Class 7: Machine Learning 1

Lilia Jimenez (PID:A16262599)

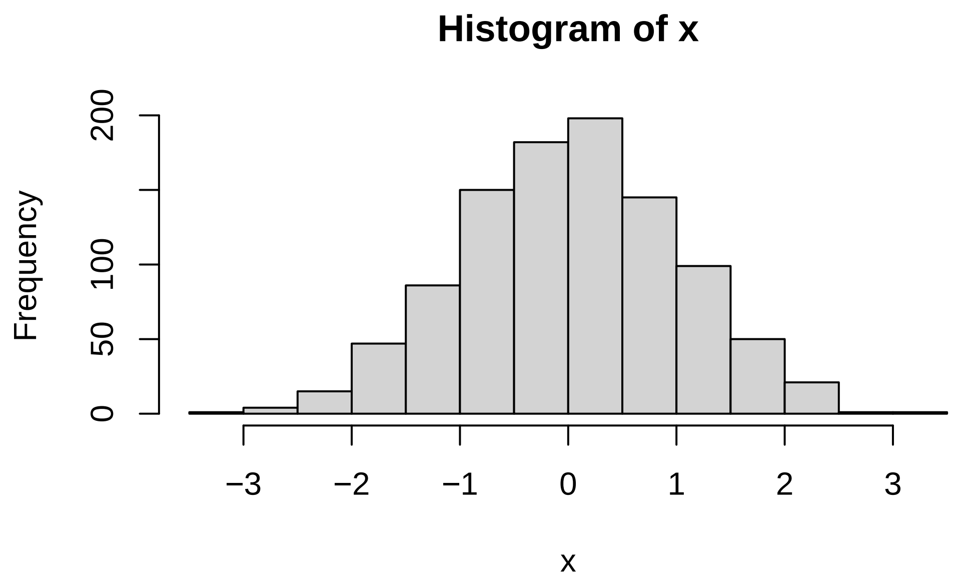
2024-01-30

#CLUSTERING METHODS The broad goal here is to find groupings (clusters) in your input data

##Kmeans

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 centered at +3

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

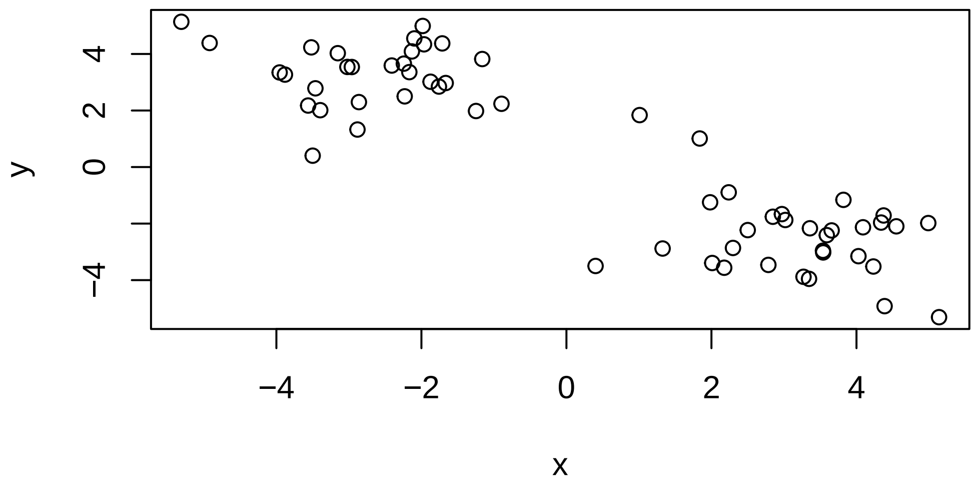
I will now make an x and y dataset with 2 groups of points

