Class07: Machine Learning 1

Kate Zhou

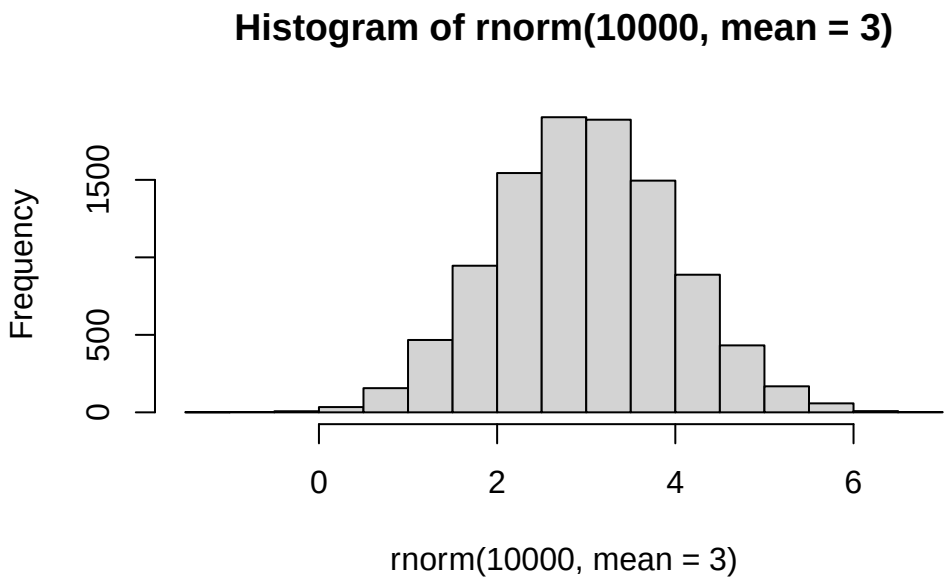
Table of contents

Today we will explore universal unsupervised machine learning method starting with clustering and dimensionality reduction.

Clustering

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

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



Return 30 numbers centered on -3 and

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

x
```

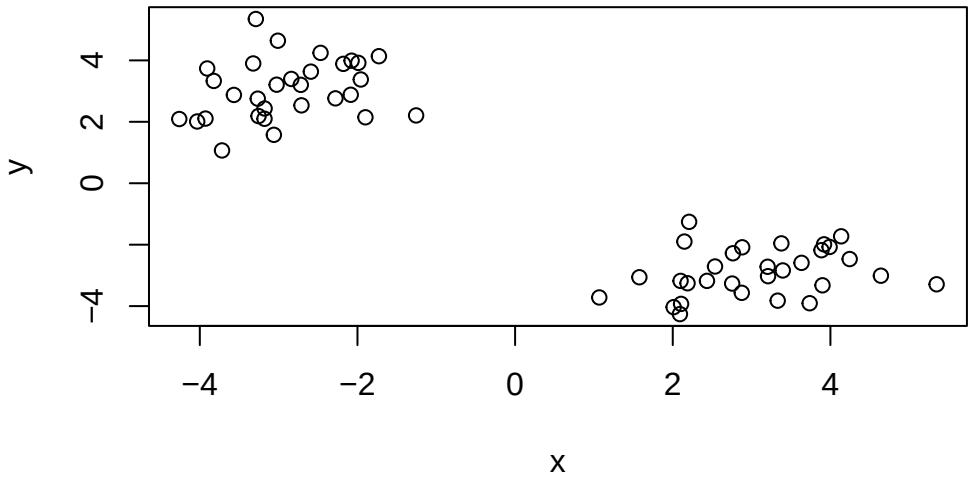
```
              x         y
 [1,] -3.289315  5.347510
 [2,] -2.719044  3.203548
 [3,] -2.074460  3.990018
 [4,] -2.710233  2.536070
 [5,] -3.927933  2.105458
 [6,] -3.567553  2.875545
 [7,] -1.898896  2.147369
 [8,] -3.060529  1.576447
 [9,] -2.591297  3.633757
[10,] -3.255484  2.187432
[11,] -1.728013  4.136525
[12,] -1.988884  3.919063
[13,] -2.279604  2.763274
[14,] -2.179609  3.886697
[15,] -3.718311  1.067879
```

