Class_07_Clustering_013124

Johann Tailor

Clustering

Today we will learn about machine learning and how to use the function `kmeans()`

Kmeans clustering

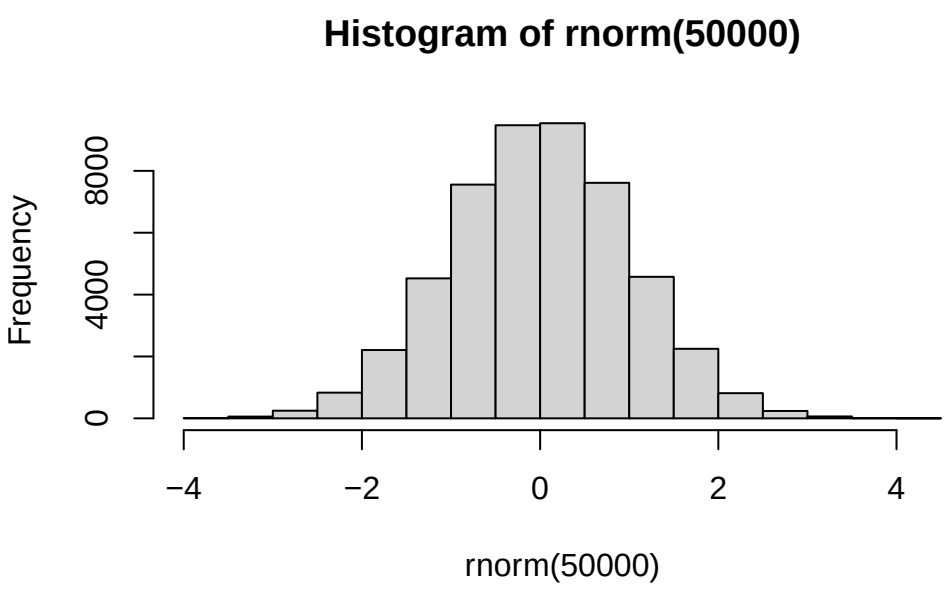
Let's see how it works:

This gives random numbers or n; Its a way to make up random data:

```
rnorm(10)
```

```
 [1] -0.17846018 -0.05324777  0.34158698 -1.11060565 -1.18524918  0.71377779
 [7]  0.18736280  1.16571408 -1.75202341 -0.32003669
```

```
hist( rnorm(50000))
```




```
# mean will be 0 and sd is 1. This is the deafult.
# check the input available
```

Make a little vector with 60 total points that re centered at +3 and half that are at -3.

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

```
 [1]  3.0373001  2.2470264  2.4539856  2.2834873  4.0784247  4.0636421
 [7]  0.8795042  1.3068491  0.7907352  4.2385510  3.6529406  4.4117635
[13]  3.9995130  2.7012992  1.6977610  4.1345080  3.8395134  2.3924632
[19]  3.6323610  2.7305478  2.4683341  3.2372524  3.0423721  3.4072697
[25]  2.6869980  4.4250433  5.0593068  3.8543323  3.2435671 -0.1920145
[31] -3.1342880 -4.2557799 -3.7782107 -3.1704535 -3.1508633 -5.2177055
[37] -1.5414362 -1.4920312 -3.7064578 -3.8424761 -1.8121204 -1.7688647
[43] -2.1805756 -4.9891784 -4.5843033 -3.1721706 -4.0764912 -5.2384689
[49] -3.8621358 -0.4035043 -3.6389355 -3.6674729 -2.4124593 -4.5744518
[55] -3.5640114 -2.1344445 -2.2579821 -2.1372122 -2.5670938 -3.3908708
```

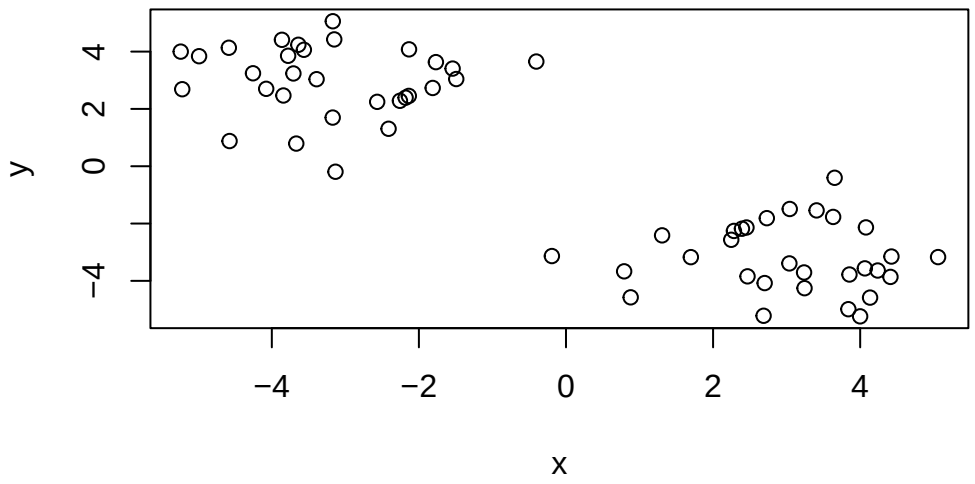
Let's allign it so that it goes -ve to +ve and make a plot to visually see two different clusters.

```r
rev.tmp <- rev(tmp)


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