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

```
               x          y
 [1,] -0.8954690  2.2379458
 [2,] -1.9822267  4.9903837
 [3,] -2.2306257  2.5003111
 [4,] -3.5185138  4.2328734
 [5,] -2.1666392  3.3579368
 [6,] -4.9207311  4.3877801
 [7,] -3.0210995  3.5407611
 [8,] -1.1608206  3.8203228
 [9,] -1.7126827  4.3740536
[10,] -3.4617375  2.7849770
[11,] -1.7585610  2.8464933
[12,] -2.1312152  4.0897340
[13,] -2.8819893  1.3263860
[14,] -2.9600204  3.5382516
[15,] -3.9553219  3.3458462
[16,] -2.0976765  4.5494848
[17,] -5.3111815  5.1389394
[18,] -3.5621290  2.1757020
[19,] -1.2471528  1.9815847
[20,] -3.4998367  0.4017925
[21,] -2.2425046  3.6576809
[22,] -2.4088333  3.5906341
[23,] -3.1534786  4.0272653
[24,] -1.9640569  4.3402315
[25,] -2.8621716  2.2968796
[26,] -3.3953630  2.0108646
[27,] -1.6651982  2.9713095
[28,] -3.8819205  3.2694195
[29,] -1.8740991  3.0180333
[30,]  1.0093213  1.8378191
[31,]  1.8378191  1.0093213
[32,]  3.0180333 -1.8740991
[33,]  3.2694195 -3.8819205
[34,]  2.9713095 -1.6651982
```

```
[35,]  2.0108646 -3.3953630
[36,]  2.2968796 -2.8621716
[37,]  4.3402315 -1.9640569
[38,]  4.0272653 -3.1534786
[39,]  3.5906341 -2.4088333
[40,]  3.6576809 -2.2425046
[41,]  0.4017925 -3.4998367
[42,]  1.9815847 -1.2471528
[43,]  2.1757020 -3.5621290
[44,]  5.1389394 -5.3111815
[45,]  4.5494848 -2.0976765
[46,]  3.3458462 -3.9553219
[47,]  3.5382516 -2.9600204
[48,]  1.3263860 -2.8819893
[49,]  4.0897340 -2.1312152
[50,]  2.8464933 -1.7585610
[51,]  2.7849770 -3.4617375
[52,]  4.3740536 -1.7126827
[53,]  3.8203228 -1.1608206
[54,]  3.5407611 -3.0210995
[55,]  4.3877801 -4.9207311
[56,]  3.3579368 -2.1666392
[57,]  4.2328734 -3.5185138
[58,]  2.5003111 -2.2306257
[59,]  4.9903837 -1.9822267
[60,]  2.2379458 -0.8954690
```

```r
plot(x)
```



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

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

Cluster means:
          x         y
1  3.221390 -2.563798
2 -2.563798  3.221390

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] 81.55993 81.55993
 (between_SS / total_SS =  86.0 %)

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 results 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
```

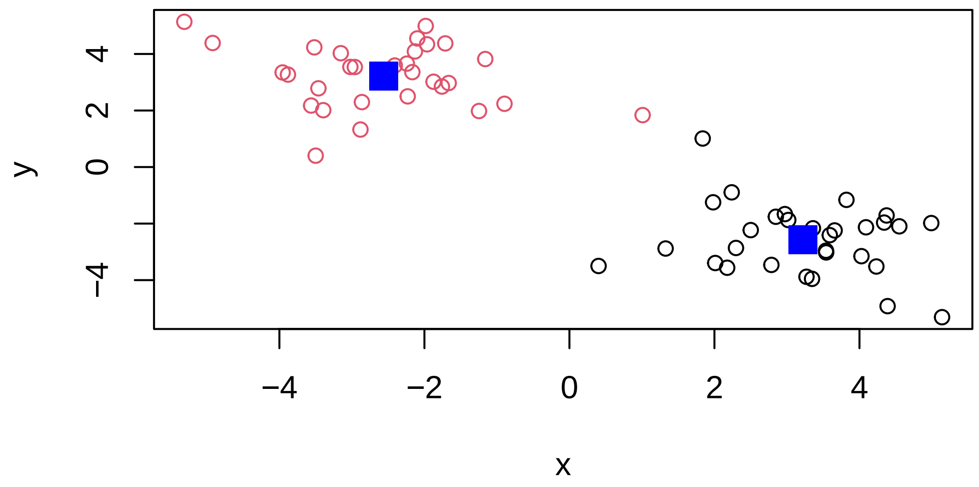
q. cluster centers?

