class 7: machine learning

annmarie (A16442048)

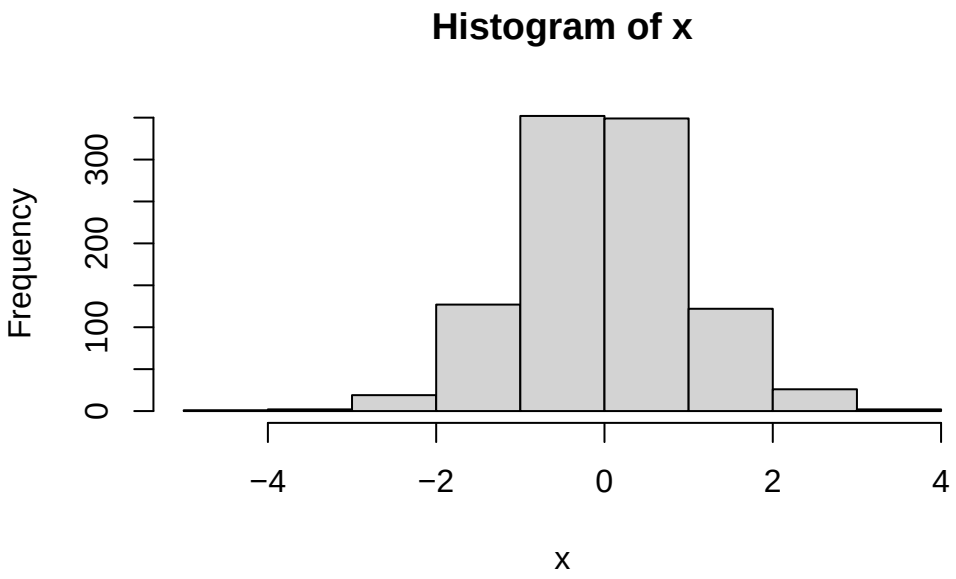
#Clustering Machine

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

##Kmeans

First let's make up some data to cluster

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



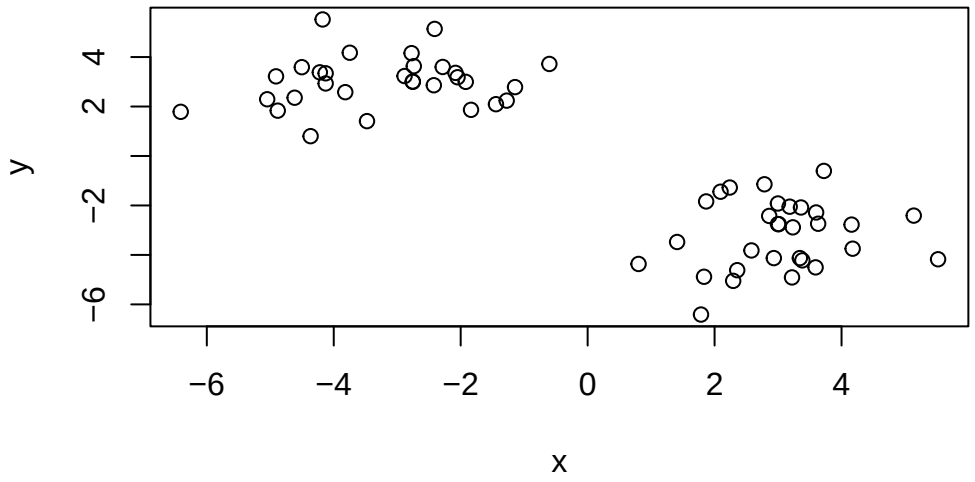
Make a vector of length 60 with 30 points centered at -3 and 30 points centered at +3

