class7: Clustering and PCA

Isabella Franco

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
1 + 1
```

```
[1] 2
```

You can add options to executable code like this

```
[1] 4
```

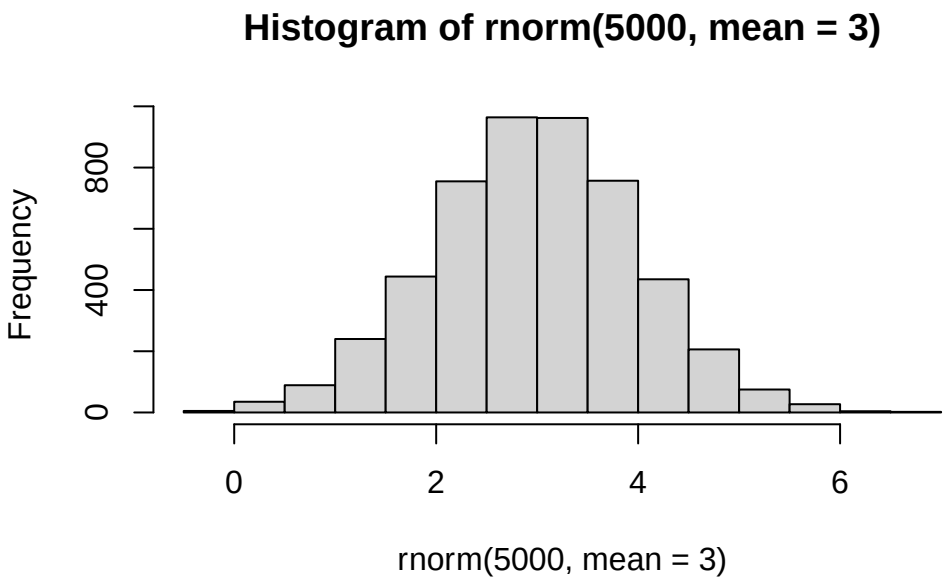
The `echo: false` option disables the printing of code (only output is displayed).

Clustering

First let's make up some data to cluster so we can get a feel for these methods and how to work with them

We can use the `rnorm()` function to get random numbers from a normal distribution around a given `mean.`

```r
hist(rnorm(5000, mean=3))
```

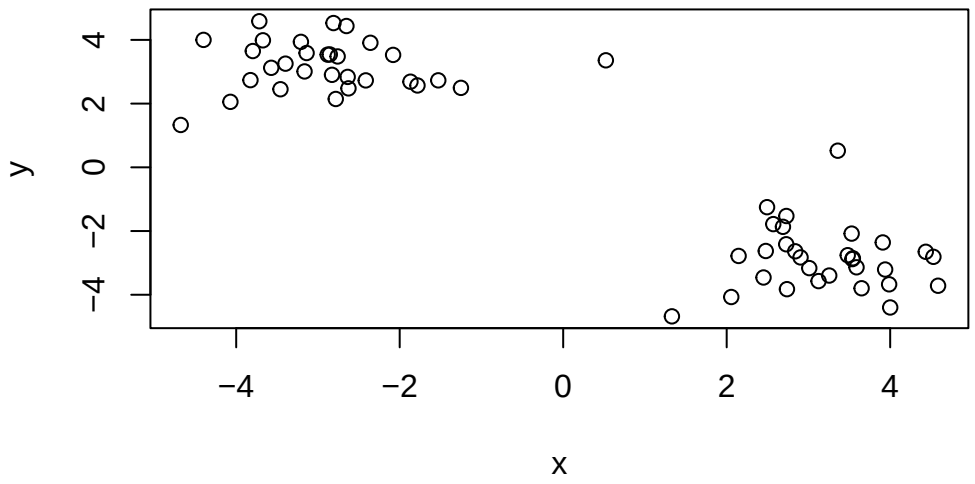


Let's get 30 points with a mean of 3

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

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

```r
plot(x)
```



K- means clustering.

Very popular clustering method that we can use wtih the `kmeans()` function in base R.

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

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

Cluster means:
          x         y
1  3.186967 -2.856342
2 -2.856342  3.186967

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 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

Within cluster sum of squares by cluster:
[1] 48.3571 48.3571
 (between_SS / total_SS =  91.9 %)
```

```
Available components:

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

What is our cluster size?

