Machine Learning

Tusha Karnani

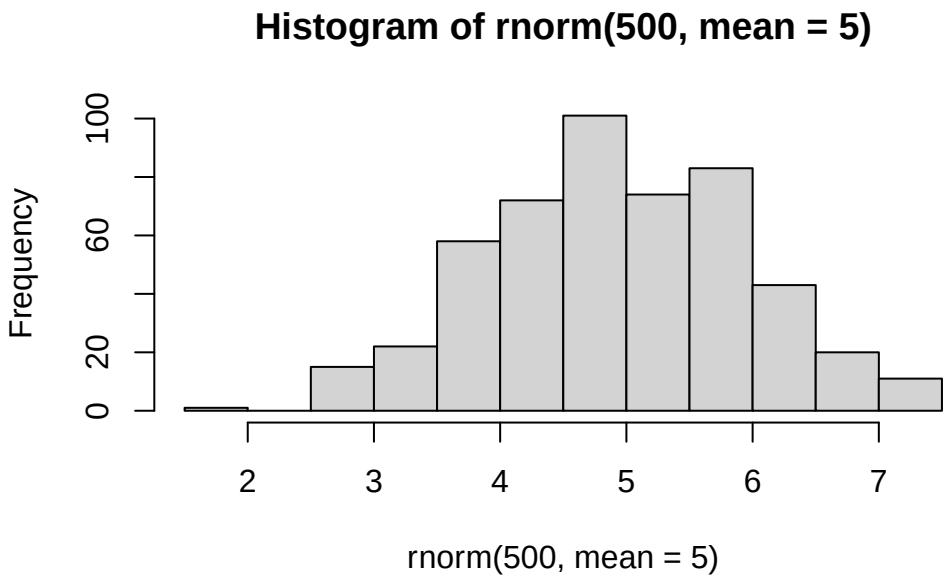
Table of contents

Fundamentals of machine learning including clustering and dimensionality reduction

K-means clustering

Creating a data set:

```
hist(rnorm(500,mean=5))
```



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

data <- cbind(x,y)
```

```r
k <- kmeans(data, 2)
k
```

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

Cluster means:
          x         y
1 -3.011475  2.962974
2  2.962974 -3.011475

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 2 2 2 2 2

Within cluster sum of squares by cluster:
[1] 48.30937 48.30937
 (between_SS / total_SS =  91.7 %)

Available components:

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

How many points are in each cluster?

```r
k$size
```

```
[1] 30 30
```

What components of the result object details: - cluster assignment/membership? - cluster center?

```r
k$cluster
```

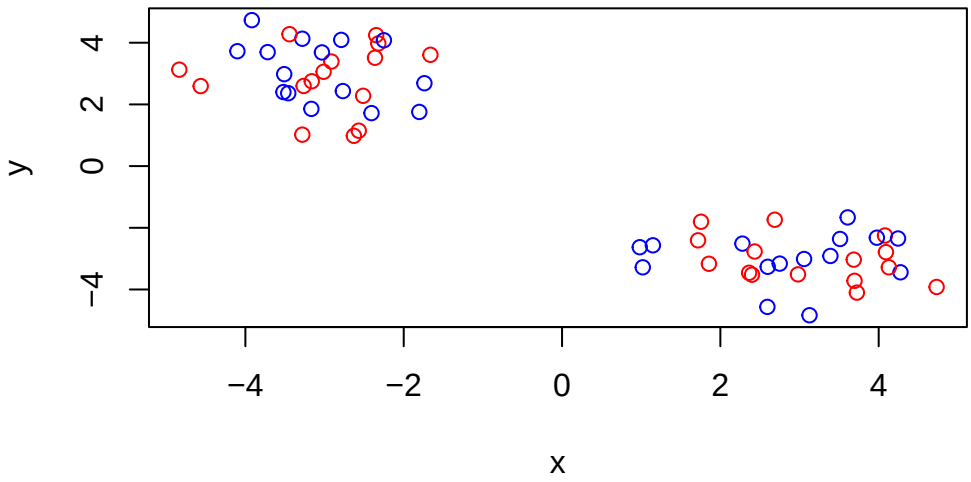
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
k$centers
```