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

```
 [1] -4.1750299 -4.1278625 -4.9094769 -6.4107414 -2.0819699 -3.4762160
 [7] -4.1274236 -3.8183303 -2.8846979 -3.7471283 -1.2748656 -2.7544763
[13] -2.4230582 -2.7387696 -2.2832829 -0.6028947 -2.7547609 -1.8370850
[19] -4.6158747 -4.5031047 -4.8821544 -2.0494528 -2.4103399 -4.2194639
[25] -1.1438706 -1.4442140 -4.3647431 -1.9199153 -2.7737390 -5.0481598
[31]  2.2933821  4.1581031  2.9981996  0.8020326  2.0939172  2.7854799
[37]  3.3827543  5.1378849  3.1836089  1.8334374  3.5912891  2.3567267
[43]  1.8673270  2.9991467  3.7217181  3.5998720  3.6318973  2.8608169
[49]  3.0114606  2.2391972  4.1754539  3.2326960  2.5806761  2.9334324
[55]  1.4095896  3.3612721  1.7862344  3.2201723  3.3435324  5.5210955
```

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

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



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

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

Cluster means:
          x         y
1  3.003747 -3.193437
2 -3.193437  3.003747

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] 85.63941 85.63941
 (between_SS / total_SS =  87.1 %)

Available components:

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

Q. From your result project `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 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
```

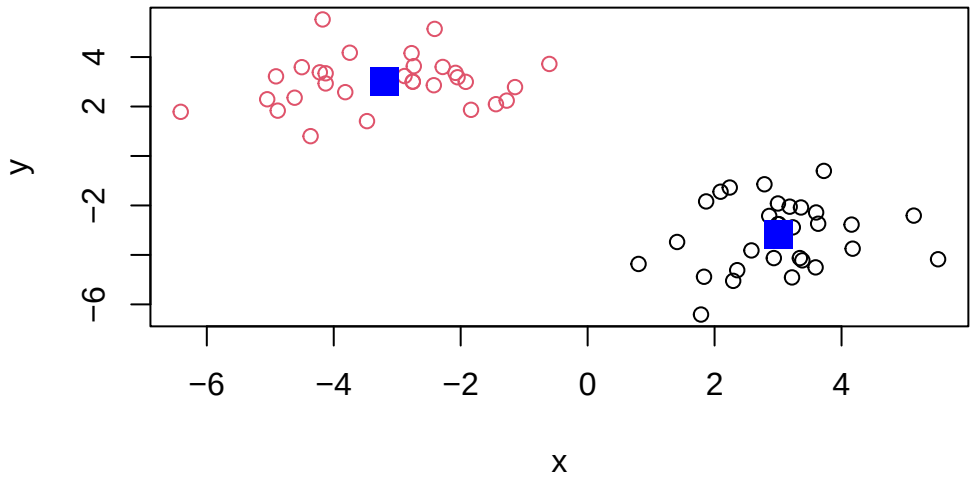
Q.Cluster centers?

```
k$centers
```

```
          x         y
1  3.003747 -3.193437
2 -3.193437  3.003747
```

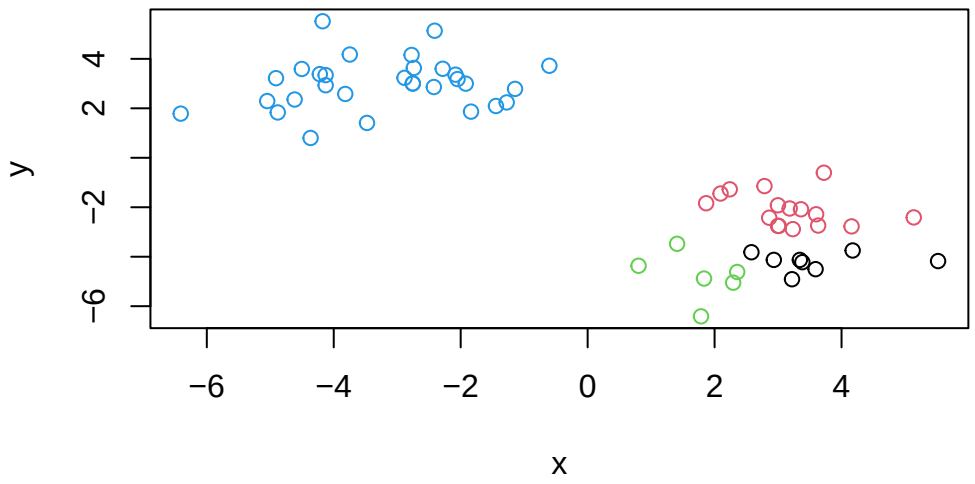
Q.Plot of our clustering results

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



We can cluster into 4 grps

```
#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.