```
km$size
```

```
[1] 30 30
```

Cluster size?

```
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 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
```

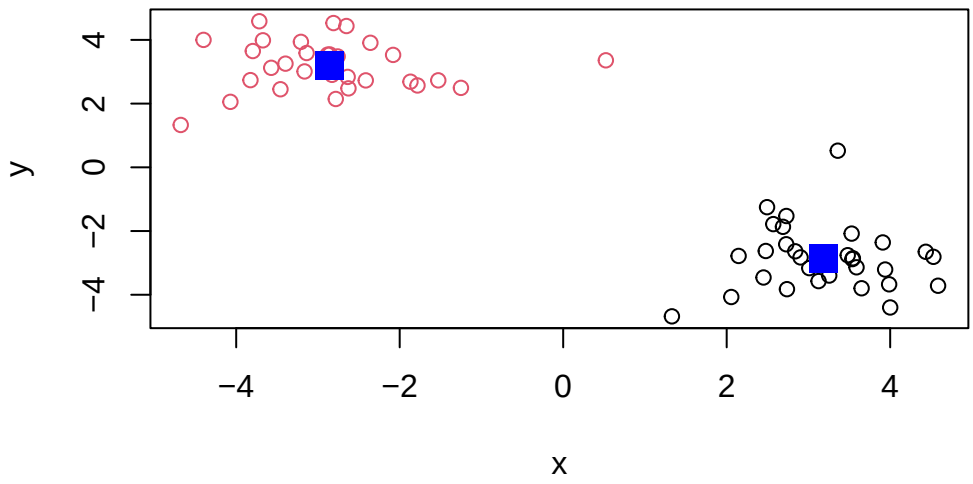
Cluster center?

```
km$centers
```

```
          x         y
1  3.186967 -2.856342
2 -2.856342  3.186967
```

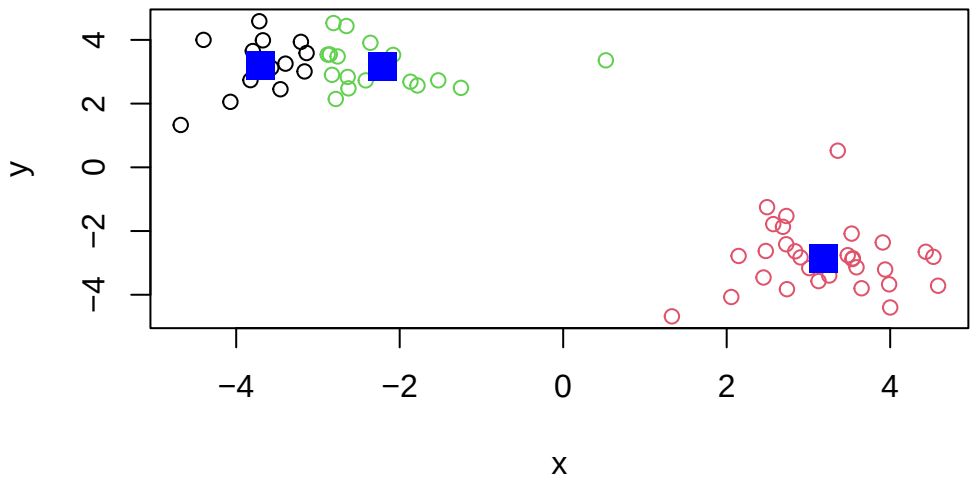
Q. plot x colored by the kmeans clusters assignment and add cluster centers as blue points

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



Q Let's cluster into 3 groups or same x data and make a plot.