```
k$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
```

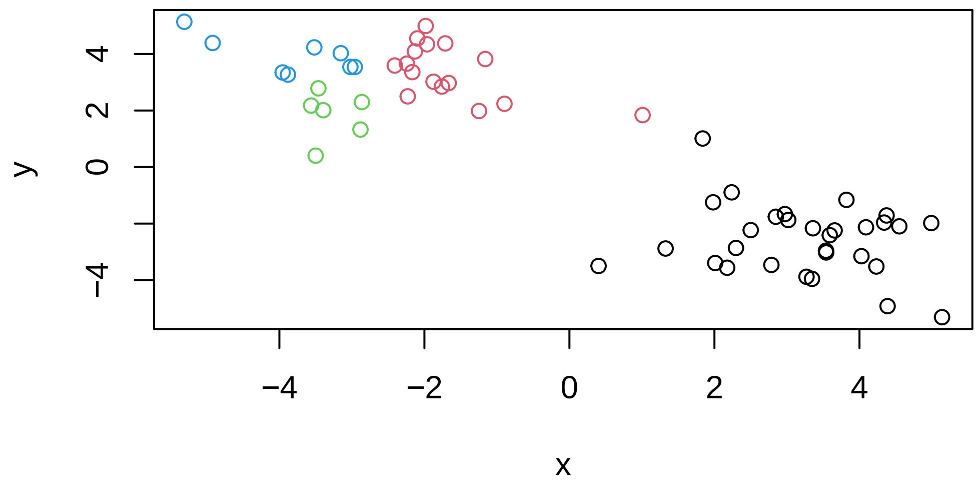
Q. Plot of our cluster

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



We can cluster into 4 groups

```
#kmeans
k4<-kmeans(x, center=4)
#Plot results
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

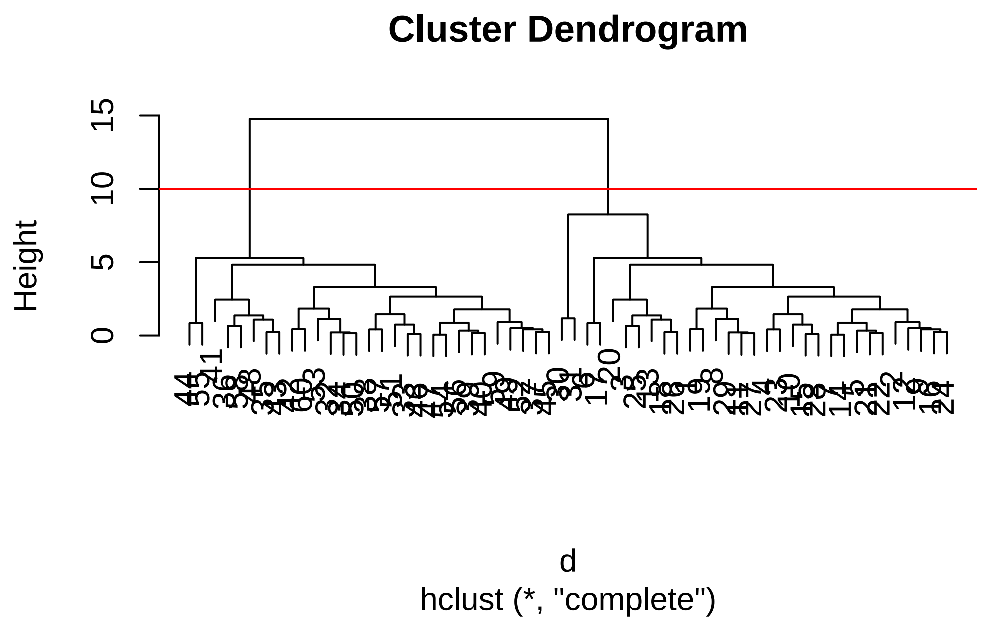
The main base R function for hierarchical clustering is 'hclust()'. Unlike 'kmeans()' you cant just pass it your data as input. you first need to calculate a distance matrix.