##Hierarchical Clustering

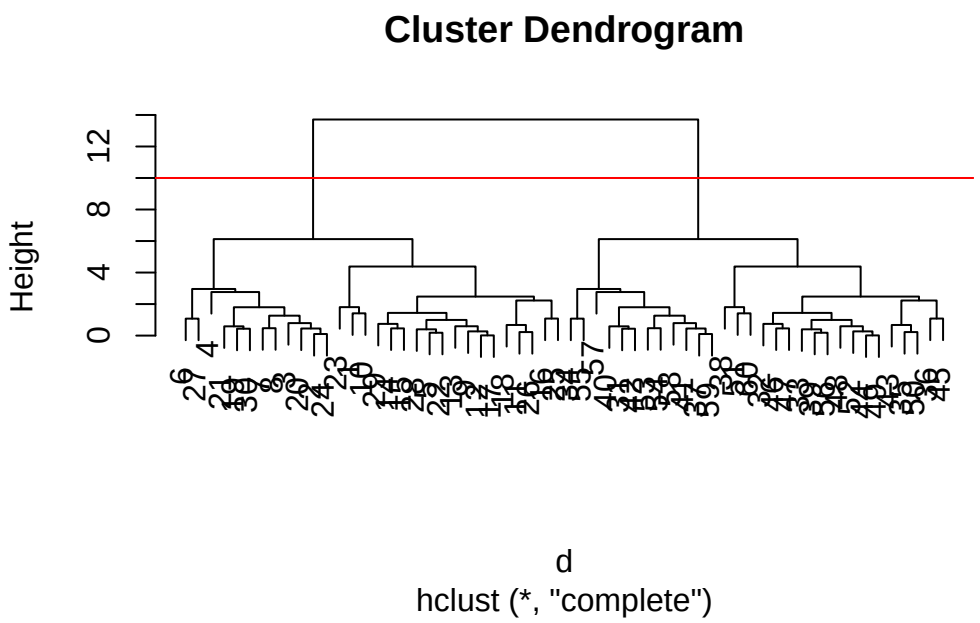
The main base R fucntion for Hierarchical CLustering is `hclust()`.Unlike `kmeans()`you can not just pass it your data as input. You first need to calculate a distance matrix.

```
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

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



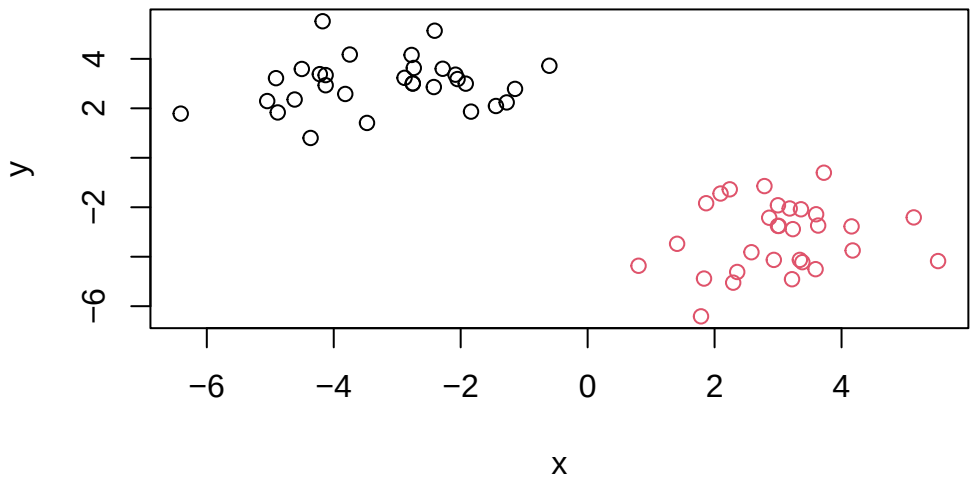
To make the "cut" and get out cluster membership vector we can use the `cutree()` function.

