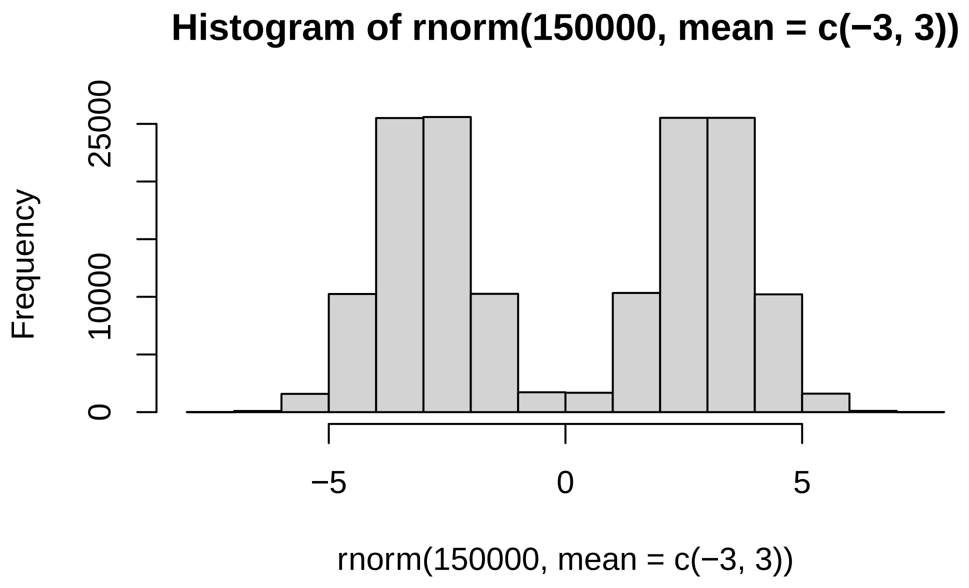
JCW_102324_Bioinfo_Class7MachineLearning1

Janie Chang-Weinberg (A69037446)

Before we get into clustering methods, let's make some sample data to cluster where we know what the answer should be.

To help with this, let's use the `rnorm()` function.

```
hist(rnorm(150000, mean=c(-3,3)))
```



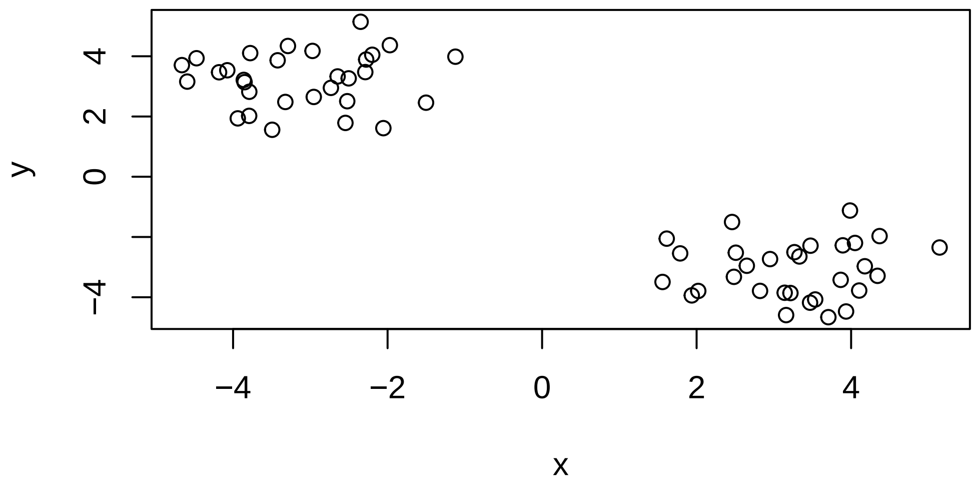
```
x <- c(rnorm(30,mean=+3),rnorm(30,mean=-3))
y <- rev(x)
```

```r
z <- cbind(x,y)
z
```

```
              x         y
 [1,]  2.821216 -3.789988
 [2,]  4.341741 -3.289801
 [3,]  2.458949 -1.502477
 [4,]  3.864960 -3.423299
 [5,]  2.482735 -3.323571
 [6,]  3.705521 -4.662714
 [7,]  3.139817 -3.851245
 [8,]  3.934705 -4.473218
 [9,]  3.158753 -4.593167
[10,]  4.368447 -1.970770
[11,]  3.985490 -1.122058
[12,]  2.649837 -2.955599
[13,]  2.950878 -2.733463
[14,]  2.507178 -2.522330
[15,]  1.558925 -3.493796
[16,]  3.892124 -2.277338
[17,]  1.937394 -3.939072
[18,]  2.021263 -3.791779
[19,]  3.266345 -2.504009
[20,]  4.177228 -2.971988
[21,]  3.214484 -3.861364
[22,]  3.468566 -4.181444
[23,]  4.104453 -3.778150
[24,]  4.050155 -2.198749
[25,]  3.474126 -2.288644
[26,]  1.787172 -2.545523
[27,]  3.535154 -4.074991
[28,]  5.145457 -2.349178
[29,]  3.330402 -2.647137
[30,]  1.613030 -2.054633
[31,] -2.054633  1.613030
[32,] -2.647137  3.330402
[33,] -2.349178  5.145457
[34,] -4.074991  3.535154
[35,] -2.545523  1.787172
[36,] -2.288644  3.474126
[37,] -2.198749  4.050155
[38,] -3.778150  4.104453
```

```
[39,] -4.181444  3.468566
[40,] -3.861364  3.214484
[41,] -2.971988  4.177228
[42,] -2.504009  3.266345
[43,] -3.791779  2.021263
[44,] -3.939072  1.937394
[45,] -2.277338  3.892124
[46,] -3.493796  1.558925
[47,] -2.522330  2.507178
[48,] -2.733463  2.950878
[49,] -2.955599  2.649837
[50,] -1.122058  3.985490
[51,] -1.970770  4.368447
[52,] -4.593167  3.158753
[53,] -4.473218  3.934705
[54,] -3.851245  3.139817
[55,] -4.662714  3.705521
[56,] -3.323571  2.482735
[57,] -3.423299  3.864960
[58,] -1.502477  2.458949
[59,] -3.289801  4.341741
[60,] -3.789988  2.821216
```

```
plot(z)
```