```
[16,] -2.085559  2.880881
[17,] -4.033235  2.011092
[18,] -3.025192  3.208888
[19,] -3.264829  2.753661
[20,] -3.009623  4.640570
[21,] -4.260457  2.090913
[22,] -2.839062  3.395845
[23,] -1.957864  3.377774
[24,] -2.468147  4.244435
[25,] -3.906349  3.736732
[26,] -3.322748  3.899529
[27,] -3.178079  2.433934
[28,] -3.179141  2.099230
[29,] -3.822976  3.331348
[30,] -1.255840  2.207455
[31,]  2.207455 -1.255840
[32,]  3.331348 -3.822976
[33,]  2.099230 -3.179141
[34,]  2.433934 -3.178079
[35,]  3.899529 -3.322748
[36,]  3.736732 -3.906349
[37,]  4.244435 -2.468147
[38,]  3.377774 -1.957864
[39,]  3.395845 -2.839062
[40,]  2.090913 -4.260457
[41,]  4.640570 -3.009623
[42,]  2.753661 -3.264829
[43,]  3.208888 -3.025192
[44,]  2.011092 -4.033235
[45,]  2.880881 -2.085559
[46,]  1.067879 -3.718311
[47,]  3.886697 -2.179609
[48,]  2.763274 -2.279604
[49,]  3.919063 -1.988884
[50,]  4.136525 -1.728013
[51,]  2.187432 -3.255484
[52,]  3.633757 -2.591297
[53,]  1.576447 -3.060529
[54,]  2.147369 -1.898896
[55,]  2.875545 -3.567553
[56,]  2.105458 -3.927933
[57,]  2.536070 -2.710233
[58,]  3.990018 -2.074460
```

```
[59,]  3.203548 -2.719044
[60,]  5.347510 -3.289315
```

Make a plot of `x`

```
plot(x)
```



K-means

The main functio 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 -2.886609  3.056296
2  3.056296 -2.886609
```

```
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 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] 45.04379 45.04379
 (between_SS / total_SS =  92.2 %)

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

```
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 cluster?

```
km$size
```

```
[1] 30 30
```

Q. Cluster assignment/membership vector

```
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 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
```

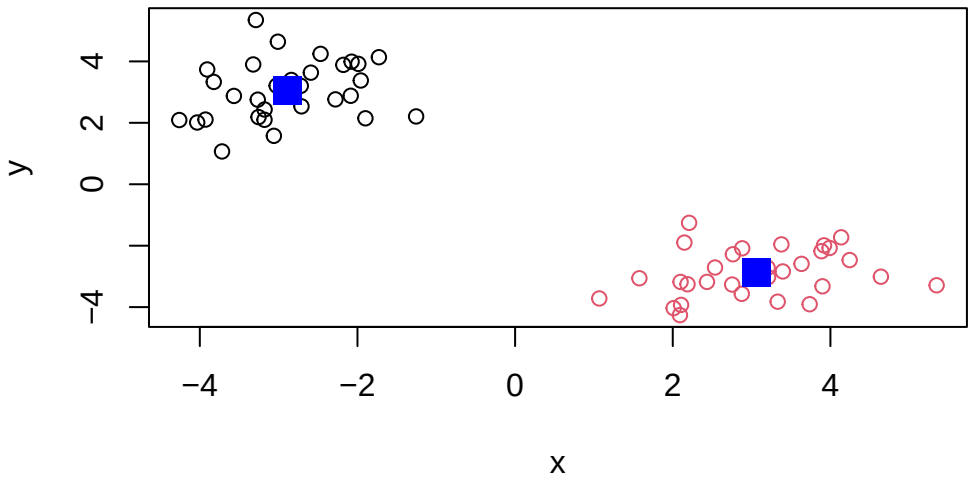
Q. Cluster centers?

```
km$centers
```

```
          x         y
1 -2.886609  3.056296
2  3.056296 -2.886609
```