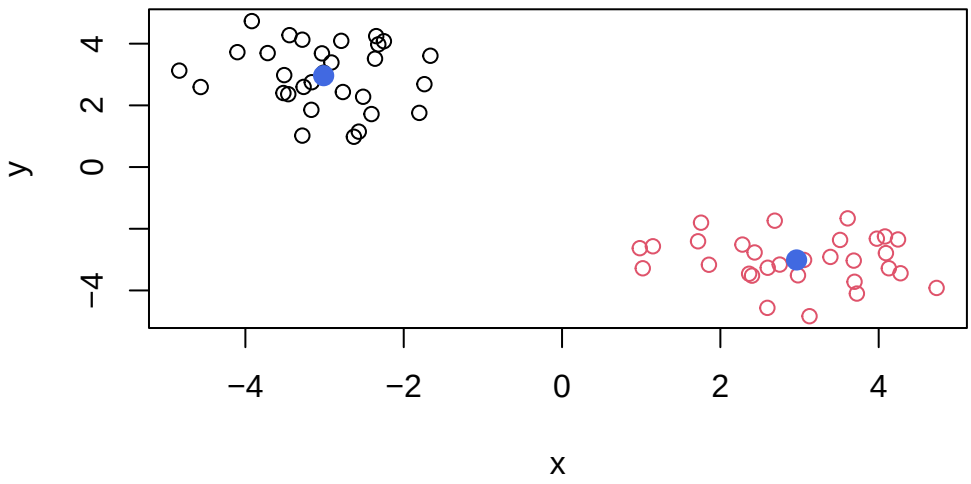
```
          x         y
1 -3.011475  2.962974
2  2.962974 -3.011475
```

Clustering results figure colored by cluster membership looks like the following:

```
plot(data, col=c("red", "blue"))
```




```
plot(data, col=k$cluster)
points(k$centers, col="royalblue", pch=20, cex=2)
```



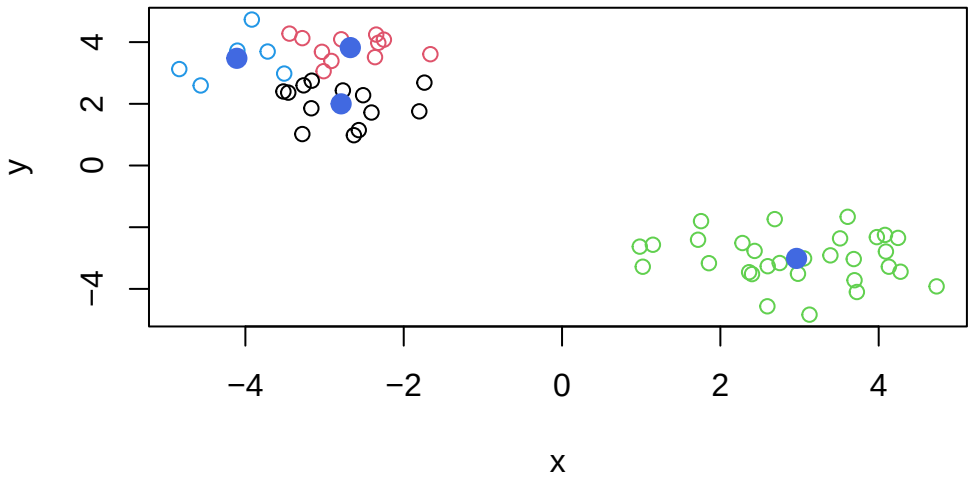
```
# pch is plotting character i.e. dot type
# cex is character expansion
```

K-means clustering is very popular as it is very fast and relatively straightforward: it takes numeric data as input and return the cluster membership, etc.

The problem is that we must tell k-means how many clusters we want.

Clustering it into 4 groups and plotting the results like above looks like the following:

```
k4 <- kmeans(data, 4)
plot(data, col=k4$cluster)
points(k4$centers, col="royalblue", pch=20, cex=2)
```



