Class 7: Machine Learning 1

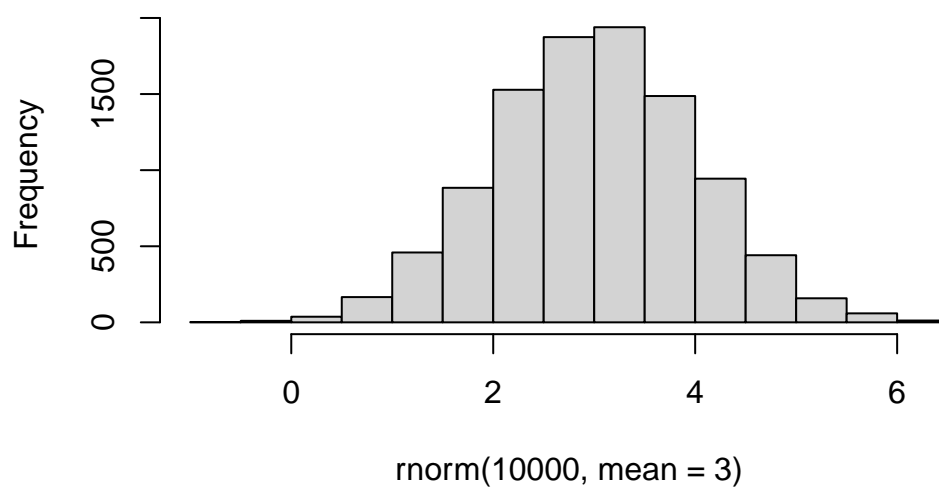Claire Lua A16922295

Table of contents

Today we will explore unsupervised machine learning methods starting with clustering and dimensionality reduction

Clustering

To start let's make up some data to cluster where we know what the answer should be. The `rnorm()` function will help us here.

```
hist(rnorm(10000, mean=3))
```

Histogram of rnorm(10000, mean = 3)

Return 30 numbers centered on -3

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

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

x
```

```
              x          y
 [1,] -3.812657  3.699062
 [2,] -3.665995  4.291565
 [3,] -3.883405  2.619533
 [4,] -2.709428  2.132100
 [5,] -3.014190  2.083283
 [6,] -2.986193  1.572515
 [7,] -5.206799  4.408173
 [8,] -3.577885  1.474522
 [9,] -2.140911  3.334586
[10,] -2.582768  3.136971
[11,] -2.683098  1.742763
[12,] -4.365954  2.903641
[13,] -4.732252  4.166750
```

```
[14,] -2.364280  2.706786
[15,] -4.836768  3.041115
[16,] -4.491823  2.899051
[17,] -3.529523  1.869232
[18,] -3.012815  2.493289
[19,] -2.739448  2.941905
[20,] -2.877336  2.921754
[21,] -2.320675  1.607043
[22,] -1.862930  2.909427
[23,] -3.088172  2.780678
[24,] -3.275759  3.146955
[25,] -2.531653  2.851240
[26,] -2.257890  1.428996
[27,] -1.977663  3.219838
[28,] -3.388429  2.577545
[29,] -2.315033  3.003922
[30,] -2.190221  2.891801
[31,]  2.891801 -2.190221
[32,]  3.003922 -2.315033
[33,]  2.577545 -3.388429
[34,]  3.219838 -1.977663
[35,]  1.428996 -2.257890
[36,]  2.851240 -2.531653
[37,]  3.146955 -3.275759
[38,]  2.780678 -3.088172
[39,]  2.909427 -1.862930
[40,]  1.607043 -2.320675
[41,]  2.921754 -2.877336
[42,]  2.941905 -2.739448
[43,]  2.493289 -3.012815
[44,]  1.869232 -3.529523
[45,]  2.899051 -4.491823
[46,]  3.041115 -4.836768
[47,]  2.706786 -2.364280
[48,]  4.166750 -4.732252
[49,]  2.903641 -4.365954
[50,]  1.742763 -2.683098
[51,]  3.136971 -2.582768
[52,]  3.334586 -2.140911
[53,]  1.474522 -3.577885
[54,]  4.408173 -5.206799
[55,]  1.572515 -2.986193
[56,]  2.083283 -3.014190
```