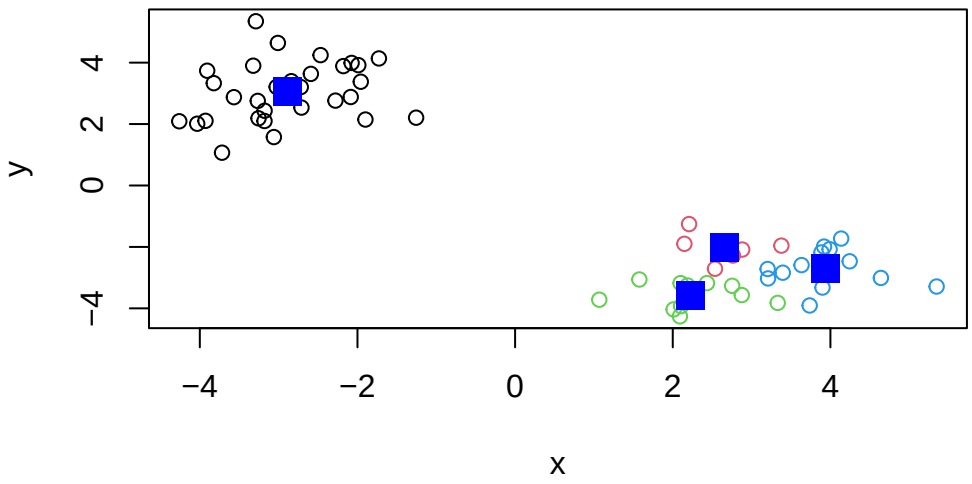
Q. Make a plot of our `kmeans()` results showing cluster assignment using different colors for each cluster/group of points and cluster centers in blue

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



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

```
km_4 <- kmeans(x, centers=4)
plot(x, col=km_4$cluster)
points(km_4$centers, col = "blue", pch=15, cex=2)
```



key-point: k-means clustering is supper 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 do3s what you tell it to in terms of `centers`.

Hierarchical Clustering

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

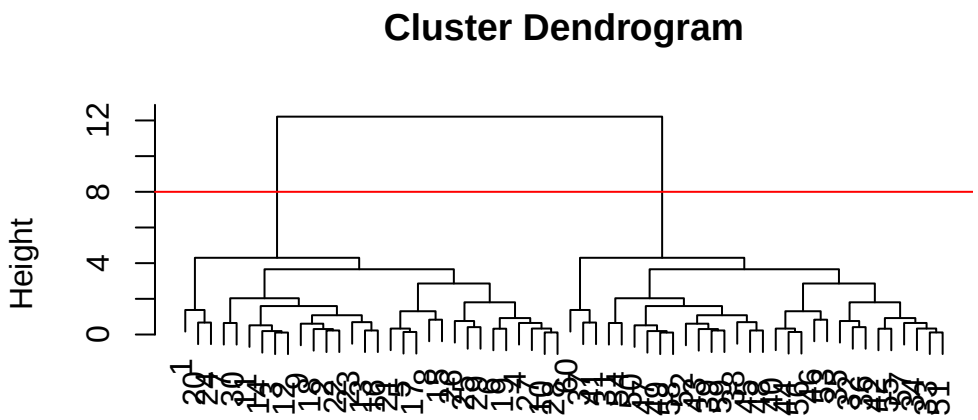
You can't just pass our dataset as is into `hclust()`. You must give "distance matrix" as input. We can get this from the `dist()` function