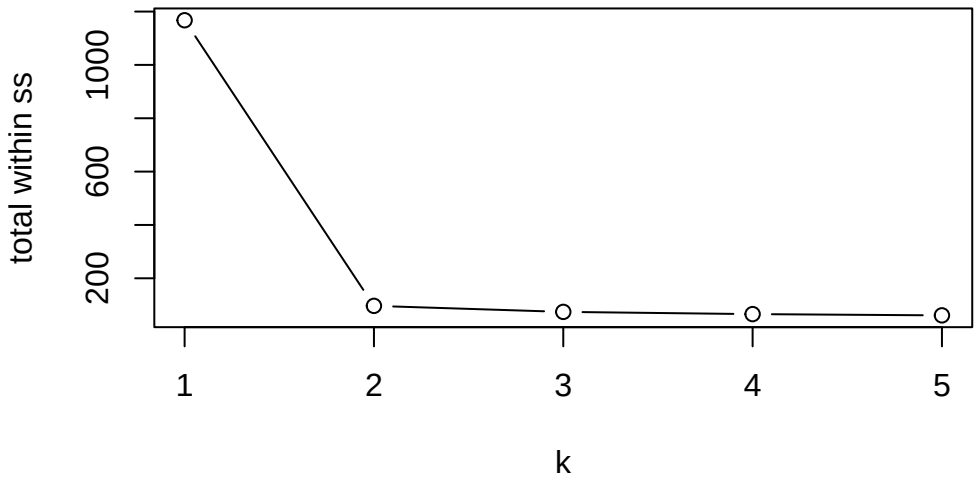
It picks lowest value of total variation within clusters. That is the tightest fit and returns that. There is a large reduction in SS at k=2 but after that the values do not go down as quickly.

```
k1 <- kmeans(data, 1)
k3 <- kmeans(data, 3)
k5 <- kmeans(data, 5)

ss <- c(k1$tot.withinss, k$tot.withinss, k3$tot.withinss, k4$tot.withinss, k5$tot.withinss)
ss
```

```
[1] 1167.43976   96.61873   74.10168   65.73590   61.03211
```

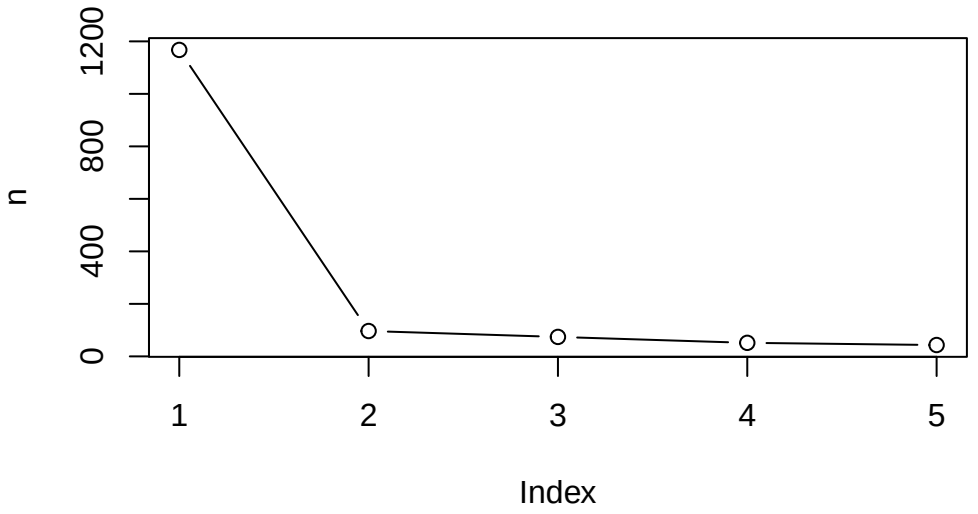
```r
plot(ss, type="b", xlab="k", ylab="total within ss")
```




```r
n <- NULL

for (i in 1:5)
{
  n <- c(n, kmeans(data, i)$tot.withinss)
}

plot(n, type="b")
```



Hierarchichal Clustering

We can't just input our data for this directly. We need to first calculate a distance matrix for our data (e.g. `dist()`) and use it as input for the `hclust()` function. Note: We can do this using not just euclidean distance but also sequence similarity, amino acid similarity, etc.