```
[57,]   2.132100 -2.709428
[58,]   2.619533 -3.883405
[59,]   4.291565 -3.665995
[60,]   3.699062 -3.812657
```

Make a plot of x

```
plot(x)
```
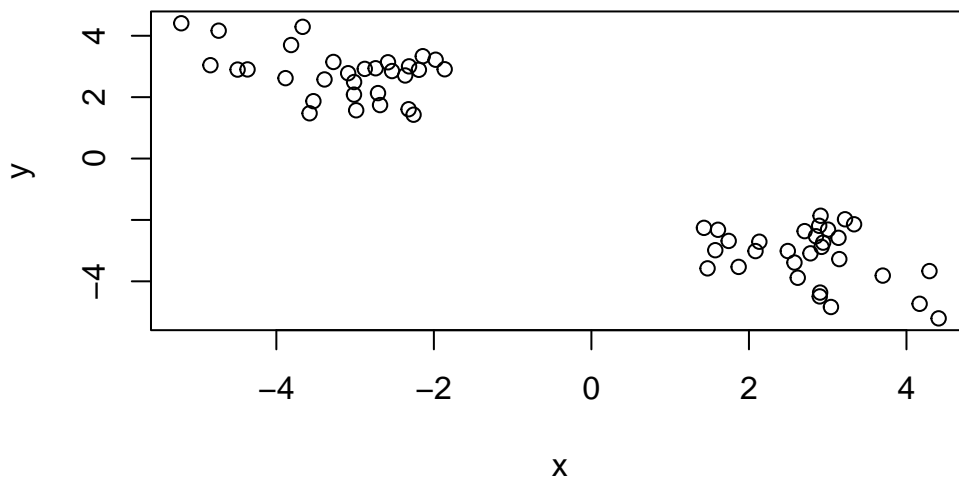

K-means

The main function in "base" R for K-means clustering is called `kmeans()`

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

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

Cluster means:
        x          y
1  2.761868 -3.147398
```

4

```
2 -3.147398  2.761868
```

```
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] 41.70991 41.70991
 (between_SS / total_SS =  92.6 %)
```

```
Available components:

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

The `kmeans()` function return a "list" with 9 components. You can see the named components of any list with the `attributes()` function.

```
attributes(km)
```

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

```
$class
[1] "kmeans"
```

> Q. How many points are in each clister?

```
km$size
```

```
[1] 30 30
```

> Q. Cluster assignment/membership vector?

```
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 1 1
```

> Q. Cluster centers?

```
km$centers
```

```
          x         y
1  2.761868 -3.147398
2 -3.147398  2.761868
```

Q. Make a plot of our `kmeans()` results showing cluster assignment using different colors for each cluster/group of points and cluster centers in blue

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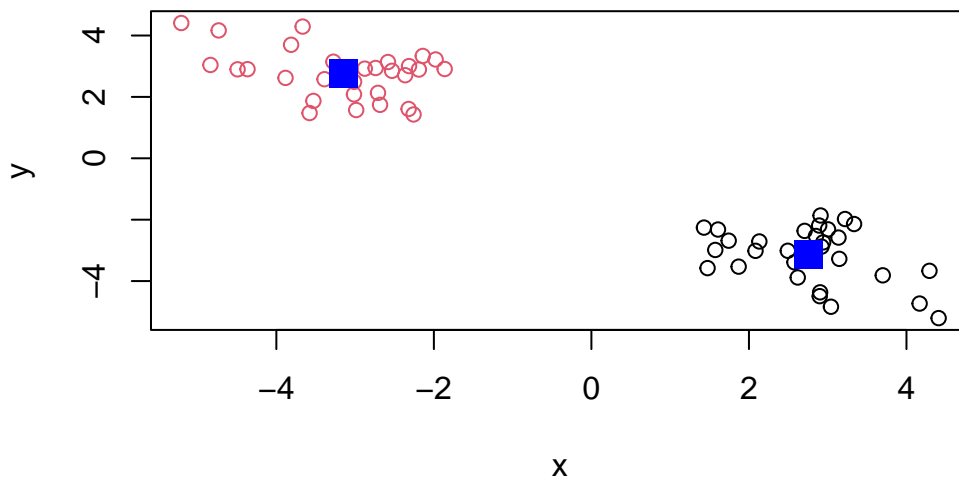

Q. Run `kmeans()` again on x and this cluster into 4 groups/clusters and plot the same result figure as above

