Class 7: Machine Learning 1

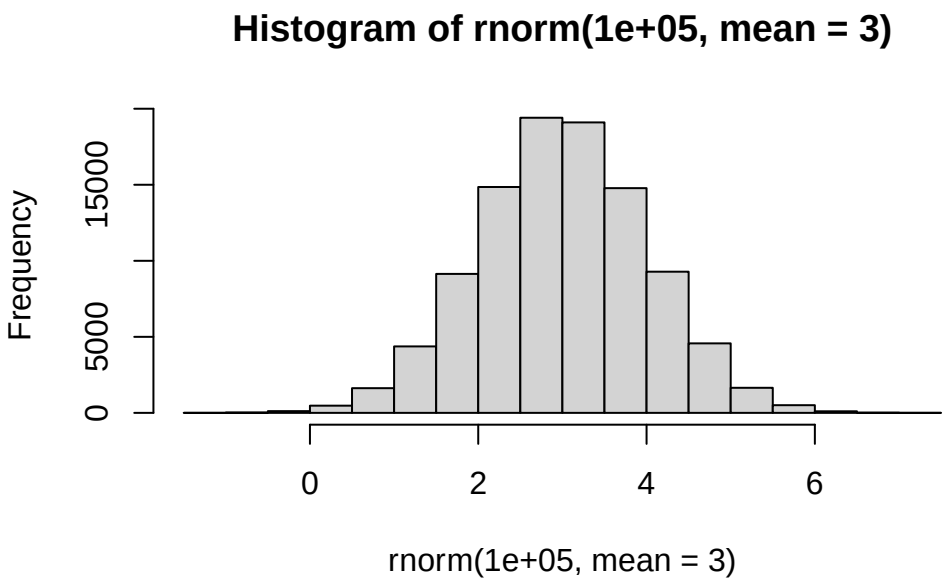
Dani Baur (A16648266)

Today we will start our multi-part exploration of some key machine learning methods. We will begin with clustering- finding groupings in data and then dimensionality reduction.

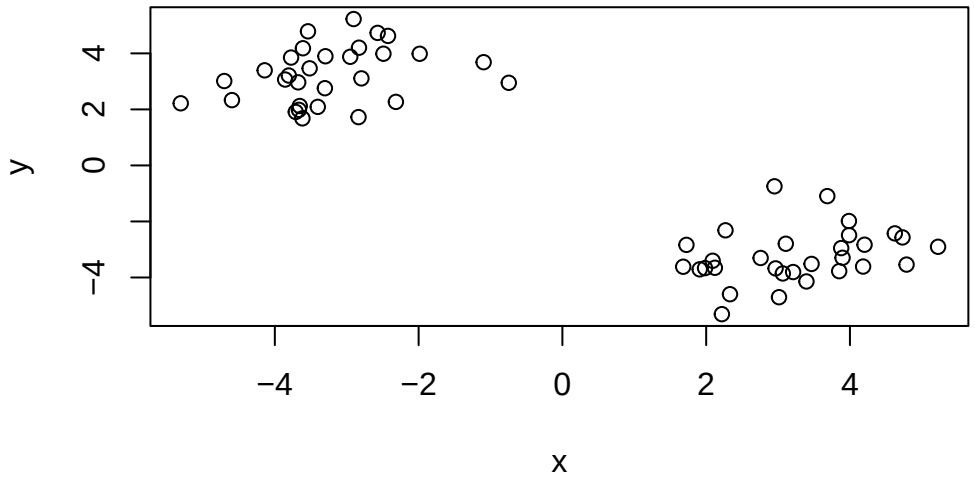
Clusting

Let's start with "k-means" clustering. The main function in base R for this `k(means)`.

```
#make up some data
hist(rnorm(100000, mean=3))
```



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



Now let's try out `k(means)`.

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

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

Cluster means:
          x         y
1 -3.237277  3.243383
2  3.243383 -3.237277

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] 55.7962 55.7962
 (between_SS / total_SS =  91.9 %)

Available components:

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

```
attributes(km)
```

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

$class
[1] "kmeans"
```

Q. how many points in each cluster?