```
              x          y
 [1,]  3.0373001 -3.3908708
 [2,]  2.2470264 -2.5670938
 [3,]  2.4539856 -2.1372122
 [4,]  2.2834873 -2.2579821
 [5,]  4.0784247 -2.1344445
 [6,]  4.0636421 -3.5640114
 [7,]  0.8795042 -4.5744518
 [8,]  1.3068491 -2.4124593
 [9,]  0.7907352 -3.6674729
[10,]  4.2385510 -3.6389355
[11,]  3.6529406 -0.4035043
[12,]  4.4117635 -3.8621358
[13,]  3.9995130 -5.2384689
[14,]  2.7012992 -4.0764912
[15,]  1.6977610 -3.1721706
[16,]  4.1345080 -4.5843033
[17,]  3.8395134 -4.9891784
[18,]  2.3924632 -2.1805756
[19,]  3.6323610 -1.7688647
[20,]  2.7305478 -1.8121204
[21,]  2.4683341 -3.8424761
[22,]  3.2372524 -3.7064578
[23,]  3.0423721 -1.4920312
[24,]  3.4072697 -1.5414362
[25,]  2.6869980 -5.2177055
[26,]  4.4250433 -3.1508633
[27,]  5.0593068 -3.1704535
[28,]  3.8543323 -3.7782107
[29,]  3.2435671 -4.2557799
[30,] -0.1920145 -3.1342880
[31,] -3.1342880 -0.1920145
[32,] -4.2557799  3.2435671
[33,] -3.7782107  3.8543323
[34,] -3.1704535  5.0593068
[35,] -3.1508633  4.4250433
```

```
[36,] -5.2177055  2.6869980
[37,] -1.5414362  3.4072697
[38,] -1.4920312  3.0423721
[39,] -3.7064578  3.2372524
[40,] -3.8424761  2.4683341
[41,] -1.8121204  2.7305478
[42,] -1.7688647  3.6323610
[43,] -2.1805756  2.3924632
[44,] -4.9891784  3.8395134
[45,] -4.5843033  4.1345080
[46,] -3.1721706  1.6977610
[47,] -4.0764912  2.7012992
[48,] -5.2384689  3.9995130
[49,] -3.8621358  4.4117635
[50,] -0.4035043  3.6529406
[51,] -3.6389355  4.2385510
[52,] -3.6674729  0.7907352
[53,] -2.4124593  1.3068491
[54,] -4.5744518  0.8795042
[55,] -3.5640114  4.0636421
[56,] -2.1344445  4.0784247
[57,] -2.2579821  2.2834873
[58,] -2.1372122  2.4539856
[59,] -2.5670938  2.2470264
[60,] -3.3908708  3.0373001
```

```
plot(x)
```



Let's run the `kmeans()` feature to see how many clusters are there:

```
k <- kmeans(x, centers = 2, nstart=20)
k
```

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

Cluster means:
          x         y
1 -3.190748  2.993488
2  2.993488 -3.190748

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] 85.1613 85.1613
 (between_SS / total_SS =  87.1 %)

Available components:
```

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

```
  #the result tells a lot of data.
```