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

```
Call:
hclust(d = d)

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

use 'plot()' to view results

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



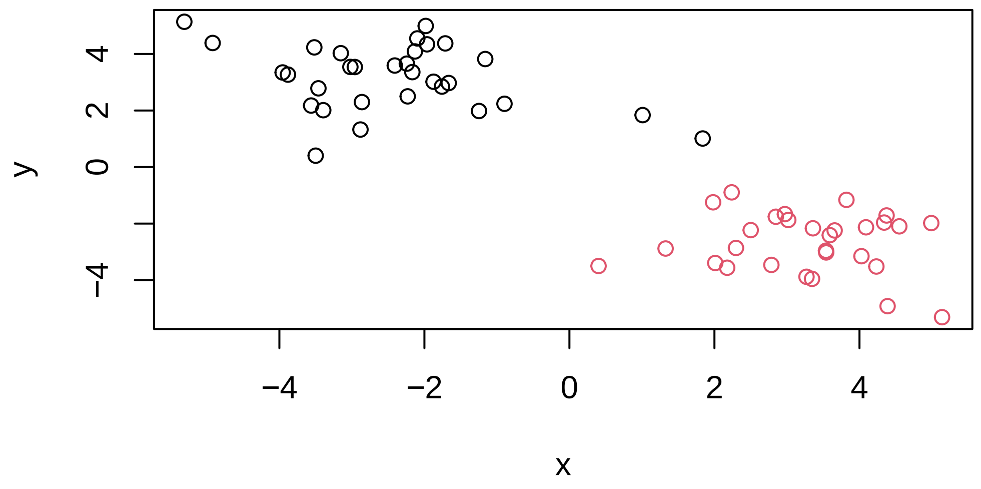
To make the "cut" and get our cluster membership number we can use the cutree()function

```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 data from the UK.

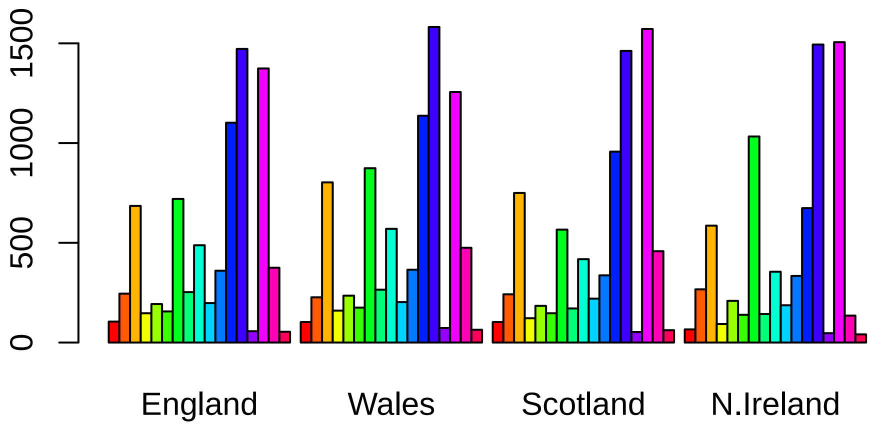
```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names = 1)
head(x)
```

```
               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
```

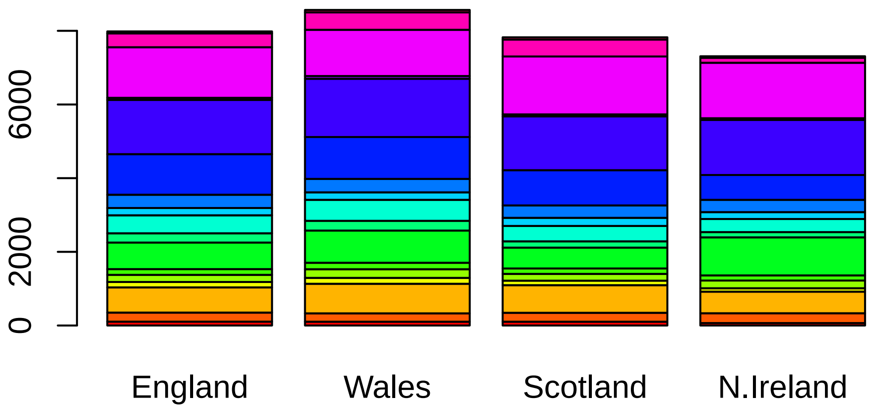
```
#rownames(x)<-x[,-1]
#x<-x[,-1]
#x
```