```
km<-kmeans(x, centers=3)
plot(x,col=km$cluster)
points(km$centers, col="blue", pch=15, cex=2)
```



Hierarchical Clustering

We can use the `hclust()` function for Hierarchical Clustering.
Unlike `kmeans()` where we could just pass in our data as input, we need to give `hclust()` a "distance matrix".

We will use the `dist()` function to start with.

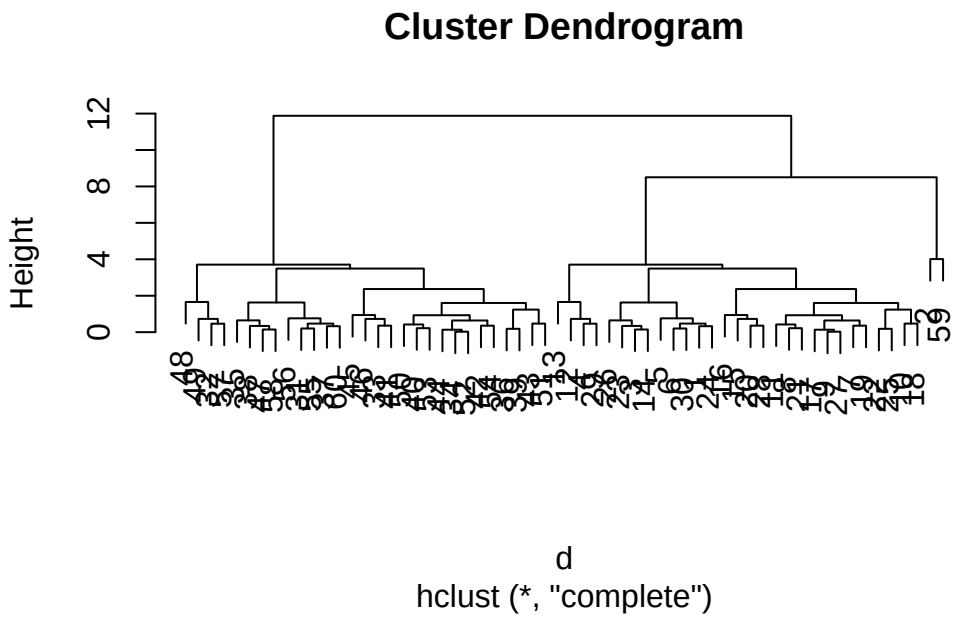
```
d<-dist(x)
```

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

```
Call:
hclust(d = d)

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

```
plot(hc)
```



I can now "cut" my tree with the `cutree()` to yield a cluster membership vector.

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

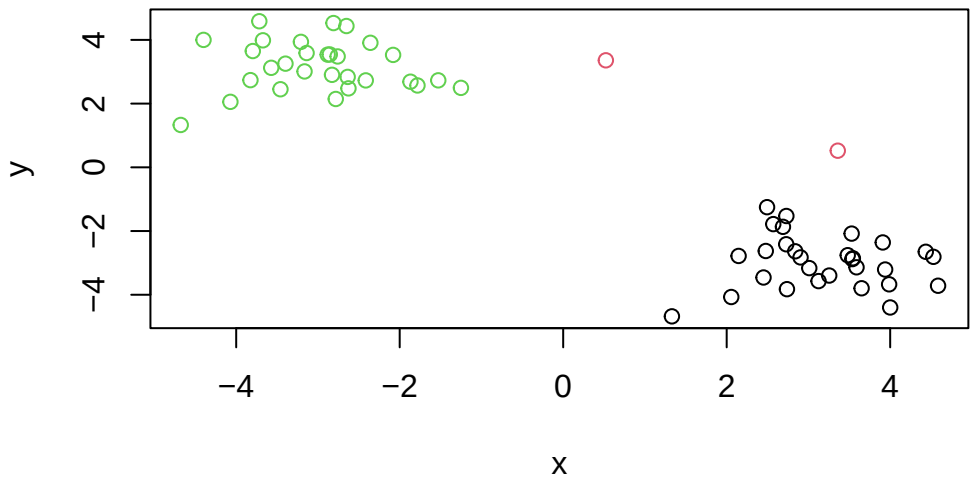
```
 [1] 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3
[39] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3
```