```
km$size
```

```
[1] 30 30
```

Q. what component of your result object details cluster assignment/membership?

```
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
```

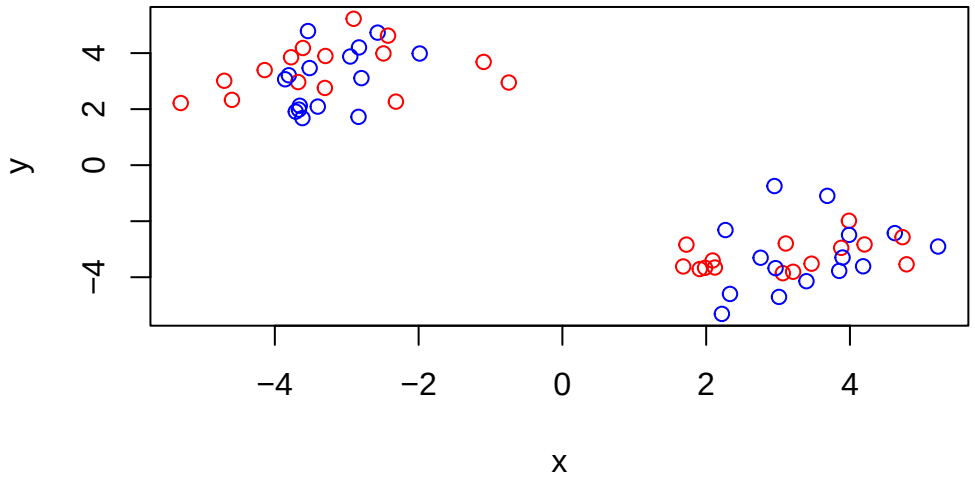
Q. what are centers/means values of each cluster?

```
km$centers
```

```
          x         y
1 -3.237277  3.243383
2  3.243383 -3.237277
```

Q. make a plot of your data showing your clustering results.