What's in this result object??

```
  attributes(k)
```

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

$class
[1] "kmeans"
```

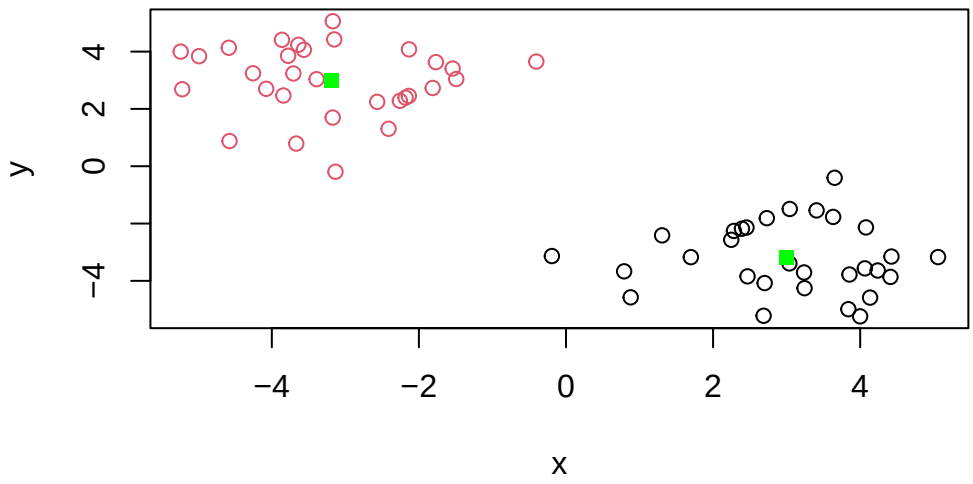
To get in-depth data about the clusters:

```
  k$centers
```

```
          x         y
1 -3.190748  2.993488
2  2.993488 -3.190748
```

Q. Plot your data `x` showing your clustering result and the center point of each cluster.

```
k <- kmeans(x, centers = 2, nstart=20)

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



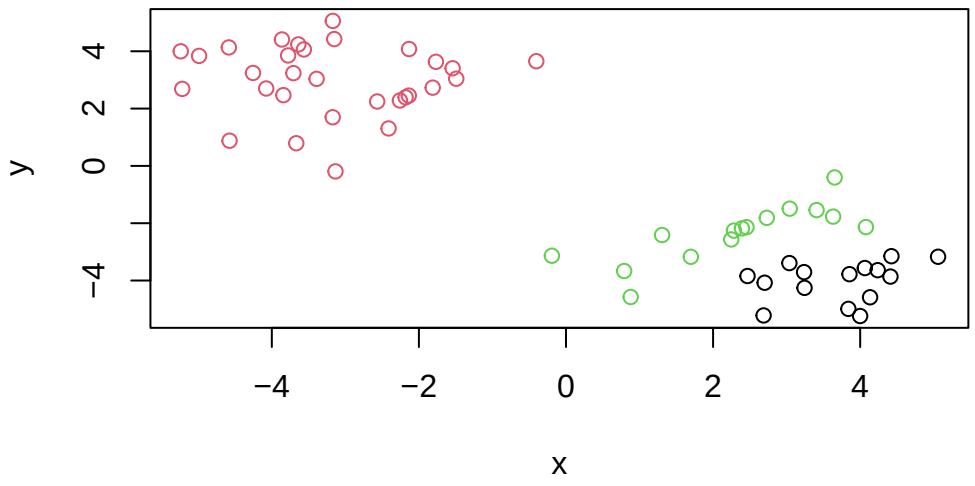
Q. Run kmeans and cluster into 3 groups and plot the results.

```
k2 <- kmeans(x, centers = 3)
k2
```

```
K-means clustering with 3 clusters of sizes 15, 30, 15

Cluster means:
          x         y
1  3.693395 -4.031089
2 -3.190748  2.993488
3  2.293581 -2.350407

Clustering vector:
 [1] 1 3 3 3 3 1 3 3 3 1 3 1 1 1 3 1 1 3 3 3 1 1 3 3 1 1 1 1 1 3 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] 14.69875 85.16130 34.58125
 (between_SS / total_SS =  89.8 %)

Available components:
```

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

```
plot(x, col=k2$cluster)
```



Lets look at something:

Larger number of clusters

```
k$tot.withinss
```

```
[1] 170.3226
```

```
k2$tot.withinss
```

```
[1] 134.4413
```

The big limitation to kmeans() is that it **imposes structure on your data** that you ask for in the first place.