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

```
Call:
hclust(d = d)

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

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

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



Each point starts as it's own "cluster" and starts to join in closest cluster. To get our main cluster assignment (membership vector), we need to "cut" the tree at the big goal

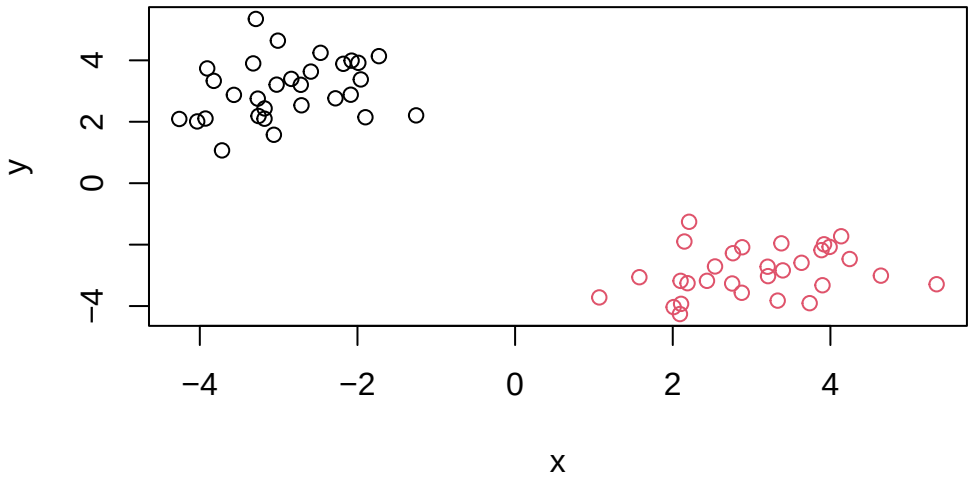
```
grps <- cutree(hc, h=8)
grps
```

```
 [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 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
```

```
table(grps)
```

```
grps
 1  2
30 30
```

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



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

Dimensionality reduction, visualization and 'structure' analysis

Principal Component Analysis (PCA)

PCA projects the features onto the principal components. Principal components are new low dimensional axes.

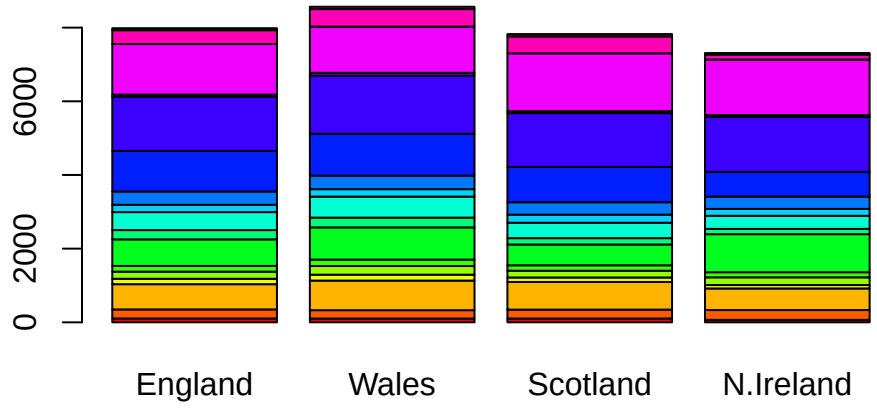
PCA is a common and highly useful dimension reduction technique used in many fields - particularly bioinformatics.

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

```
url <- "https://bioboot.github.io/bggn213_f17/class-material/UK_foods.csv"
x <- read.csv(url, row.names = 1)
head(x)
```

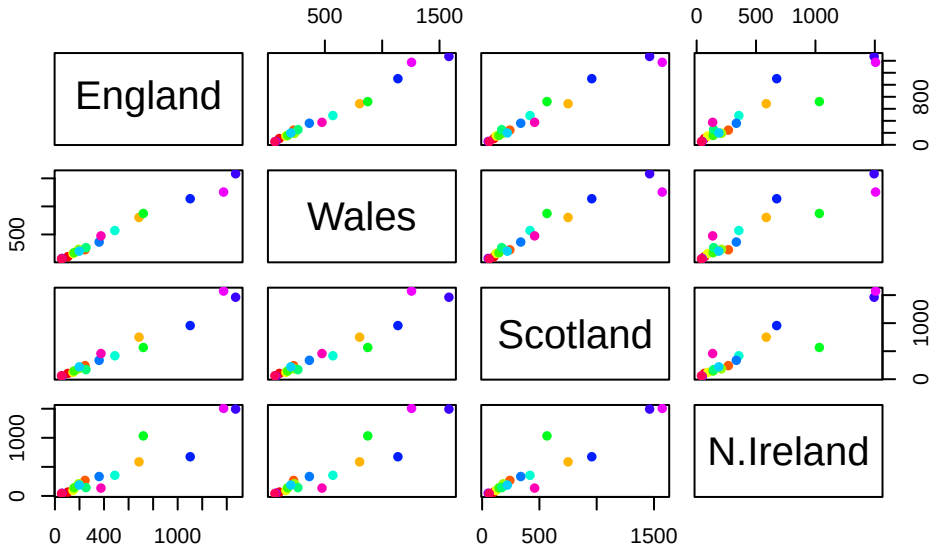
```
               England Wales Scotland N.Ireland
Cheese             105   103      103        66
Carcass_meat       245   227      242       267
Other_meat         685   803      750       586
Fish               147   160      122        93
Fats_and_oils      193   235      184       209
Sugars             156   175      147       139
```

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



One conventional plot that can be useful

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



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 results with

