Class 7: Machine Learning 1

Ani A16647613

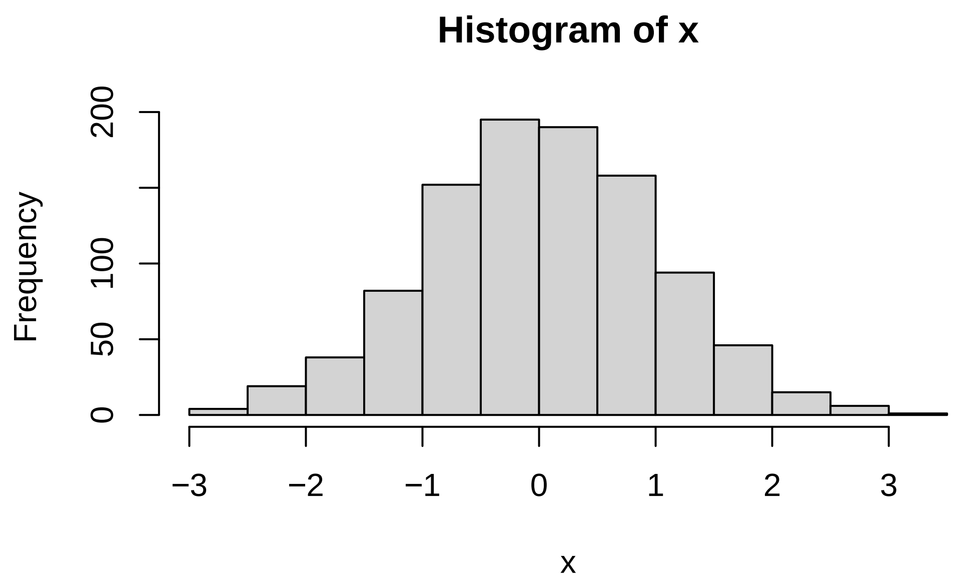
Clustering Methods

The broad goal here is to find groupings (clusters) in your input data.

##K means

First, lets make up some data to cluster.

```
x <- rnorm(1000)
hist(x)
```



Make a vector of length 60 with 30 points centered at -3 and 30 points at +3.

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

```
 [1] -4.627177 -3.026079 -2.597380 -1.932038 -2.037396 -4.159953 -2.144216
 [8] -3.773737 -1.611692 -4.031699 -2.356316 -2.769998 -3.023750 -4.282115
[15] -1.071738 -2.885831 -2.852391 -2.189565 -2.204870 -4.049411 -4.353746
[22] -2.583500 -4.045128 -4.028440 -3.786038 -4.574942 -3.688313 -3.259243
[29] -3.538056 -2.257877  3.050598  2.324934  3.129494  2.765599  3.083536
[36]  1.804232  3.359325  2.827801  2.318489  3.525650  1.802170  1.294771
[43]  3.031259  3.232475  2.411985  3.144426  3.410688  2.825867  3.074261
[50]  1.356501  2.698256  2.165933  2.787521  3.349106  2.316083  2.286532
[57]  4.969153  2.409445  3.580665  1.893275
```

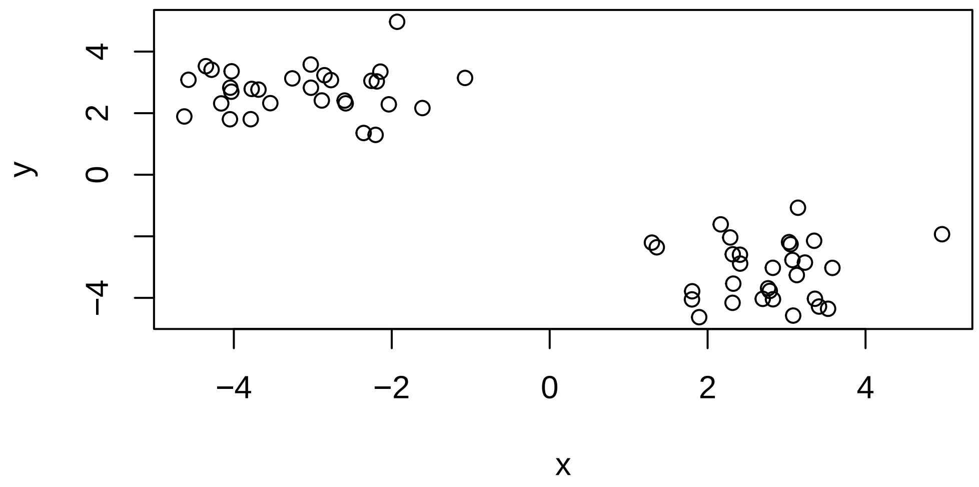
I will now make a smaller x and y dataset with 2 groups of points.