Hierarchical clustering

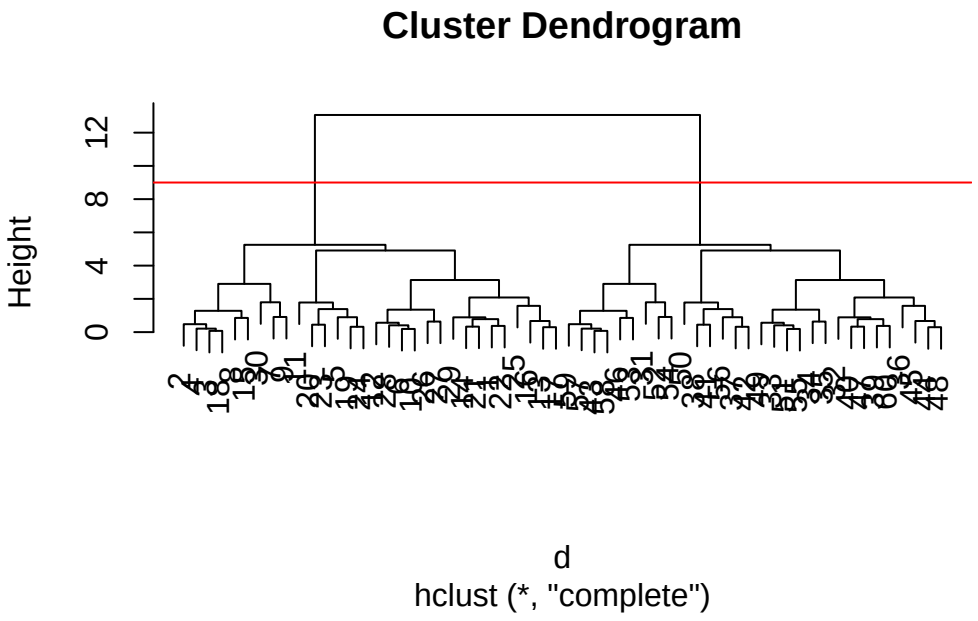
`hclust()` wont work with just the data, if you put it. The data needs to come from `dist()` (a distance matrix).

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

hc
```

```
Call:
hclust(d = d)

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

hc by itself doesn't give you much information so we use a new plot. Like this:

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

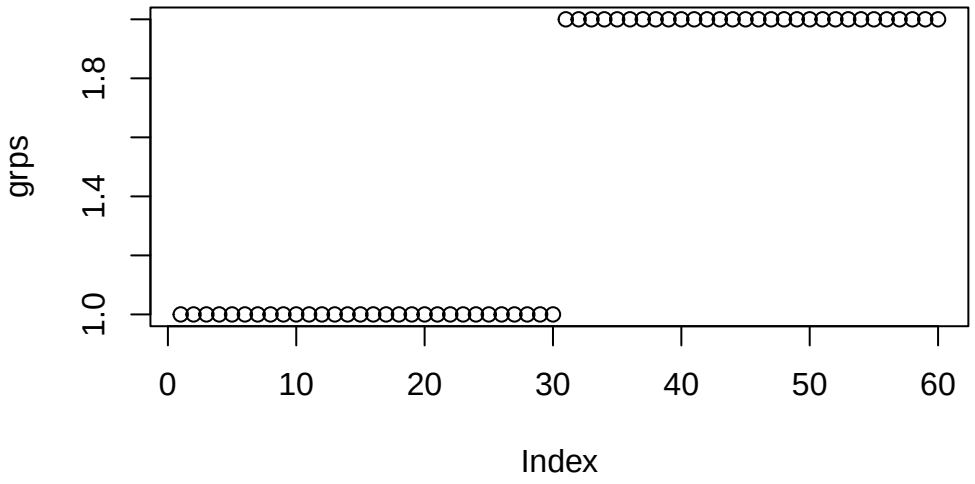


Let's now get the cluster membership vector we need to cut the tree at a given height we choose to seperate out some clusters. The function to do this calls for `cutree()`.

```
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 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
```

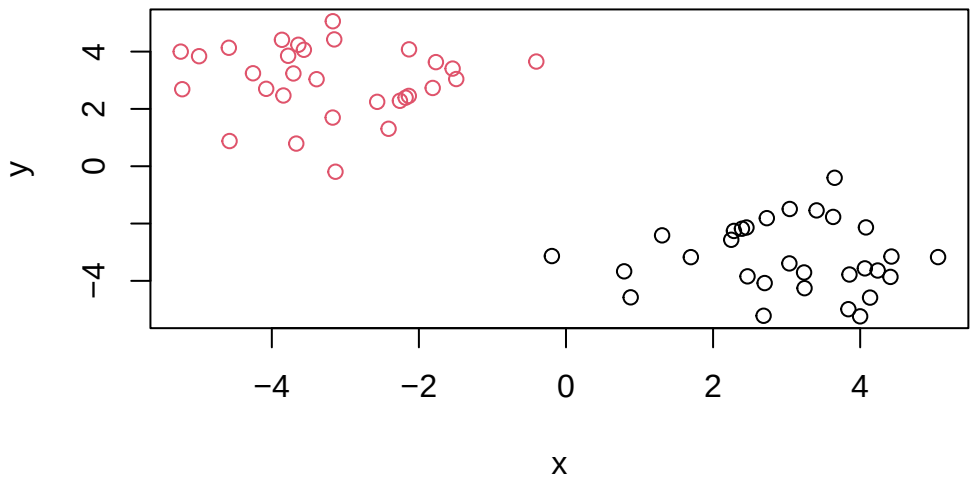
```
plot(grps)
```




```
# You can use cluster (specify the no. of clusters) or by height (h).
```

Q. Plot the results of data (x) and color our hclust.

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



Principal Component Analysis

We will start with PCA of a tiny dataset from UK gov.