```
km4 <- kmeans(x, centers=4)
km4
```

```
K-means clustering with 4 clusters of sizes 8, 22, 16, 14
```

```
Cluster means:
```

```
          x           y
1 -4.374456   3.503611
2 -2.701195   2.492143
3  3.275713  -2.960242
4  2.174616  -3.361291


Clustering vector:
  [1] 1 1 1 2 2 2 1 2 2 2 2 1 1 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 4 3 4 3 3 3
[39] 3 4 3 3 4 4 4 4 3 3 4 4 3 3 4 3 4 4 4 4 3 3


Within cluster sum of squares by cluster:
[1]   5.745069 13.537338 18.785766 12.670510
 (between_SS / total_SS =  95.5 %)


Available components:

[1] "cluster"       "centers"       "totss"         "withinss"      "tot.withinss"
[6] "betweenss"     "size"          "iter"          "ifault"
```
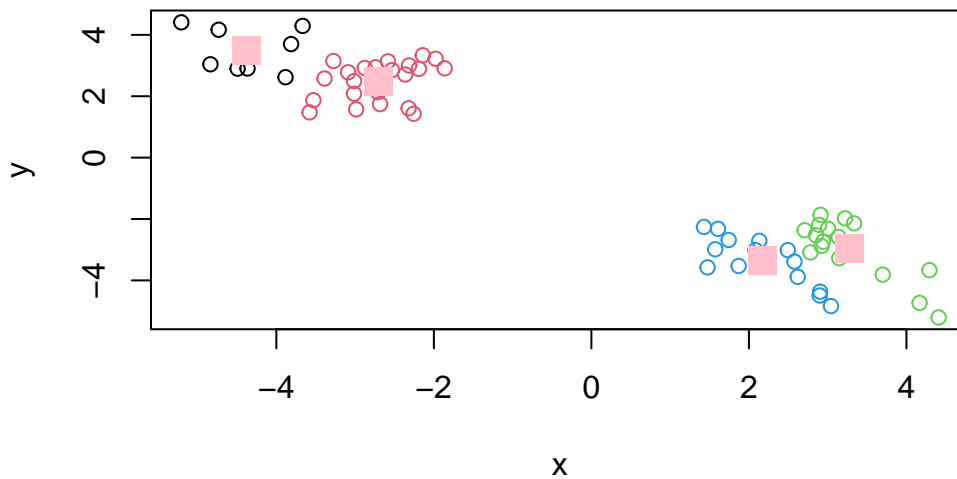
```r
plot(x,col=km4$cluster)
points(km4$centers, col="pink", pch=15, cex=2)
```


key point: K-means clustering is super popular but can be miss-used. One big

limitation is that it can impose a clustering pattern on your data even if clear natural grouping don't exist - i.e. it does what you tell it to do in terms of `centers`

Hierarchical Clustering

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

You can't just pass our dataset as is into `hclust()` you must give "distance matrix" as input. We can get this from the `dist()` function in R.
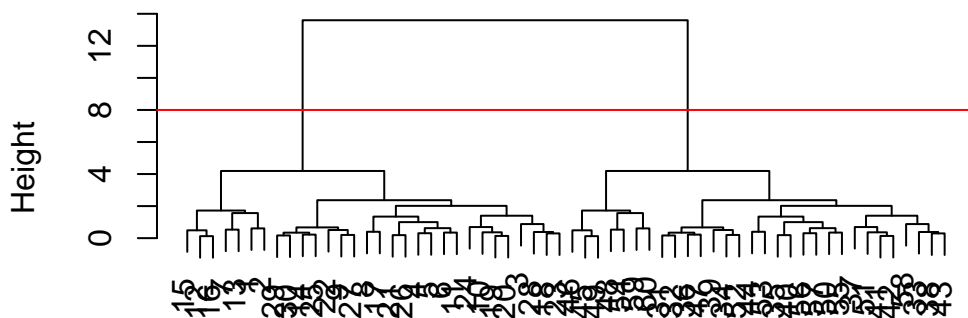
```
d <- dist(x)
hc <- hclust(d)
hc
```