```
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 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
```

Make a plot of our data colored by hclust results

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



Principal Component Analysis (PCA)

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

```
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
```

Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

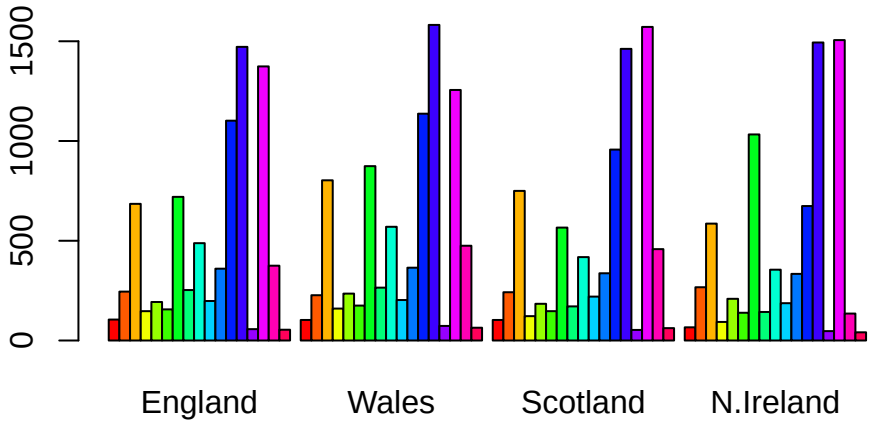
```
ncol(x)
```