Spotting differences

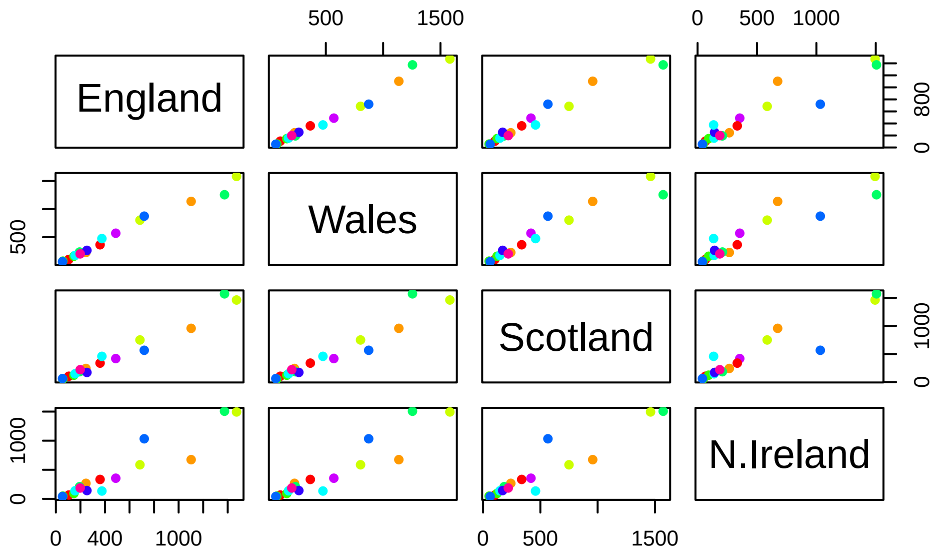
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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



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



PCA to the rescue

the main base “r” function for PCA is called ‘prcomp()’. here we need to switch the columns and rows by using ‘t()’

```
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
```

Q. How much variance was captured in 2 PCs

96.5%

To make our main “PC score plot” (a.k.a “PC1 VS PC2 plot” or “PC plot” or “ordination plot”)

```
attributes(pca)
```

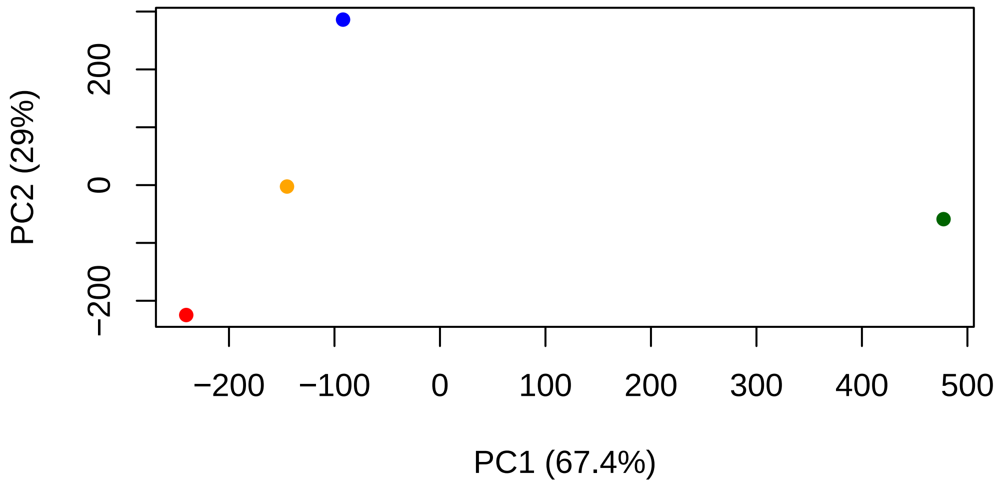
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$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 -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
```

```r
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- often called the "loadings" or "contributions" to the PCs.