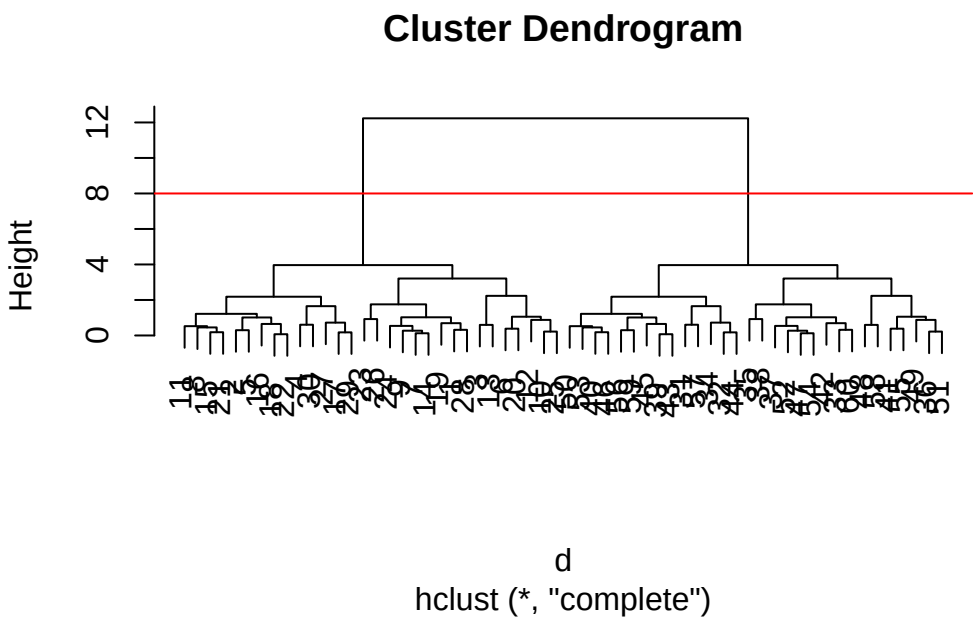
```
d <- dist(data)
# 60x60 matrix with the distance of every point in the dataset with every other point

hc <- hclust(d)
hc
```

```
Call:
hclust(d = d)

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

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



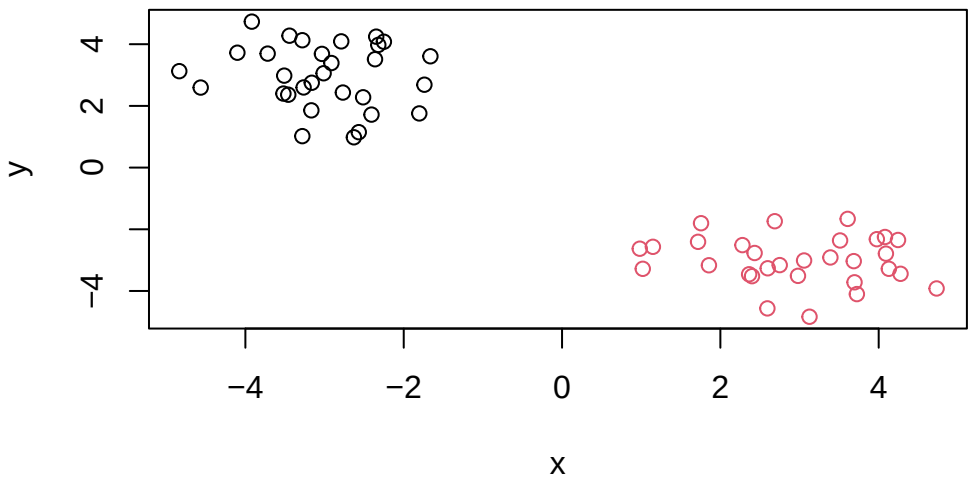
The numbers between 1-30 and 31-60 are on separate sides of the graphs i.e. they are highly separated by distance.

To get our cluster membership vector (i.e. our main clustering result) we can "cut" the tree at a given height or a height that yields a given "k" number of groups.

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

Plot the data with our hclust result coloring

```
plot(data, col=hccol)
```



Principal Component Analysis

PCA of UK food data

For this, I imported a data set containing comsumption patterns of 6 food groups across 4 regions.

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