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

```
              x         y
 [1,] -4.627177  1.893275
 [2,] -3.026079  3.580665
 [3,] -2.597380  2.409445
 [4,] -1.932038  4.969153
 [5,] -2.037396  2.286532
 [6,] -4.159953  2.316083
 [7,] -2.144216  3.349106
 [8,] -3.773737  2.787521
 [9,] -1.611692  2.165933
[10,] -4.031699  2.698256
[11,] -2.356316  1.356501
[12,] -2.769998  3.074261
[13,] -3.023750  2.825867
[14,] -4.282115  3.410688
[15,] -1.071738  3.144426
[16,] -2.885831  2.411985
[17,] -2.852391  3.232475
[18,] -2.189565  3.031259
[19,] -2.204870  1.294771
[20,] -4.049411  1.802170
[21,] -4.353746  3.525650
[22,] -2.583500  2.318489
```

```
[23,] -4.045128  2.827801
[24,] -4.028440  3.359325
[25,] -3.786038  1.804232
[26,] -4.574942  3.083536
[27,] -3.688313  2.765599
[28,] -3.259243  3.129494
[29,] -3.538056  2.324934
[30,] -2.257877  3.050598
[31,]  3.050598 -2.257877
[32,]  2.324934 -3.538056
[33,]  3.129494 -3.259243
[34,]  2.765599 -3.688313
[35,]  3.083536 -4.574942
[36,]  1.804232 -3.786038
[37,]  3.359325 -4.028440
[38,]  2.827801 -4.045128
[39,]  2.318489 -2.583500
[40,]  3.525650 -4.353746
[41,]  1.802170 -4.049411
[42,]  1.294771 -2.204870
[43,]  3.031259 -2.189565
[44,]  3.232475 -2.852391
[45,]  2.411985 -2.885831
[46,]  3.144426 -1.071738
[47,]  3.410688 -4.282115
[48,]  2.825867 -3.023750
[49,]  3.074261 -2.769998
[50,]  1.356501 -2.356316
[51,]  2.698256 -4.031699
[52,]  2.165933 -1.611692
[53,]  2.787521 -3.773737
[54,]  3.349106 -2.144216
[55,]  2.316083 -4.159953
[56,]  2.286532 -2.037396
[57,]  4.969153 -1.932038
[58,]  2.409445 -2.597380
[59,]  3.580665 -3.026079
[60,]  1.893275 -4.627177
```

```
plot(x)
```



```r
k <- kmeans(x, centers=2)
k
```

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

Cluster means:
          x         y
1  2.741001 -3.124754
2 -3.124754  2.741001

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 1 1

Within cluster sum of squares by cluster:
[1] 43.21223 43.21223
 (between_SS / total_SS =  92.3 %)

Available components:

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

Q. From your result object `k` how many points are in each cluster?

```
k$size
```

```
[1] 30 30
```

Q. What "component" of your result object details the cluster membership?

```
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 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
```

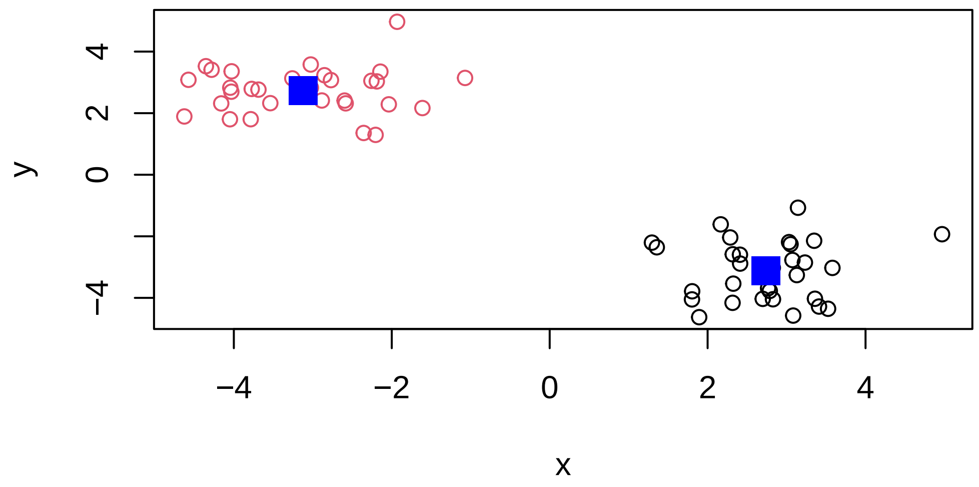
Q. Cluster centers?

```
k$centers
```

```
          x         y
1  2.741001 -3.124754
2 -3.124754  2.741001
```

Q. Plot of our clustering results

```
plot(x, col= k$cluster)
points(k$centers, col="blue", pch=15, cex=2)
```



Now to make it into 4 groups.