You can also tell `cutree()` to cut where it yields "k" groups.

```
cutree(hc, k=2)
```

```
 [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 1 2
```

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



Principal Component Analysis (PCA)

```
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
```

> Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

Nrow() can be used to give us the number of rows and ncol() can be used to give us the number of columns as shown below.

```
nrow(x)
```

```
[1] 17
```

```
ncol(x)
```

```
[1] 4
```

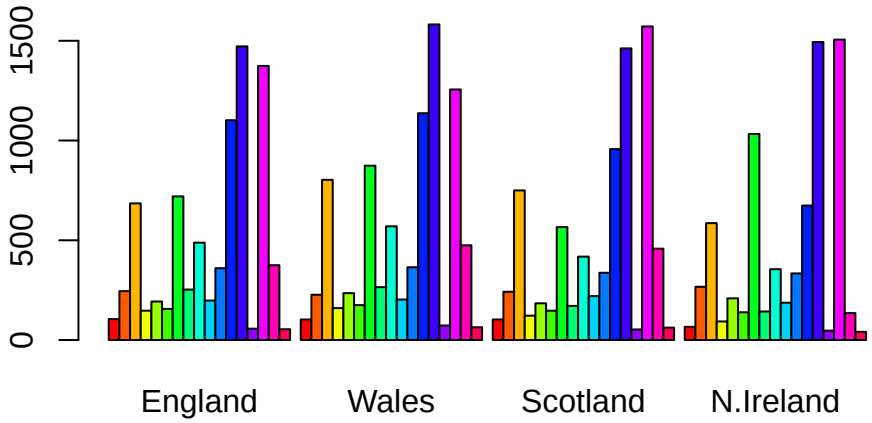
```
head(x)
```

```
               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
```

> Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

I used row.names= as opposed to: rownames(x) <- x[,1] x <- x[,-1] head(x) because this code is destructive as subtracts a column everytime you run it.

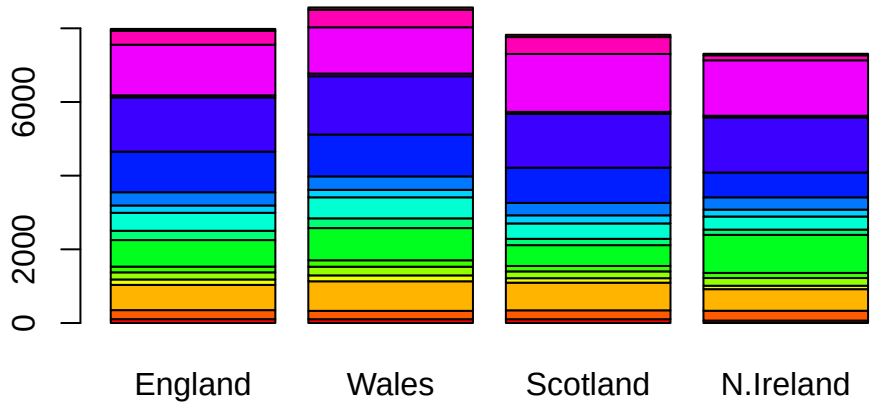
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