```
[1] 17  4
```

```
head(food)
```

```
               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
```

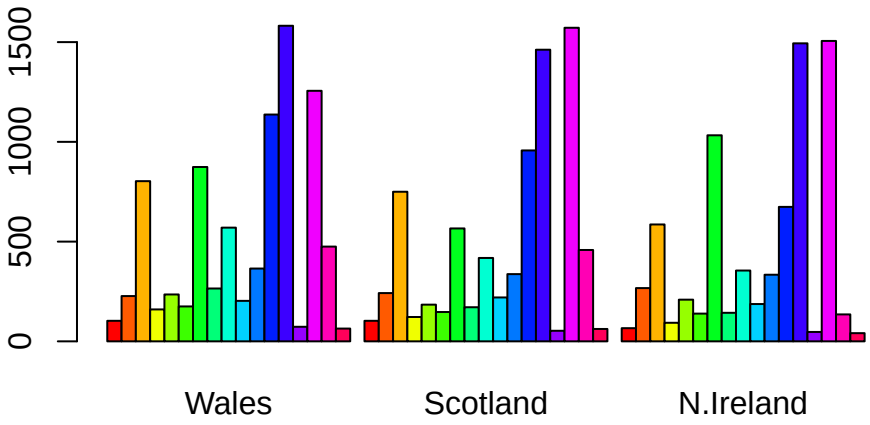
```
rownames(food) <- food [,1]
food <- food[,-1]
food
```

```
     Wales Scotland N.Ireland
105    103      103        66
245    227      242       267
685    803      750       586
147    160      122        93
193    235      184       209
156    175      147       139
720    874      566      1033
253    265      171       143
488    570      418       355
198    203      220       187
360    365      337       334
1102  1137      957       674
1472  1582     1462      1494
57      73       53        47
1374  1256     1572      1506
375    475      458       135
54      64       62        41
```

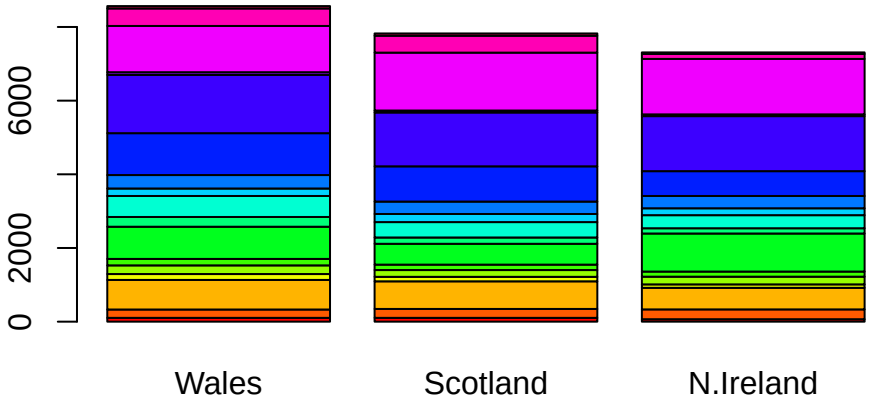
```
# Can get disruptive if run again (keeps removing columns)
```

Some base figures

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

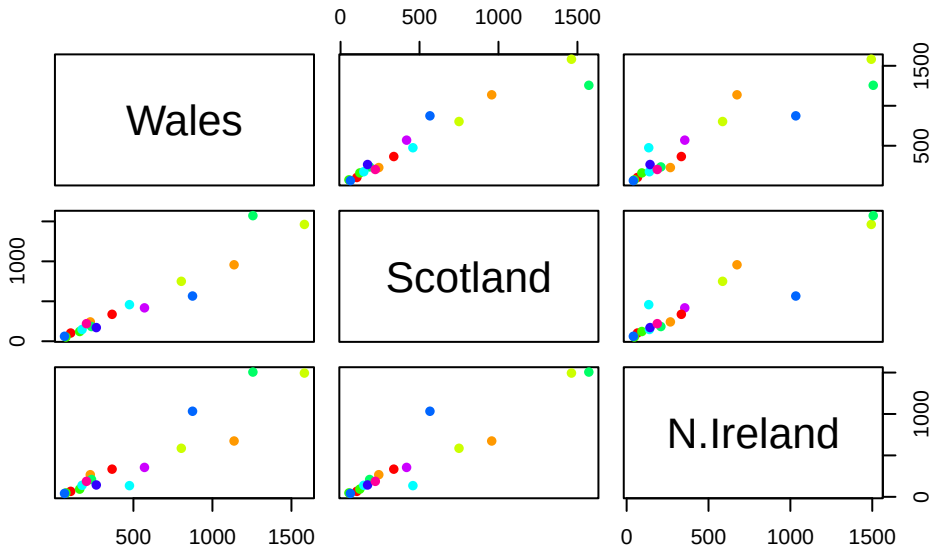