```
attributes(pca)
```

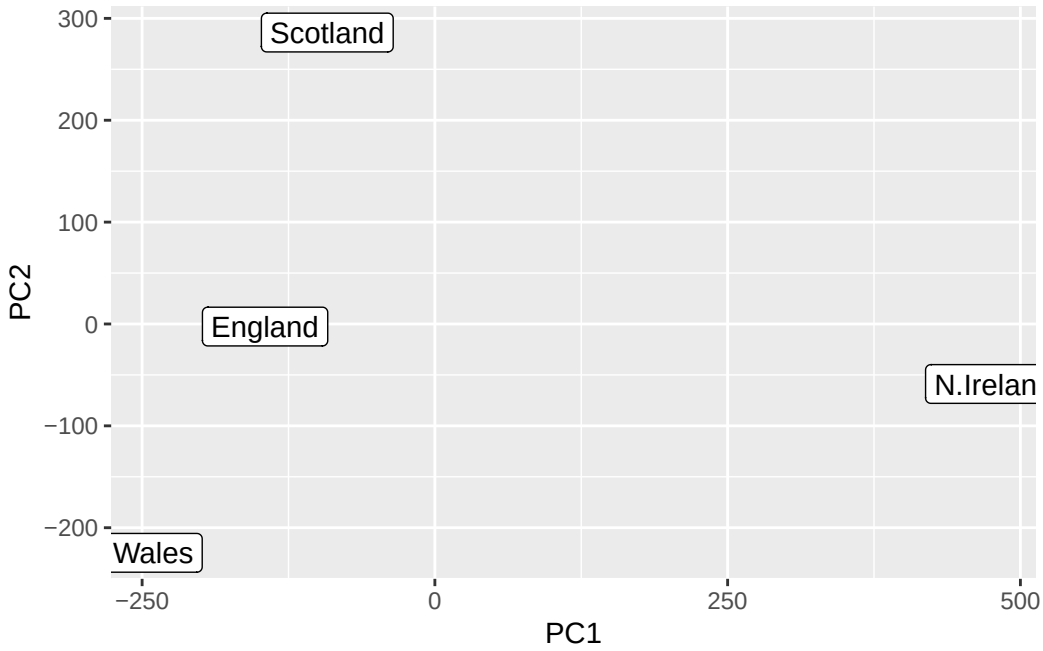
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$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
```