K-means clustering

The function in base R for clustering is `kmeans()`

```
km <- kmeans(z,2)
km
```

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

Cluster means:
          x         y
1  3.231550 -3.105716
2 -3.105716  3.231550

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 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

Within cluster sum of squares by cluster:
[1] 48.98385 48.98385
 (between_SS / total_SS =  92.5 %)
```

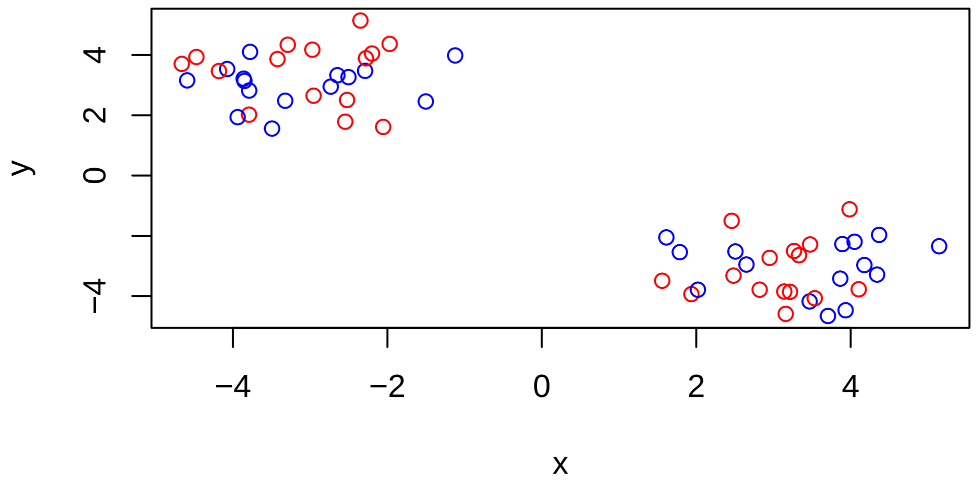
```
Available components:

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

Q. Print out the cluster membership vector (ie, our main answer)

```r
km$cluster
```

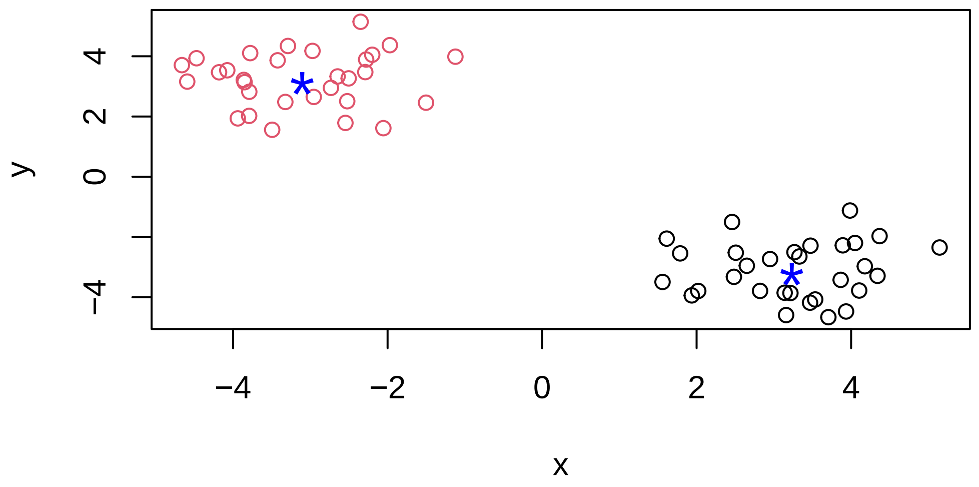
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```r
plot(z, col=c("red","blue"))
```



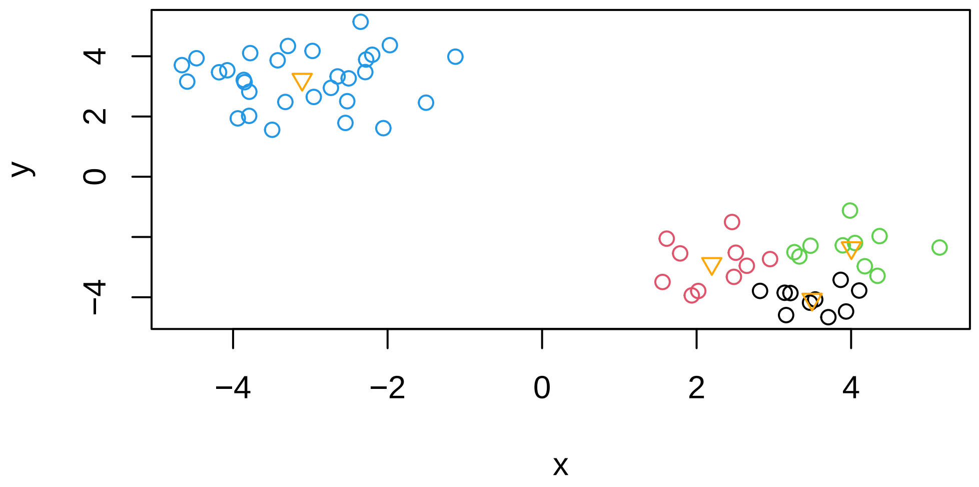
To plot with color by cluster result and add cluster centers:

```r
plot(z, col=km$cluster)
points(km$centers, col="blue", pch=42,cex=2)
```



Q. Can you cluster our data in `z` into four clusters?

```r
km4 <- kmeans(z,centers=4)
plot(z, col=km4$cluster)
points(km4$centers, col="orange",pch=25)
```



Hierarchical Clustering

The main function for hierarchical clustering in base R is called `hclust()`

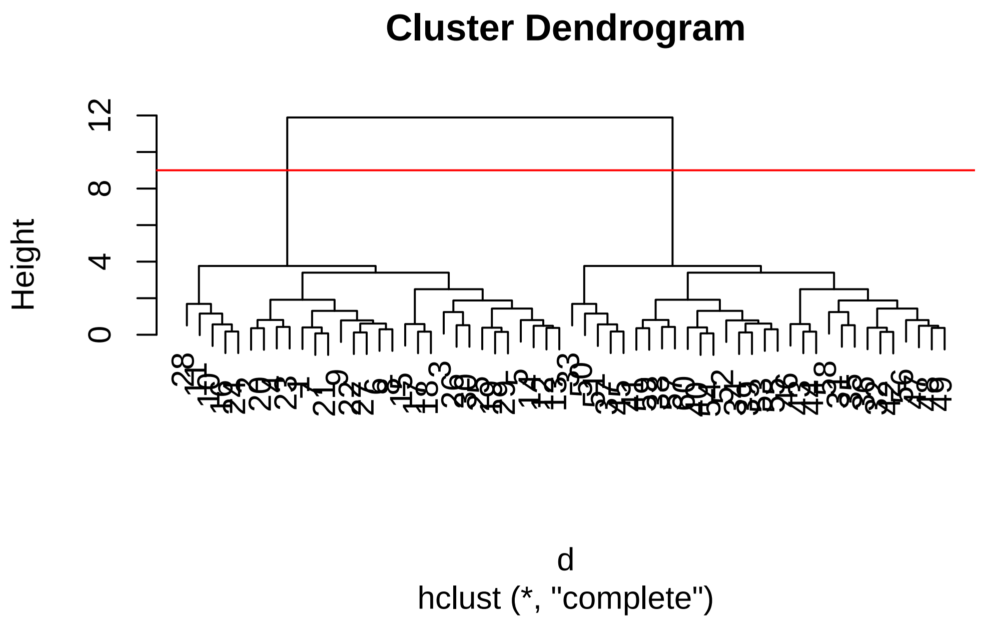
Unlike `kmeans()`, you cannot just pass in the data as input. You first need a distance matrix from the data.

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