```
barplot(as.matrix(food), beside=F, col=rainbow(nrow(food)))
```



There is one plot that can be useful for small datasets

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



Note: It can be challenging to spot major trends and patters even in relatively small multivariate datasets (here we have 17, typically we have 1000s)

PCA

In base R, the main PCA function is `prcomp()`

I will take the transpose of our food data so the "foods" are in the columns

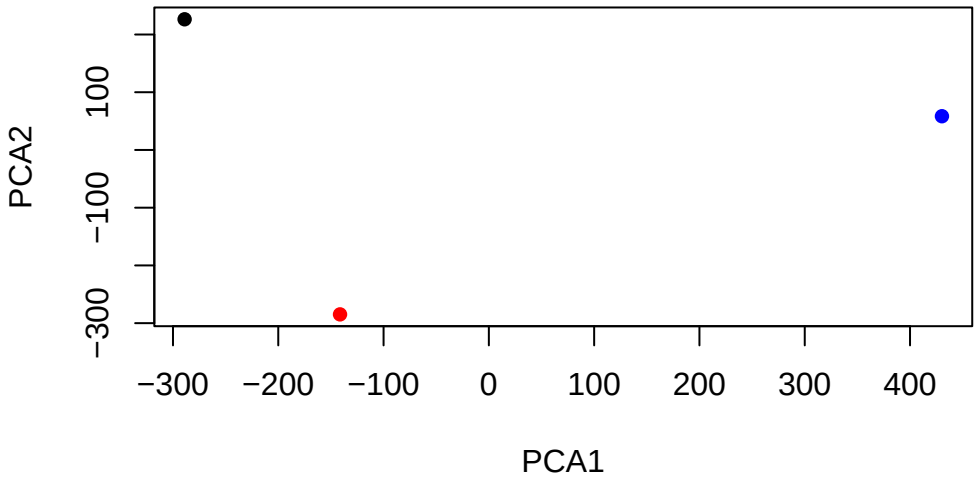
```
pca <- prcomp(t(food))
summary(pca)
```

```
Importance of components:
                            PC1      PC2       PC3
Standard deviation     379.8991 260.5533 1.438e-13
Proportion of Variance   0.6801   0.3199 0.000e+00
Cumulative Proportion    0.6801   1.0000 1.000e+00
```

```
pca$x
```

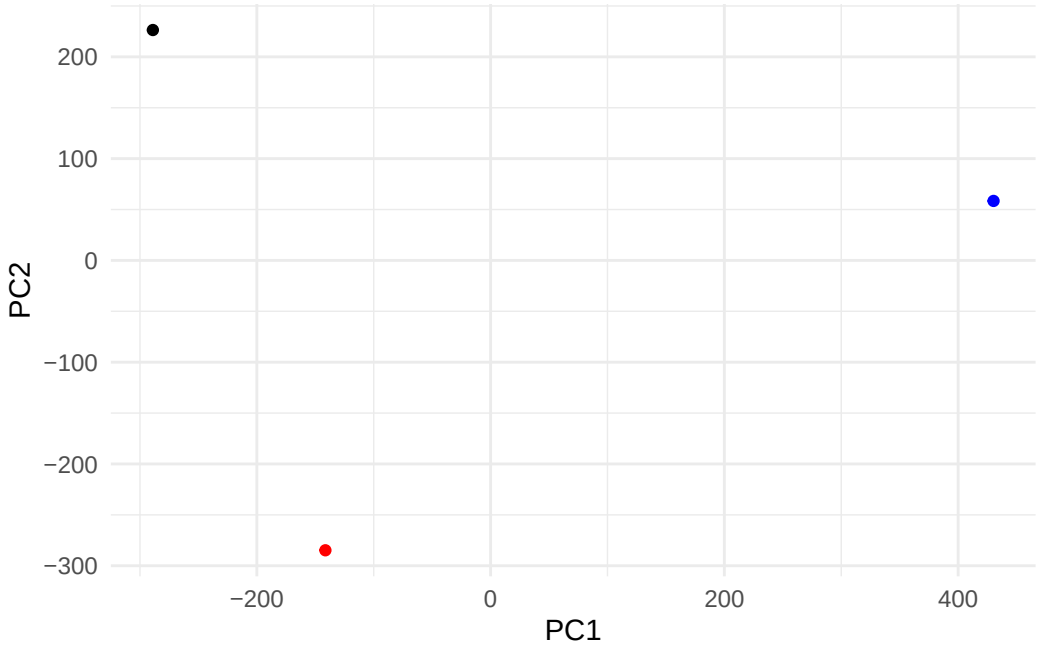
```
                PC1        PC2           PC3
Wales     -288.9534  226.36855  1.776357e-15
Scotland  -141.3603 -284.81172  4.298784e-13
N.Ireland  430.3137   58.44317 -9.592327e-14
```