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

```
K-means clustering with 4 clusters of sizes 30, 10, 7, 13

Cluster means:
          x         y
1 -3.124754  2.741001
2  3.338730 -2.452689
3  2.034808 -2.325283
4  2.661467 -4.072212

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 4 2 4 4 4 4 4
[39] 3 4 4 3 2 2 3 2 4 2 2 3 4 3 4 2 4 3 2 3 2 4

Within cluster sum of squares by cluster:
[1] 43.212230  7.266583  2.513883  5.625190
 (between_SS / total_SS =  94.8 %)
```

```
Available components:

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

```
k4$size
```

```
[1] 30 10  7 13
```

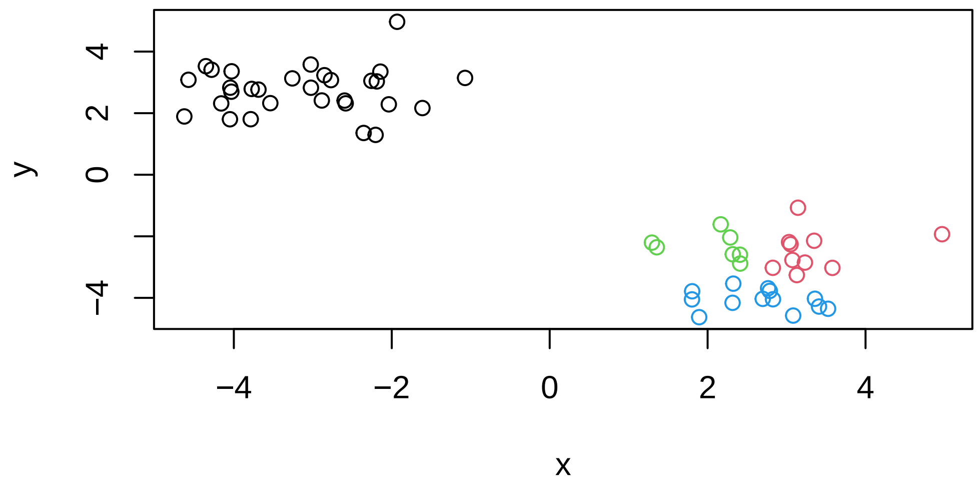
```
k4$cluster
```

```
 [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 4 2 4 4 4 4 4
[39] 3 4 4 3 2 2 3 2 4 2 2 3 4 3 4 2 4 3 2 3 2 4
```

```
k4$centers
```

```
          x         y
1 -3.124754  2.741001
2  3.338730 -2.452689
3  2.034808 -2.325283
4  2.661467 -4.072212
```

```
plot(x, col=k4$cluster)
```



A big limitation of kmeans is that it does what you ask even if you ask for silly clusters.

Hierarchial Clustering

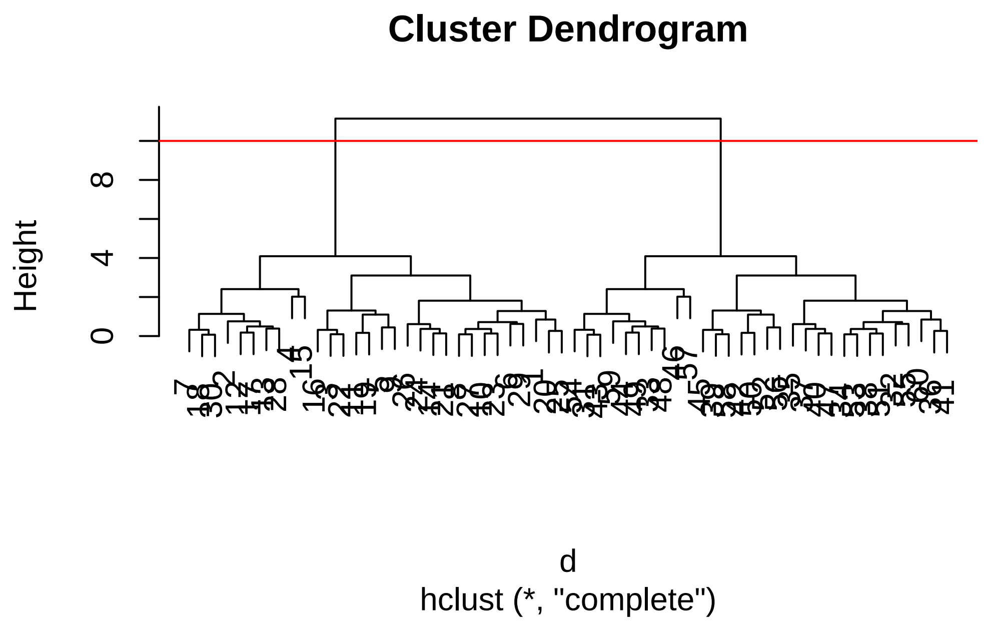
The main base R function for Hierarchial Clustering is `hclust()`. Unlike `kmeans()` you cannot just pass in your data as an input. You first need to calculate a distance matrix.

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