```

Call:
hclust(d = d)

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

There is a specific hclust plot() method...

```r
plot(hc)
abline(h=9, col="red")
```



To get the main clustering result (ie, the membership vector), you can “cut” the cluster dendrogram at a given height. To perform this, use `cutree()`

```r
grps <- cutree(hc, h=9)
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
```

Principal Component Analysis

“Principal component analysis (PCA) is a well established”multivariate statistical technique” used to reduce the dimensionality of a complex data set to a more manageable number (typically 2D or 3D). This method is particularly useful for highlighting strong paterns and relationships in large datasets (i.e. revealing major similarities and diferences) that are otherwise hard to visualize. As we will see again and again in this course PCA is often used to make all sorts of bioinformatics data easy to explore and visualize.” -ripped from webpage

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

```
[1] 17  5
```

```r
x
```

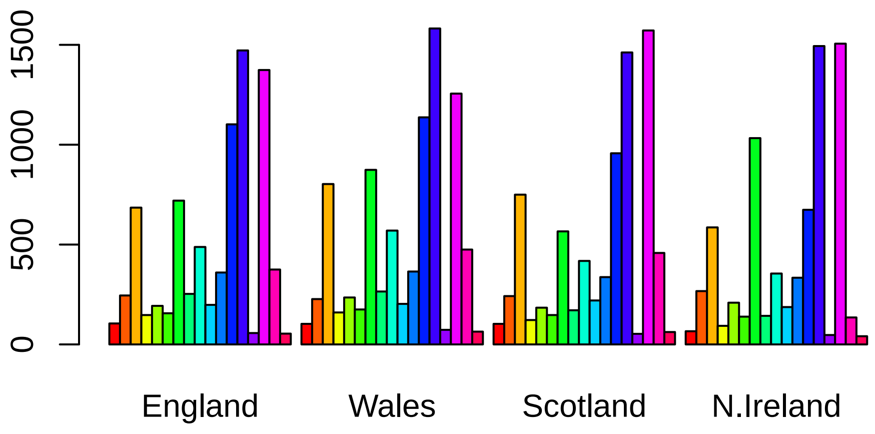
```
                     X England Wales Scotland N.Ireland
1               Cheese     105   103      103        66
2        Carcass_meat      245   227      242       267
3          Other_meat      685   803      750       586
4                 Fish     147   160      122        93
5       Fats_and_oils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

```r
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
```

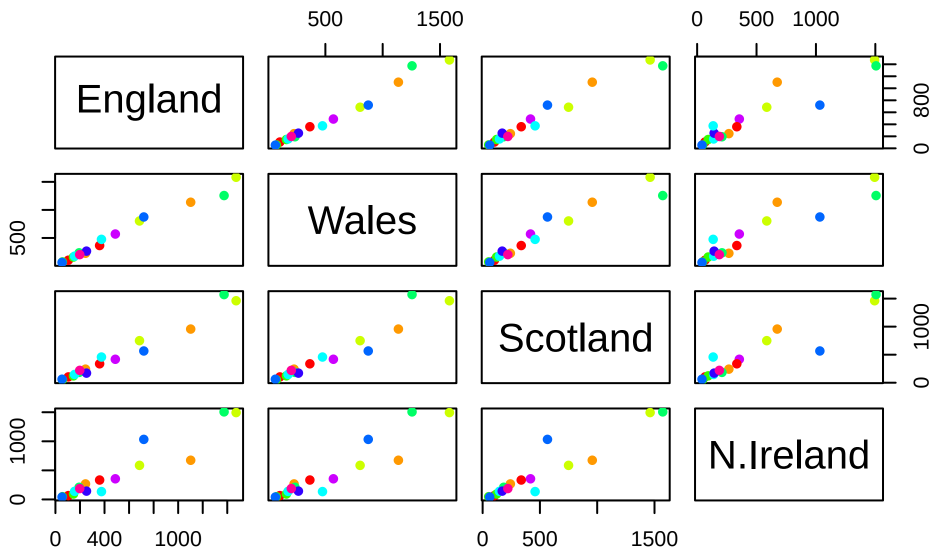
Plot 1 (rainbow bar plot)

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



Q. 3 (pls do this for a barplot)

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



Principal Component Analysis can help organize data

The main function to do PCA in base R is `prcomp()` The way the data are organized currently will not work, you must first transpose the data