```
Call:
hclust(d = d)

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

The result of `hclust()` don't have a useful `print()` method but do have a special `plot()` method

```
plot(hc)
abline(h=8, col="red")
```

Cluster Dendrogram

d
hclust (*, "complete")

To get our main cluster assignment (membership vector) we need to "cut" the tree at the big goal posts...
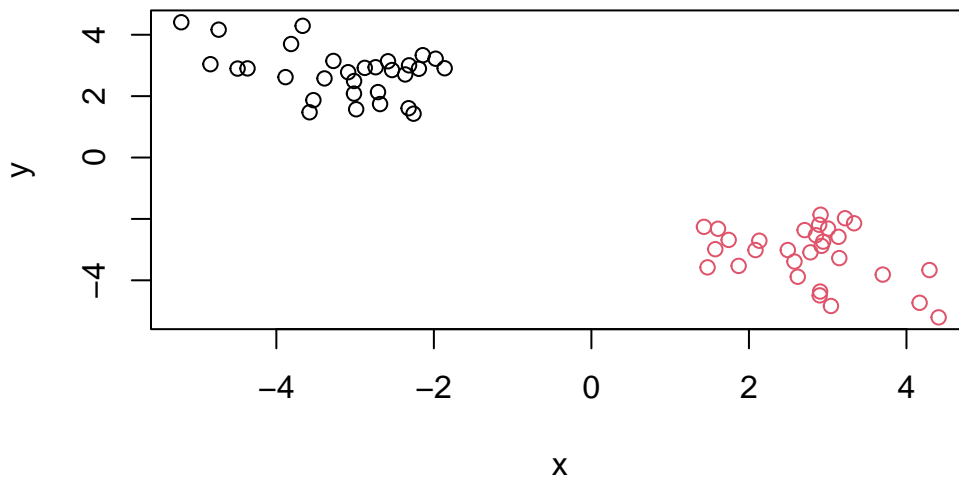
```
grps <- cutree(hc, h=8)
grps
```

```
 [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
```

```
table(grps)
```

```
grps
 1  2
30 30
```

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

9

Hierarchical Clustering is distinct in that the dendrogram (tree figure) can reveal the potential grouping in your data (unlike K-means)

Principal Component Analysis (PCA)

PCA is a common and highly useful dimensionality reduction technique used in many fields - particularly bioinformatics.

Here we will analyze some data from the UK on food consumption.