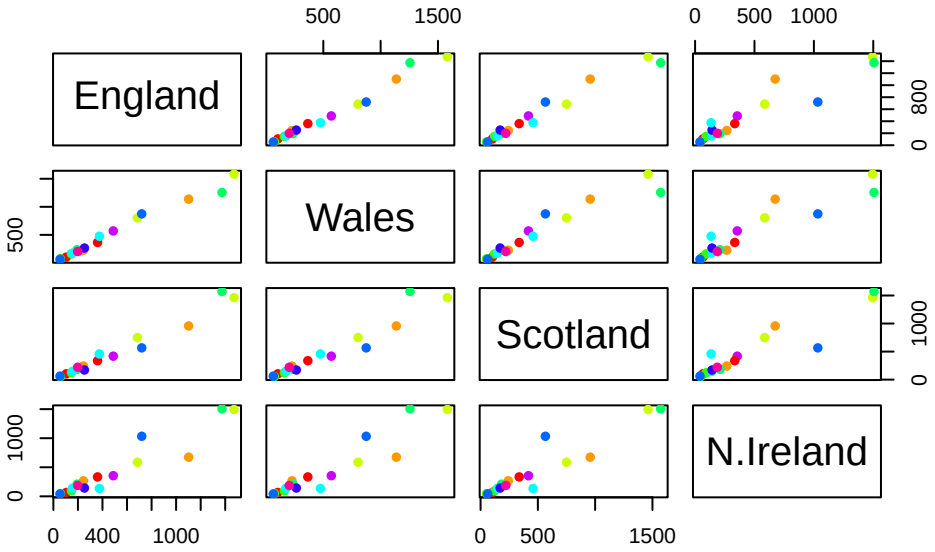
```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names = 1)

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

One useful plot from the lab :

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




```
# But this also doesnt give you much data
```

Enter PCA

The main fucntion to do PCA in "base" R is called `prcomp()`

It wants our foods as the column and the countries as the rows.so we transpose it using `t()`.

```
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
```

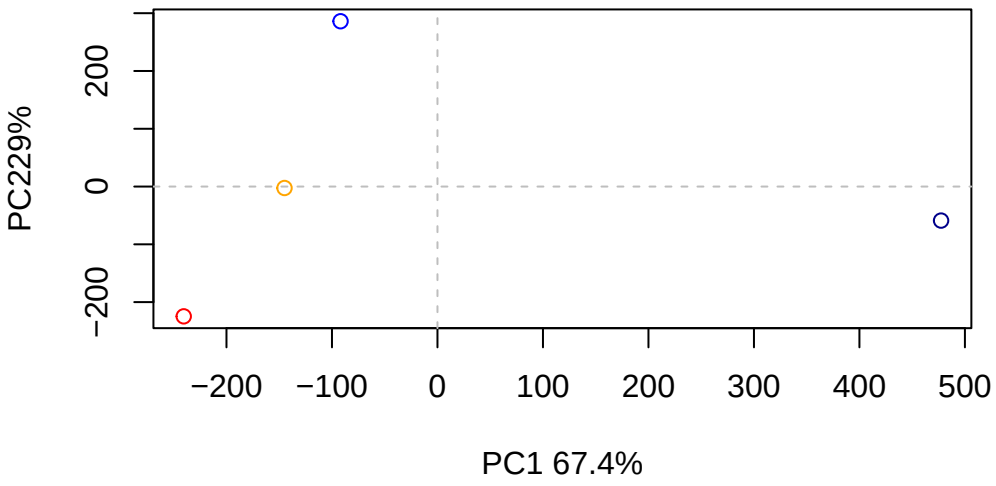
```
attributes(pca)
```

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

$class
[1] "prcomp"
```

```
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
```

```
plot(pca$x[,1], pca$x[,2], xlab = "PC1 67.4%", ylab = "PC229%", col=c("orange", "red", "bl

abline(h=0, col="gray", lty=2)
abline(v=0, col="gray", lty=2)
```


PC229%
200
0
−200
−200
−100
0
100
200
300
400
500
PC1 67.4%