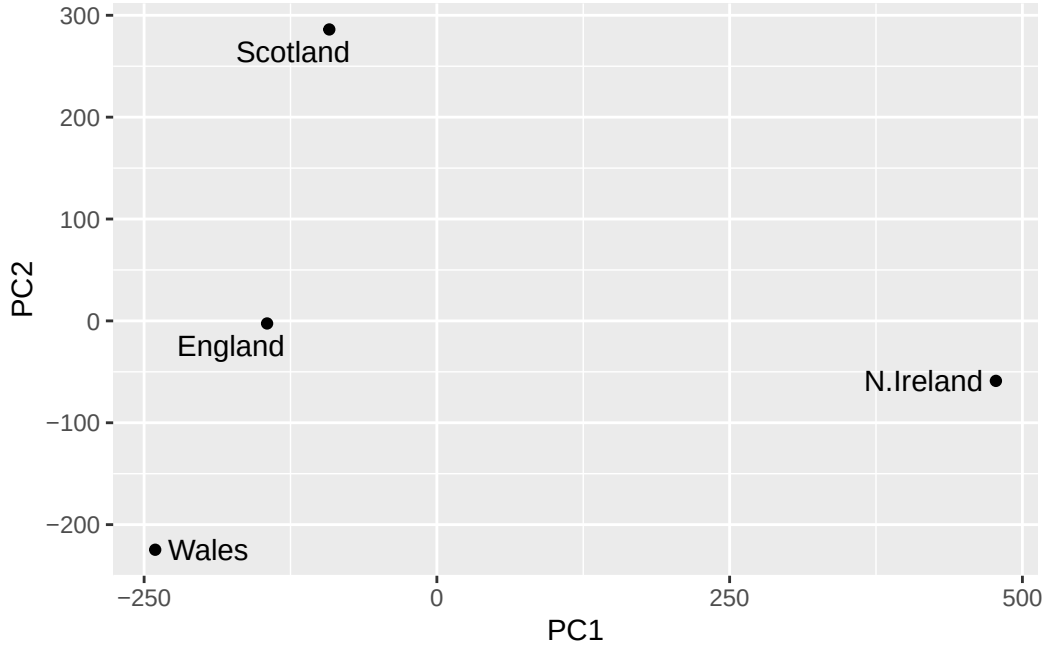
```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -9.152022e-15
Wales     -240.52915 -224.646925 -56.475555  5.560040e-13
Scotland   -91.86934  286.081786 -44.415495 -6.638419e-13
N.Ireland  477.39164  -58.901862  -4.877895  1.329771e-13
```

```
library(ggplot2)
library(ggrepel)
# Make a plot of pca$x with PC1 and PC2
ggplot(pca$x) +
  aes(PC1, PC2, label=rownames(pca$x)) +
  geom_point() +
  geom_label()
```




```
ggplot(pca$x) +
  aes(PC1, PC2, label=rownames(pca$x)) +
```

```
  geom_point() +
  geom_text_repel()
```



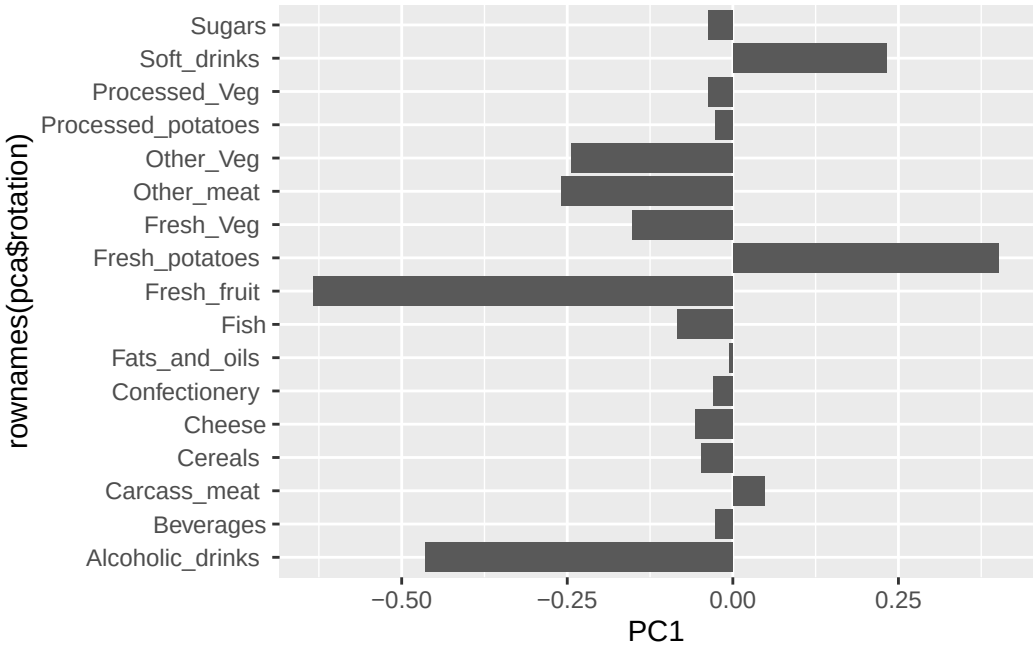
The second major result is contained in `pca$rotation` object or component. Let's plot this to see what PCA is picking up…

```
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
```

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



From the first pca$x plot, we see that North Ireland is separated from other 3 on the PC1 axis. This plot shows how their consumption are different in categories on PC1 axis.