Data import

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

head(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
```

```r
x <- read.csv(url, row.names=1)
head(x)
```
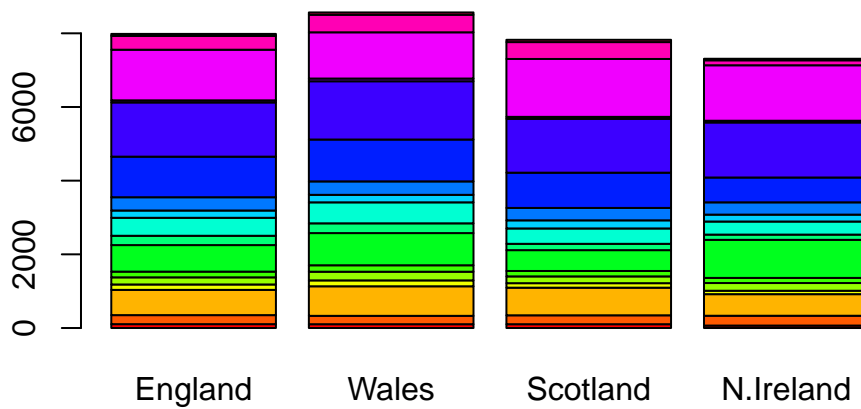
```
              England Wales Scotland N.Ireland
Cheese            105   103      103        66
Carcass_meat      245   227      242       267
Other_meat        685   803      750       586
Fish              147   160      122        93
Fats_and_oils     193   235      184       209
Sugars            156   175      147       139
```

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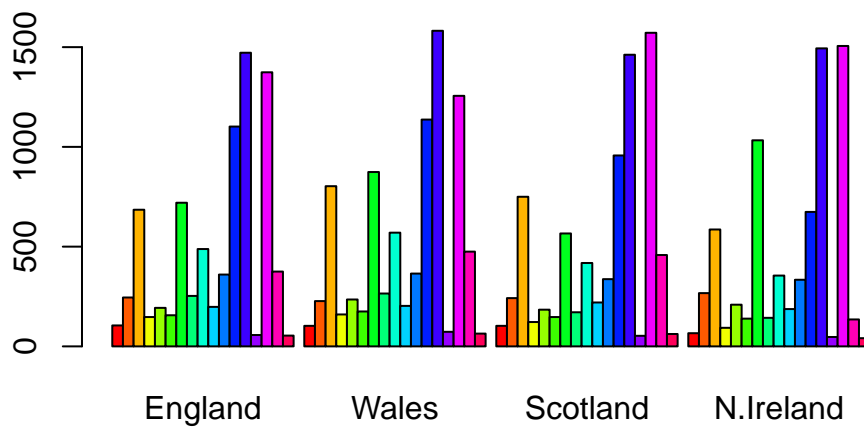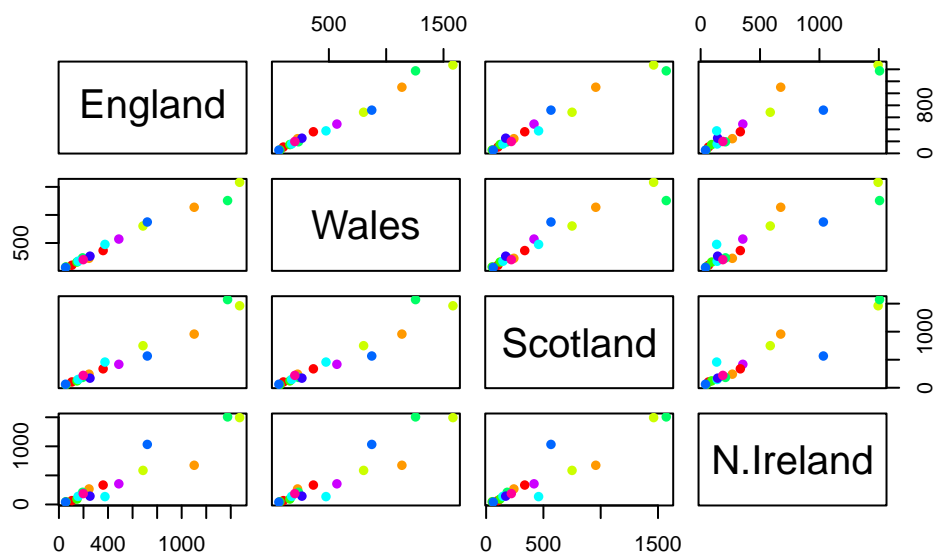


```r
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```

One conventional plot that can be useful is called a "pairs" plot

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


12

PCA to the rescue

The main function in base R 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
```

The `prcomp()` function returns a list object of our result with 5 attributes/components

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

The two main "results" in here are `pca$x` and `pca$rotation`. The first of these (`pca$x`) contains the scores of the data on the new PC axis - we use these to make our "PCA plot".
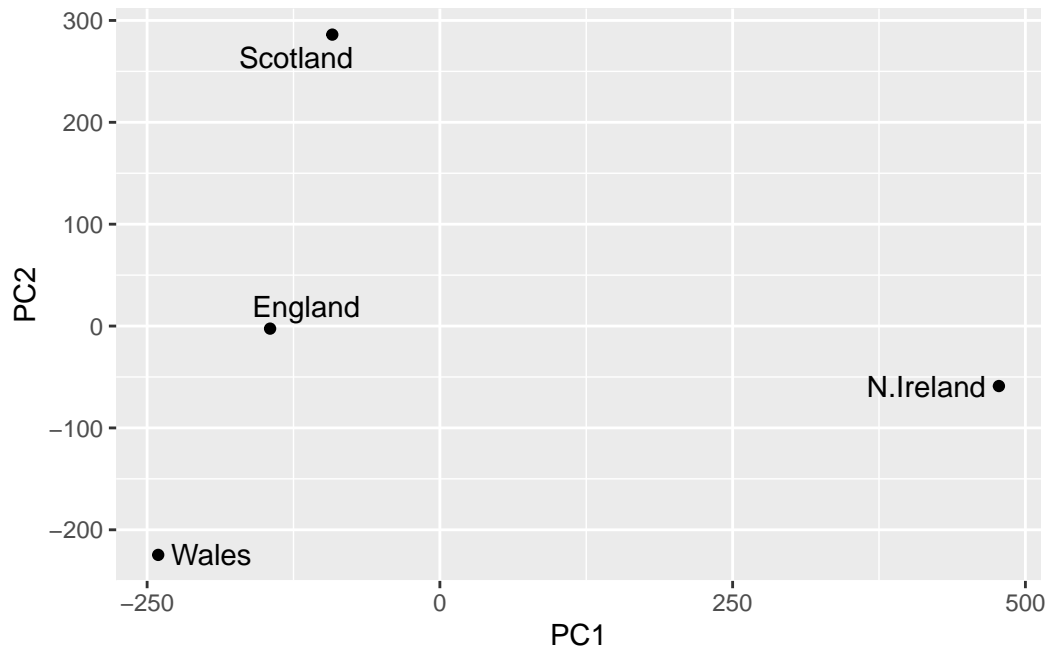
```
pca$x
```

