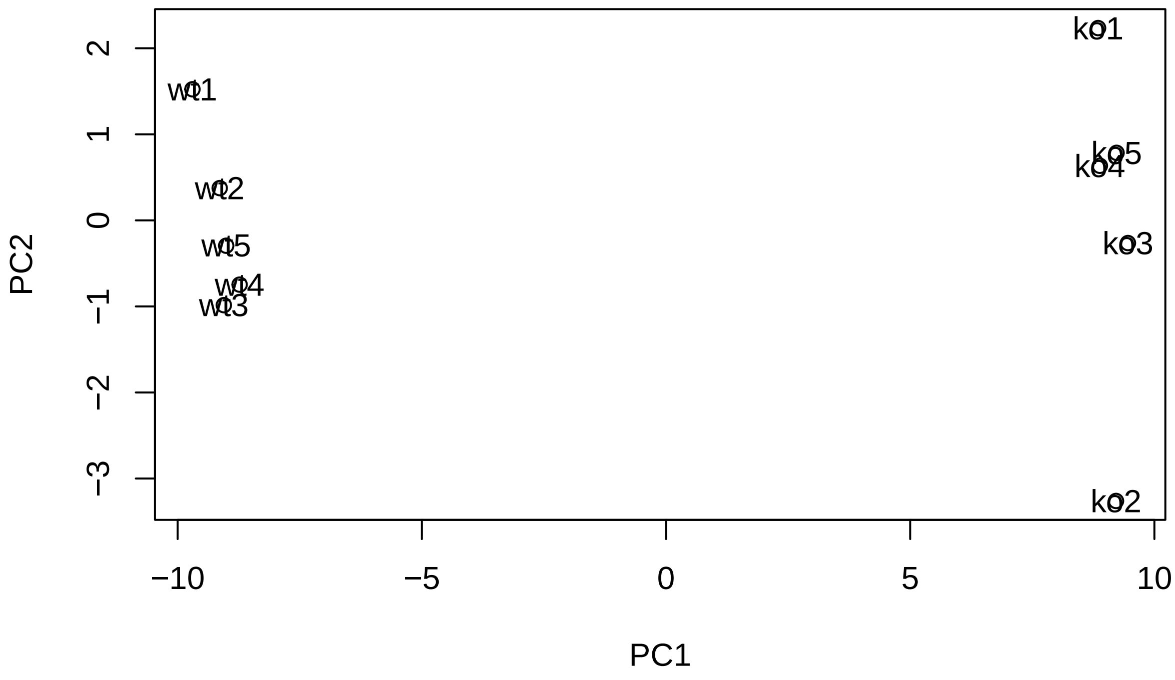
```
##  [1] "wt1" "wt2" "wt3" "wt4" "wt5" "ko1" "ko2" "ko3" "ko4" "ko5"
```

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

```
## 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
```

#sumamry above shows that PC2 in the plot below is not as significant as PC1. i.e. it is the short line on the graph

```
plot(pca.rna$x[,1:2])
text(pca.rna$x[,1:2], labels = colnames(rna.data))
```


ko1
wt1
ko5
ko4
wt2
ko3
wt5
wt4
wt3
ko2
PC1
PC2