```r
plot(x, col=c("red", "blue"))
```

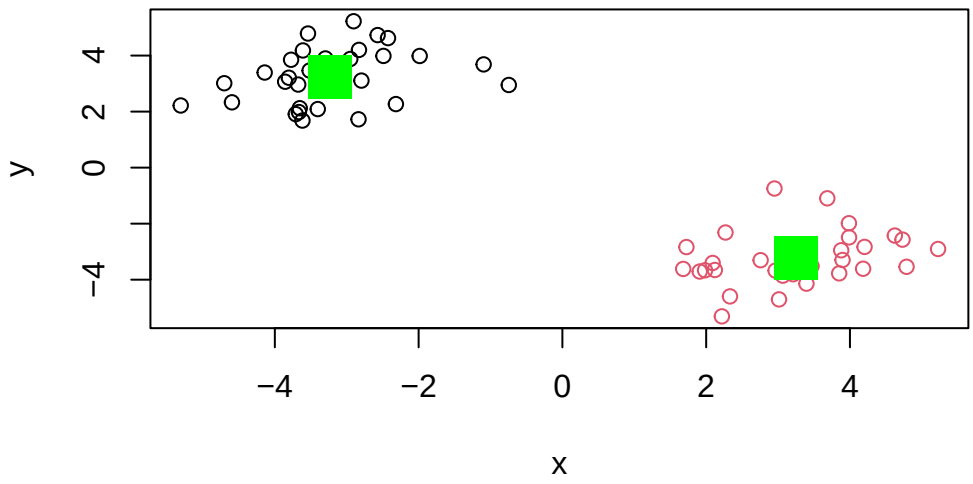



```r
c(1:5) + c(100,1)
```

```
Warning in c(1:5) + c(100, 1): longer object length is not a multiple of
shorter object length
```

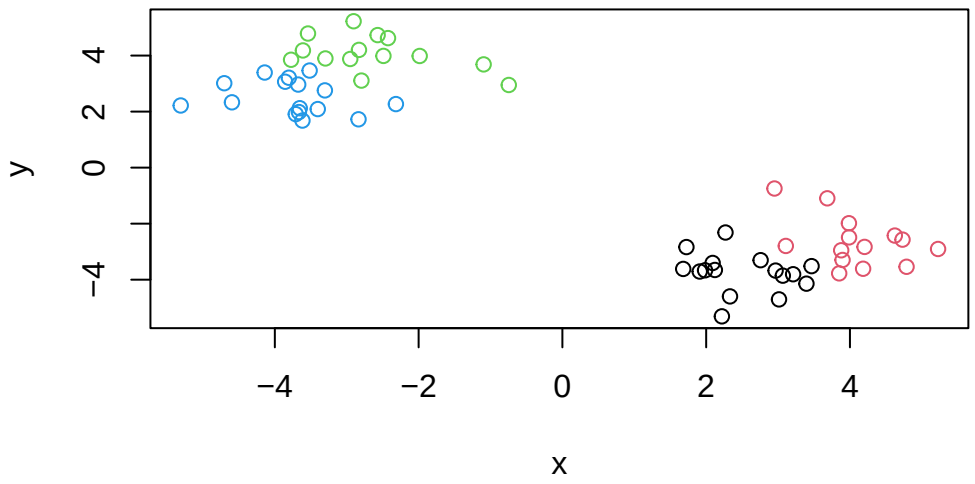
```
[1] 101   3 103   5 105
```

```r
plot(x, col=km$cluster)
points(km$centers, col="green", pch=15, cex=3)
```



Q. run `k(means)` again and cluster in 4 groups and plot the results.

```
km4 <- kmeans(x, centers=4)
plot(x, col=km4$cluster)
```



Hierarchical Clustering

This form of clustering aims to reveal the structure in your data by progressively grouping points into a even smaller number of clusters.

The main function in base R for this called `hclust()`. This function does not take our input data directly but wants a "distance matrix" that details how (dis)similar all our input points are to each other.

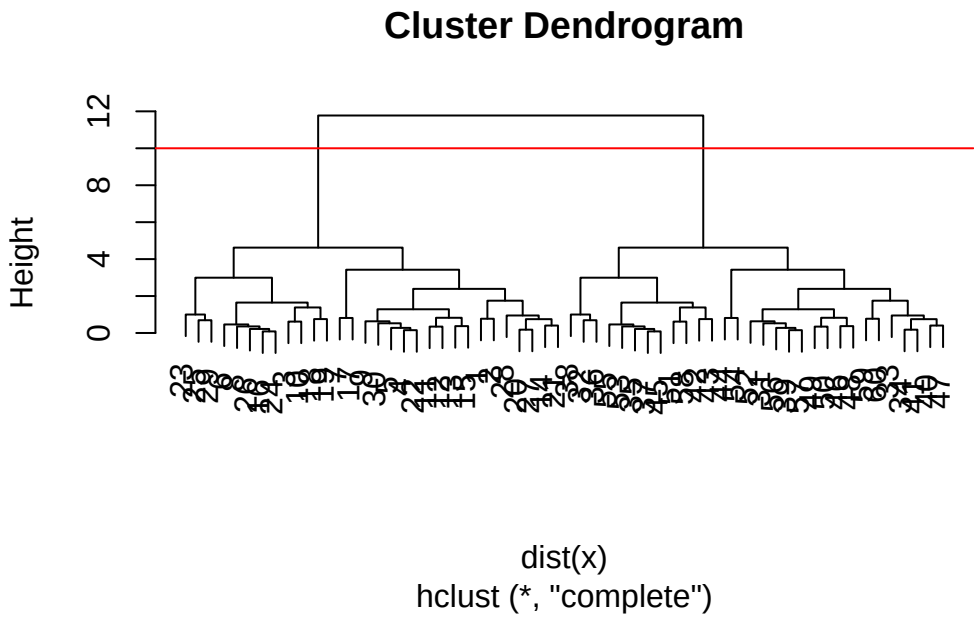
```
hc <- hclust(dist(x))
hc
```