```
                 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
```

```
library(ggplot2)
library(ggrepel)

# Make a plot of pca$x with PC1 vs PC2

ggplot(pca$x, aes(PC1, PC2, label=rownames(pca$x))) +
  geom_point() +
  geom_text_repel()
```
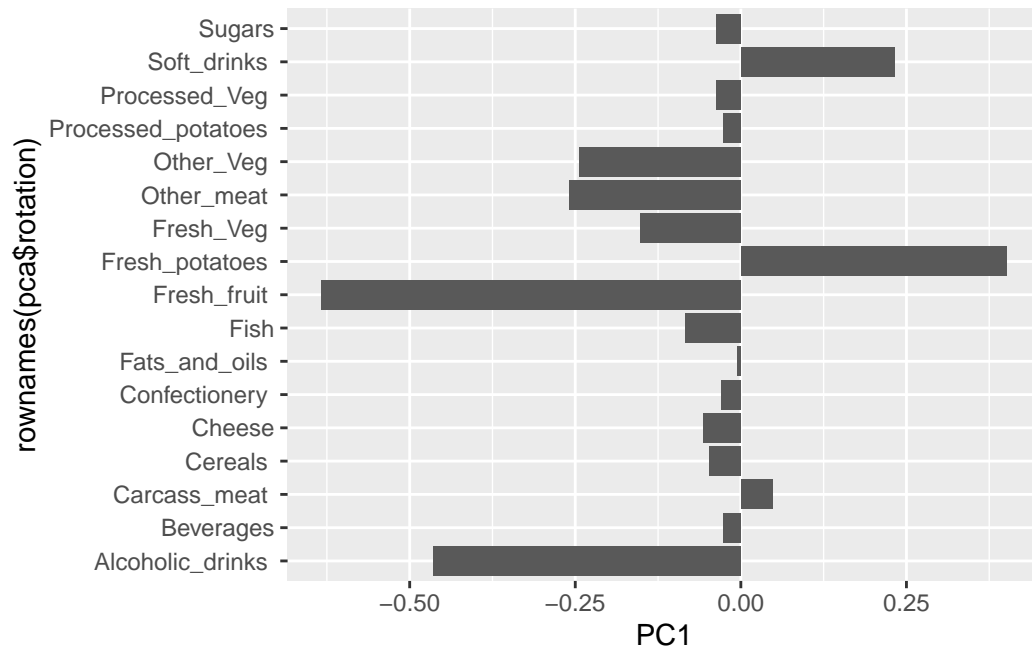
```
# PC1 affect the data the most according to how much they are scattered
# In this graph, we are comparing the four data points in regard to PC1 and PC2
```

The second major result is contained in the `pca$rotation` object or component. Let's plot this to see what PCA is picking up...

```
ggplot(pca$rotation) +
  aes(PC1, rownames(pca$rotation)) +
  geom_col()
```

```
# For this graph, we are just comparing the "food" data point in the PC1 dimension
# The bar plot demonstrates how much each food variable contributes to PC1, with more influen
```