```
[1] 4
```

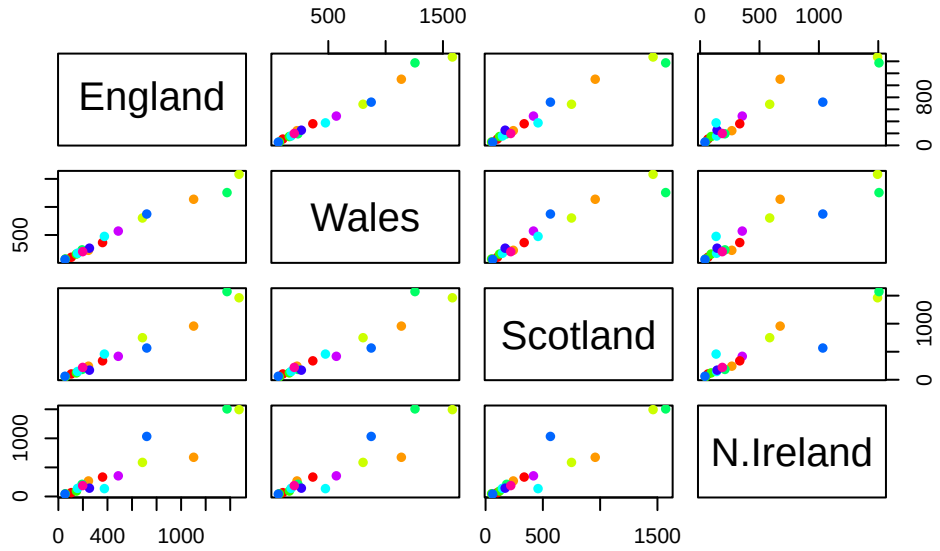
```
nrow(x)
```

```
[1] 17
```

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

```
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 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)
```

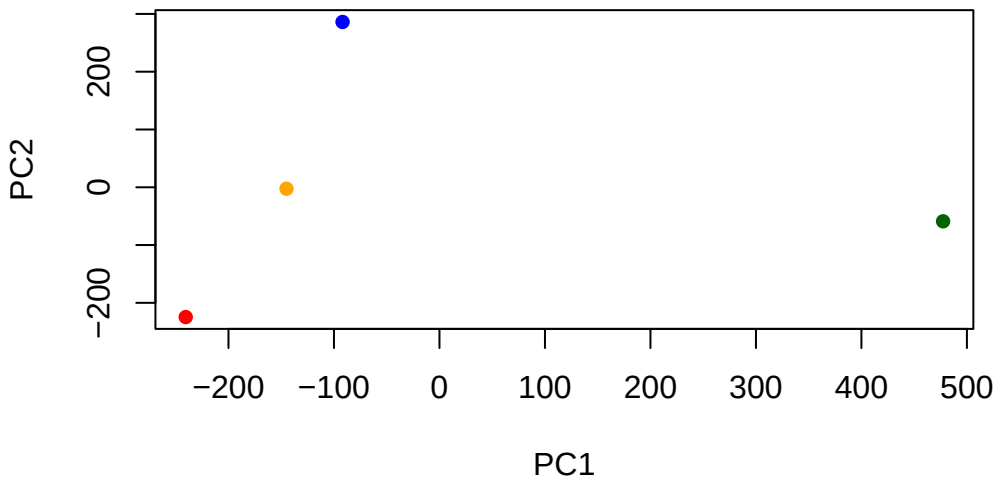
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

We after after the `pca$x` result component to make our 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
```

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



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

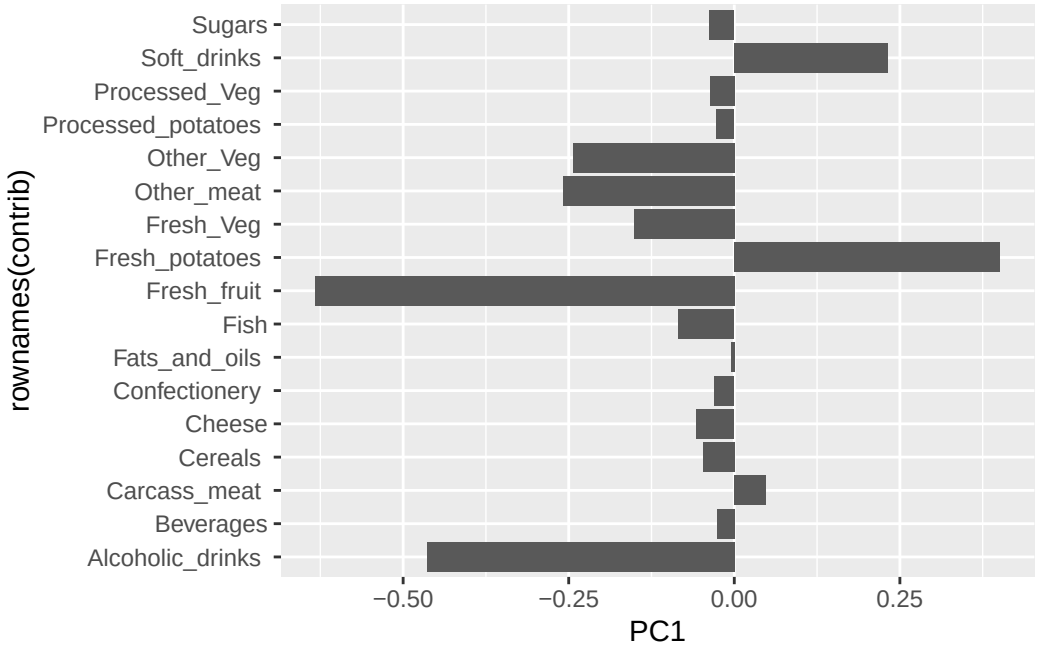
This is contained in the `pca$rotation` object - folks often call this the "loadings" or "contributions" to the PCs.

```
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

```
library(ggplot2)

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

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


rownames(contrib)
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
−0.50
−0.25
0.00
0.25
PC1