```

Call:
hclust(d = d)

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

Use `plot()` to view the results

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



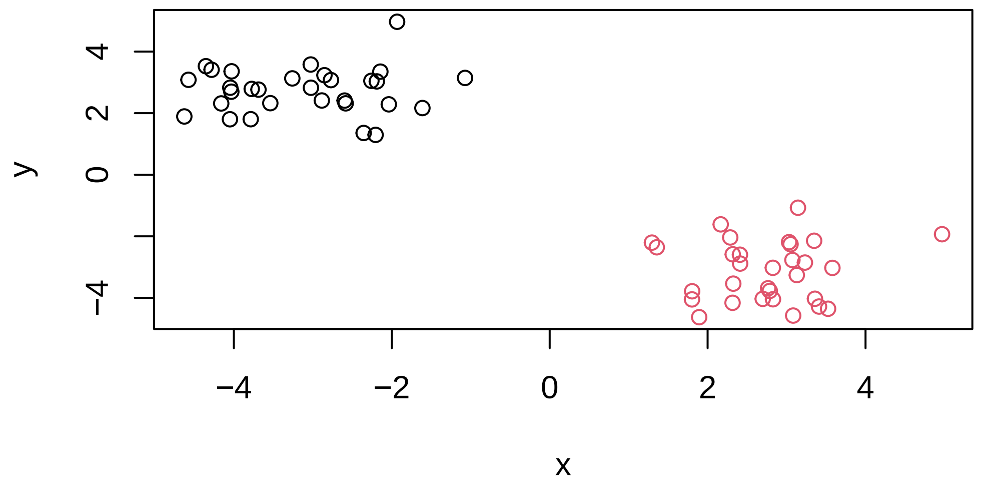
To make the "cut" and get our cluster membership vector we can use the `cutree()` function.

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

```
 [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 2 2
```

Make a plot of our data colored by hclust results

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



Principal Component Analysis (PCA)

Here we will do Principal Component Analysis (PCA) on some food data from the UK.

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

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

#not a good method, overrides the functions
```

> Q. 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
```

```
#or
dim(x)
```

```
[1] 17  4
```

Preview the first six rows

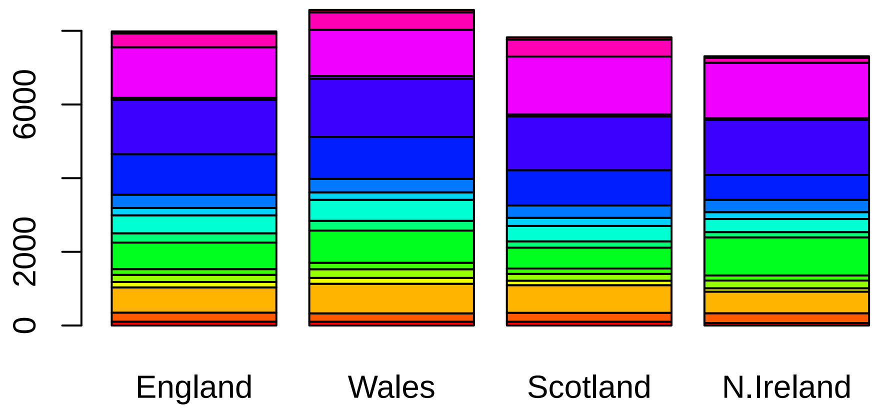
```
head(x)
```

```
               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
```