```
Call:
hclust(d = dist(x))

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

The print out above is not very useful (unlink that from kmeans) but there is a useful `plot()` method.

```
plot(hc)
abline(h=10, col="red")
```

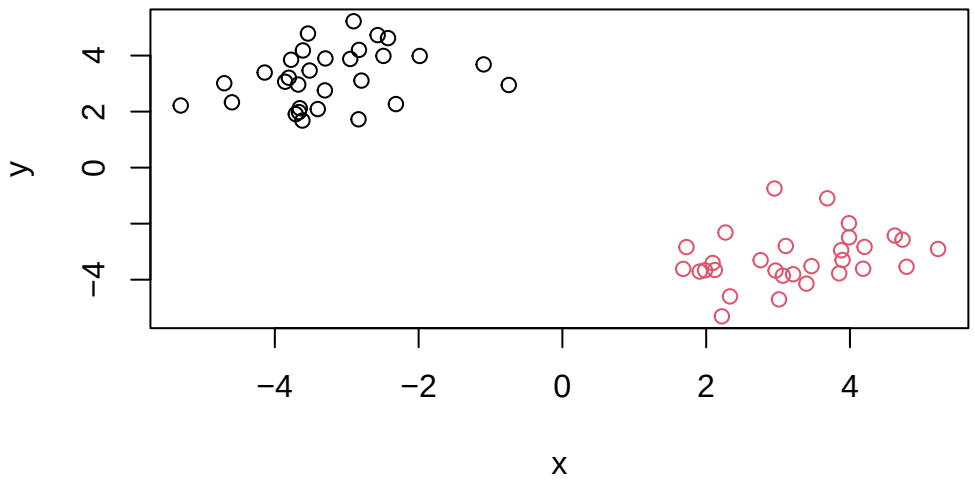


To get my main result (my cluster membership vector), I need to "cut" my tree using the function `cutree()`.

```
grps <- cutree(hc, h=10)
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
```

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



Lab 7: PCA

The goal of PCA is to reduce the dimensionality of a dataset down to some smaller subset of new variables (called PCs) that are useful bases for further analysis, like visualization, clustering, etc.

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

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

```
dim(x)
```

```
[1] 17  5
```

→ There are 17 rows and 5 columns in data frame x.

```
# checking the data
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
```

```
# don't want food types to be its own column
rownames(x) <- x[,1]
x <- x[,-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
```

```
# checking to see if change applied to column number
dim(x)
```

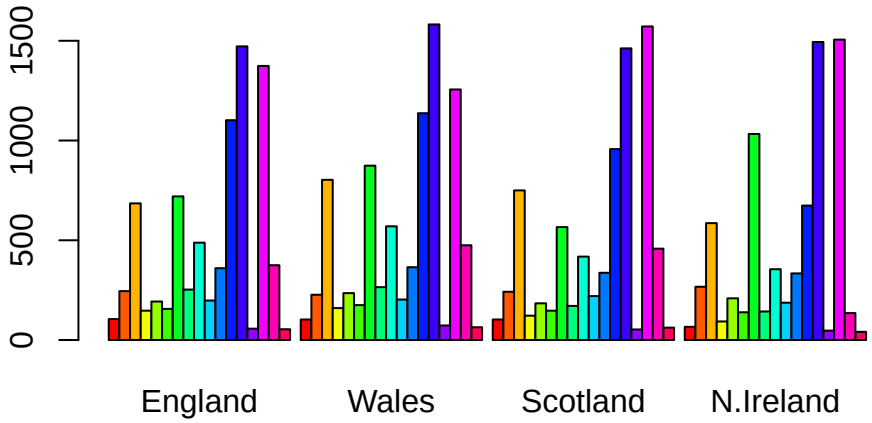
```
[1] 17  4
```

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?

```
x <- read.csv(url, row.names=1)
```