```r
cols <- c("black","red","blue")
plot(pca$x[,1], pca$x[,2], col=cols, pch=16, xlab="PCA1", ylab="PCA2")
```




```r
library(ggplot2)
```

```r
ggplot(pca$x) +
  aes(PC1, PC2) +
  geom_point(col=cols) +
  theme_minimal()
```

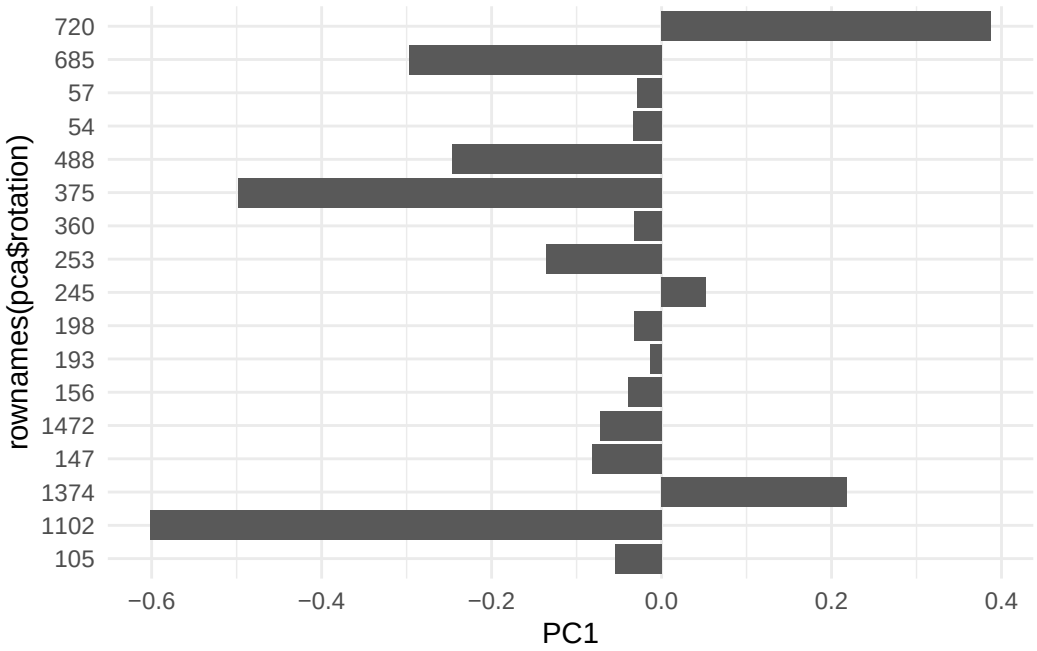


```
pca$rotation
```

```
             PC1          PC2          PC3
105  -0.05515951 -0.015926208  0.089159718
245   0.05228588 -0.014247351  0.160169691
685  -0.29754711  0.017770827 -0.143714947
147  -0.08127351  0.050871661 -0.035378388
193  -0.01378426  0.095789177 -0.088149169
156  -0.03995614  0.043238659  0.062522585
720   0.38787509  0.714518360 -0.188185663
253  -0.13584223  0.144666425 -0.044940419
488  -0.24608178  0.226299860 -0.277334193
198  -0.03217825 -0.042547197 -0.092922896
360  -0.03250215  0.045390849  0.144304277
1102 -0.60208698  0.178285653  0.102836115
1472 -0.07242201  0.213840430 -0.659763755
57   -0.02896604  0.030761774  0.009935304
1374  0.21794320 -0.555250465 -0.557926740
375  -0.49854565 -0.110688746 -0.178079520
54   -0.03330887 -0.005704759  0.006954684
```

```
# How much each of these variables weighs on each of the axes/pcs
# Telling us what makes two countries different from each other (based on PCA results)
```

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


rownames(pca$rotation)
720
685
57
54
488
375
360
253
245
198
193
156
1472
147
1374
1102
105
−0.6
−0.4
−0.2
0.0
0.2
0.4
PC1