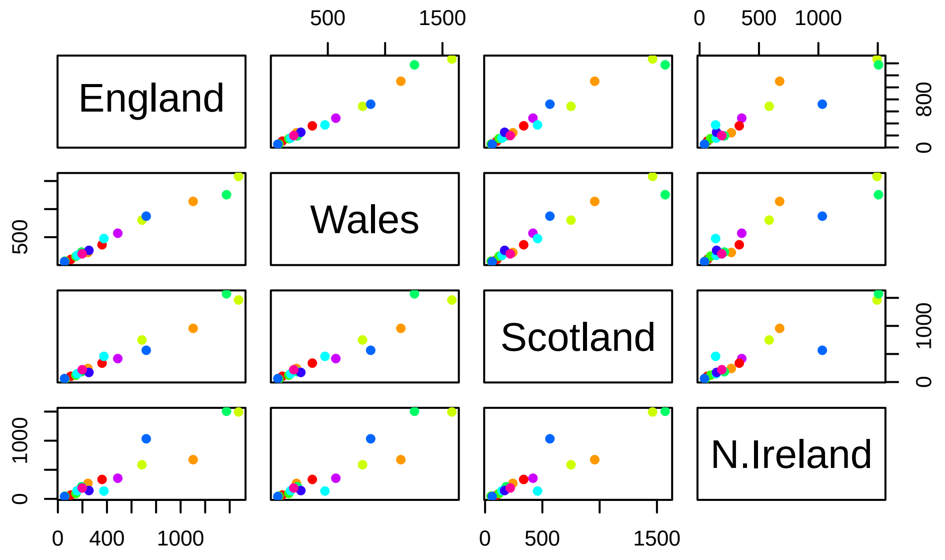
> Q2. Which approach to solving the `row-name problems` mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

I would prefer to use the `dim()` or `nrow()` function and then adding `head()` as a second function, since the x <- x[,-1] function reloads and overrides functions so it is not as reliable as `dim()`.

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



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



##PCA to the rescue

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

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

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 3.176e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

> Q. How much variance is captured in 2 PCs?

96.5%

To make our main "PC score plot" (aka "PC1 vs PC2 plot" or "PC plot" or "ordination plot").

```
attributes(pca)
```

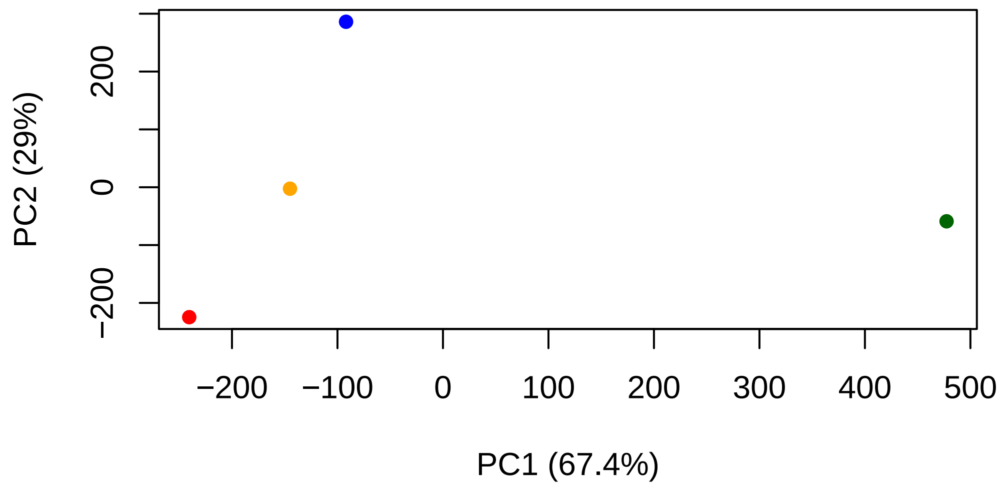
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

We are after the `pca$x` result component to make our main PCA plot.

```
pca$x
```

```
                 PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -4.894696e-14
Wales     -240.52915 -224.646925 -56.475555  5.700024e-13
Scotland   -91.86934  286.081786 -44.415495 -7.460785e-13
N.Ireland  477.39164  -58.901862  -4.877895  2.321303e-13
```

```
mycols <- c("orange", "red", "blue", "darkgreen")
plot(pca$x[,1], pca$x[,2], col=mycols, pch=16, xlab="PC1 (67.4%)", ylab="PC2 (29%)")
```



Another important result from PCA is how the original variables (in this case the foods) contribute to the PCs.