```
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
```

Check what is inside `pca` that has been calculated

```
attributes(pca)
```

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

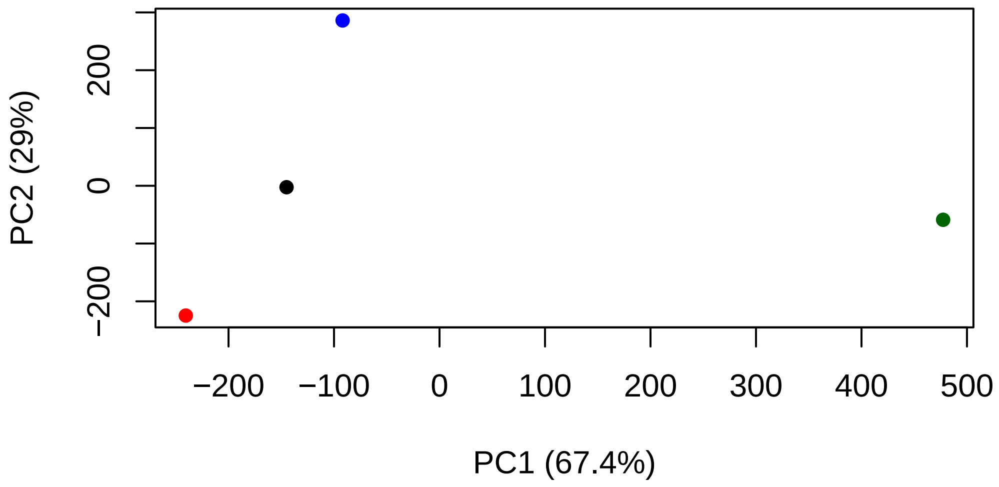
```
$class
[1] "prcomp"
```

```r
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
```

To make our main results figure called a "PC plot" (or score/ordination/PC1vsPC2 plot)

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



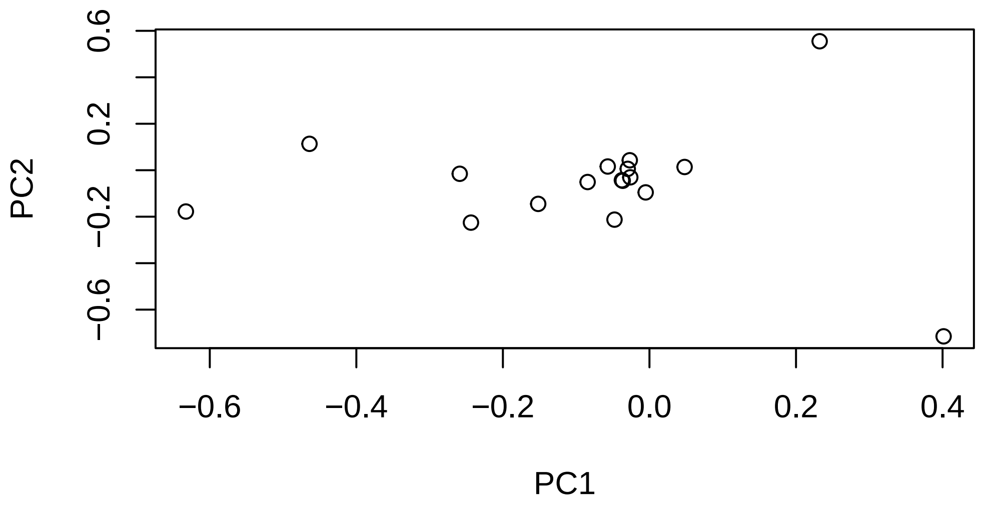
variable loadings plot

can give us insight as to how the original variables (in this case the the foods) contribute to our PC axis

```
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
```

```
plot(pca$rotation)
```


PC2
0.6
0.2
−0.2
−0.6
−0.6
−0.4
−0.2
0.0
0.2
0.4
PC1