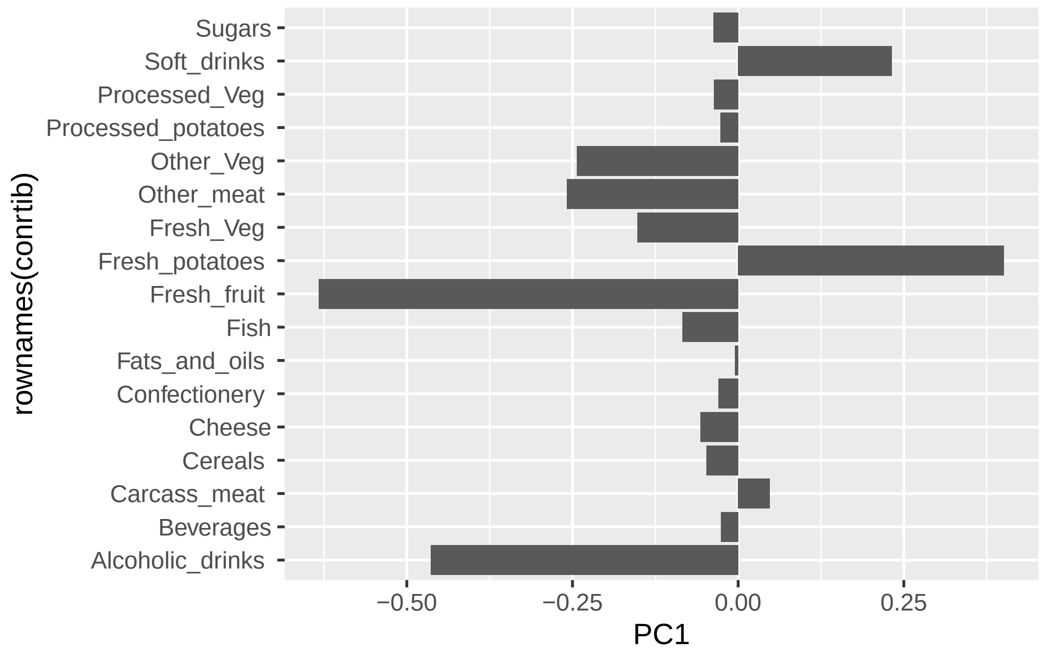
```r
pca$rotation
```

```
                       PC1          PC2         PC3          PC4
Cheese         -0.056955380  0.016012850  0.02394295 -0.409382587
Carcass_meat    0.047927628  0.013915823  0.06367111  0.729481922
Other_meat     -0.258916658 -0.015331138 -0.55384854  0.331001134
Fish           -0.084414983 -0.050754947  0.03906481  0.022375878
Fats_and_oils  -0.005193623 -0.095388656 -0.12522257  0.034512161
Sugars         -0.037620983 -0.043021699 -0.03605745  0.024943337
Fresh_potatoes  0.401402060 -0.715017078 -0.20668248  0.021396007
Fresh_Veg      -0.151849942 -0.144900268  0.21382237  0.001606882
Other_Veg      -0.243593729 -0.225450923 -0.05332841  0.031153231
```

```
Processed_potatoes  -0.026886233  0.042850761 -0.07364902 -0.017379680
Processed_Veg       -0.036488269 -0.045451802  0.05289191  0.021250980
Fresh_fruit         -0.632640898 -0.177740743  0.40012865  0.227657348
Cereals             -0.047702858 -0.212599678 -0.35884921  0.100043319
Beverages           -0.026187756 -0.030560542 -0.04135860 -0.018382072
Soft_drinks          0.232244140  0.555124311 -0.16942648  0.222319484
Alcoholic_drinks    -0.463968168  0.113536523 -0.49858320 -0.273126013
Confectionery       -0.029650201  0.005949921 -0.05232164  0.001890737
```

We can make a plot along pc1

```r
library(ggplot2)

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

ggplot(conrtib)+
  aes(PC1, rownames(conrtib))+
  geom_col()
```


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(conrtib)
PC1