This is contained in the `pca$rotation` object - called the "loadings" or "contributions" to the PCs.

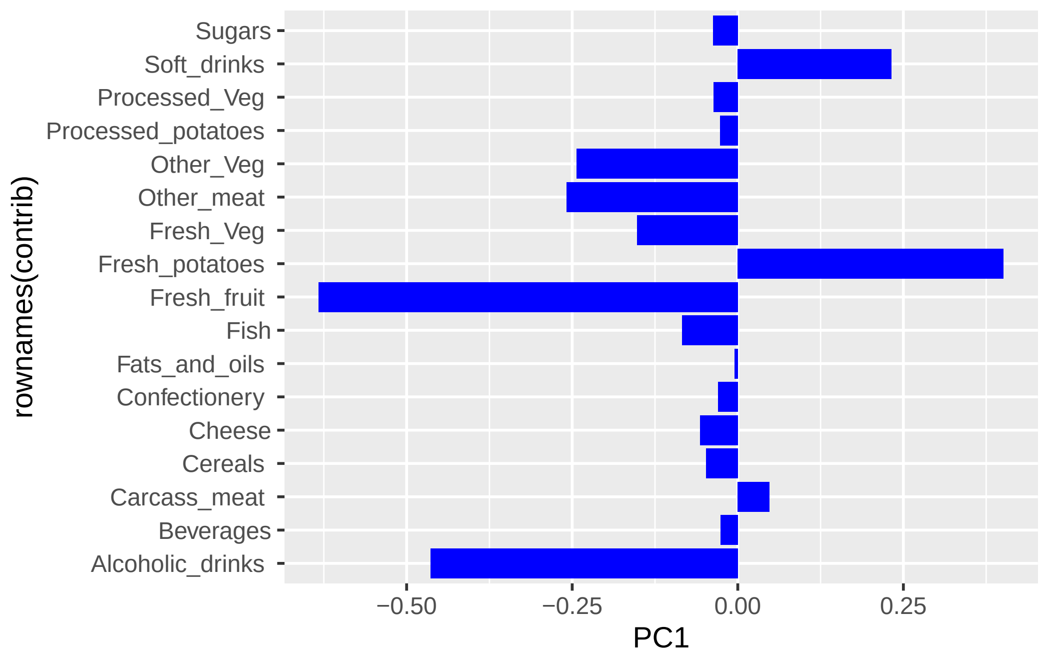
```
pca$rotation
```

```
                            PC1          PC2         PC3          PC4
Cheese              -0.056955380  0.016012850  0.02394295 -0.694538519
Carcass_meat         0.047927628  0.013915823  0.06367111  0.489884628
Other_meat          -0.258916658 -0.015331138 -0.55384854  0.279023718
Fish                -0.084414983 -0.050754947  0.03906481 -0.008483145
Fats_and_oils       -0.005193623 -0.095388656 -0.12522257  0.076097502
Sugars              -0.037620983 -0.043021699 -0.03605745  0.034101334
Fresh_potatoes       0.401402060 -0.715017078 -0.20668248 -0.090972715
Fresh_Veg           -0.151849942 -0.144900268  0.21382237 -0.039901917
Other_Veg           -0.243593729 -0.225450923 -0.05332841  0.016719075
Processed_potatoes  -0.026886233  0.042850761 -0.07364902  0.030125166
Processed_Veg       -0.036488269 -0.045451802  0.05289191 -0.013969507
Fresh_fruit         -0.632640898 -0.177740743  0.40012865  0.184072217
Cereals             -0.047702858 -0.212599678 -0.35884921  0.191926714
```

```
Beverages           -0.026187756 -0.030560542 -0.04135860  0.004831876
Soft_drinks          0.232244140  0.555124311 -0.16942648  0.103508492
Alcoholic_drinks    -0.463968168  0.113536523 -0.49858320 -0.316290619
Confectionery       -0.029650201  0.005949921 -0.05232164  0.001847469
```

Now to make a plot around PC1

```r
library(ggplot2)

contrib <- as.data.frame(pca$rotation)

ggplot(contrib) +
  aes(PC1, rownames(contrib))+ geom_col(fill="blue")
```


Sugars
Soft_drinks
Processed_Veg
Processed_potatoes
Other_Veg
Other_meat
Fresh_Veg
Fresh_potatoes
Fresh_fruit
Fish
Fats_and_oils
Confectionery
Cheese
Cereals
Carcass_meat
Beverages
Alcoholic_drinks
rownames(contrib)
PC1