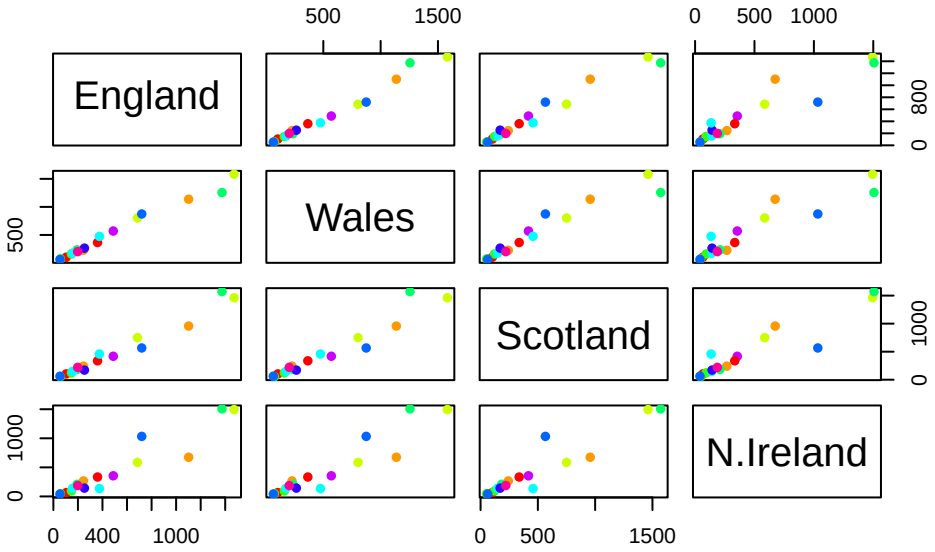
Q3: Changing what optional argument in the above barplot() function results in the following plot?

If I change beside=T, to beside=F, I make a stacked plot which is useless

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




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



Q5: Generating all pairwise plots may help somewhat. Can you make sense of the following code and resulting figure? What does it mean if a given point lies on the diagonal for a given plot?

Although difficult to interpret, this visual representation is more useful than our bar plots as it gives us comparative data points and allows us to visualize how similar the consumption is among different places. Points clustering on the diagonal represent more similar consumption and points straying farther from the diagonal suggests differing consumption of that item between places. The downside to this approach is when comparing so many things such as genes, this would be a very difficult plot to analyze so to make this easier: we use PCA.

The main PCA function in Base R is called `prcomp()` it expects the transpose of our data

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

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 4.189e-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)
```

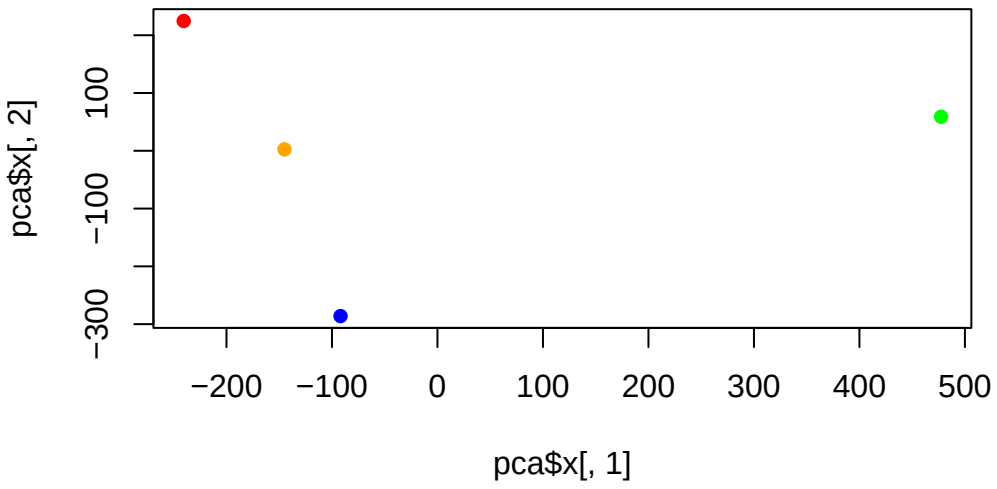
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

```
pca$x
```

```
                PC1         PC2         PC3           PC4
England   -144.99315    2.532999 -105.768945  2.842865e-14
Wales     -240.52915  224.646925   56.475555  7.804382e-13
Scotland   -91.86934 -286.081786   44.415495 -9.614462e-13
N.Ireland  477.39164   58.901862    4.877895  1.448078e-13
```

```
plot(pca$x[,1],pca$x[,2],col=c("orange","red","blue","green"), pch=16)
```



Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set? In terms of this data set, N. Ireland has the largest difference in food consumption compared to the rest of the UK.