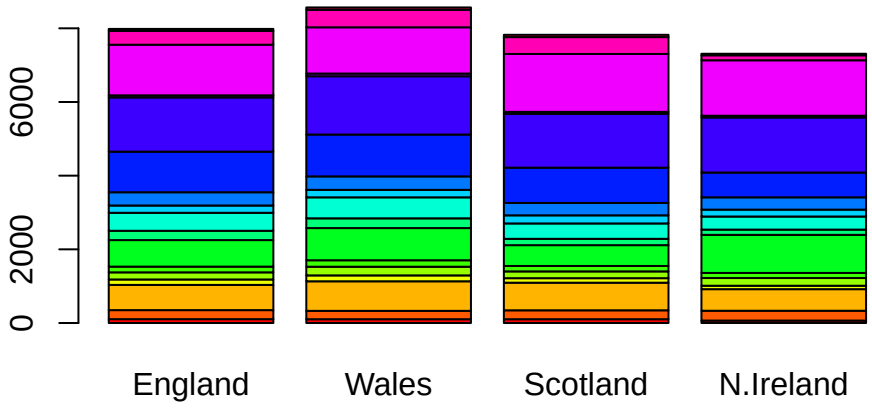
→ The second method shown above is more efficient and concise but it assumes that the data is listed in a way that could be presented better. The first method allows you to see the data first then make adjustments as you see fit.

```
# spotting differences and trends
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?

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



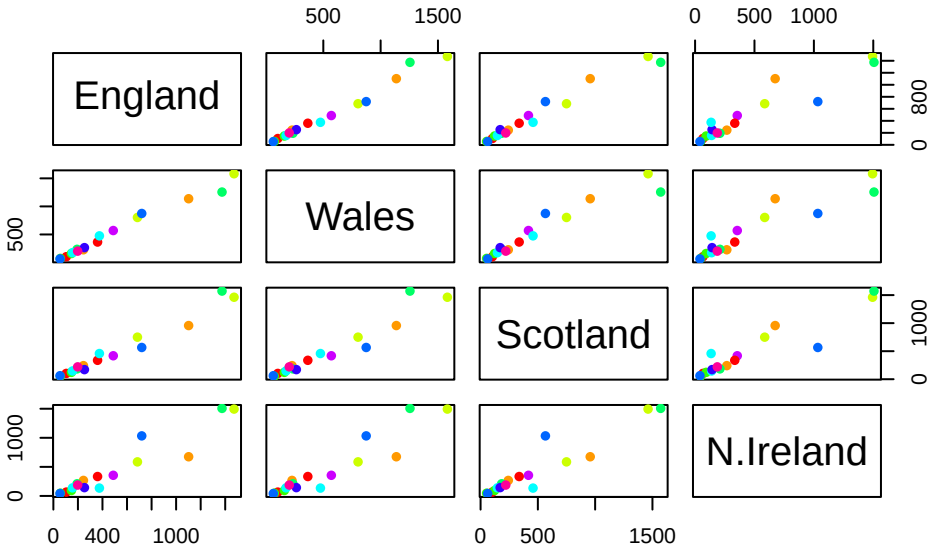
→ Changing beside=TRUE to beside=FALSE changes the result of the graph.

The so-called "pairs" plot can be useful for small datasets:

```
rainbow(nrow(x))
```

```
 [1] "#FF0000" "#FF5A00" "#FFB400" "#F0FF00" "#96FF00" "#3CFF00" "#00FF1E"
 [8] "#00FF78" "#00FFD2" "#00D2FF" "#0078FF" "#001EFF" "#3C00FF" "#9600FF"
[15] "#F000FF" "#FF00B4" "#FF005A"
```

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



So the pairs plot is useful for small datasets but it can be lots of work to interpret and gets untractable for larger datasets.

So PCA to the rescue…

The main function to do PCA in base R is called `prcomp()`. This function wants the transpose of our data in this case.

```
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"
```

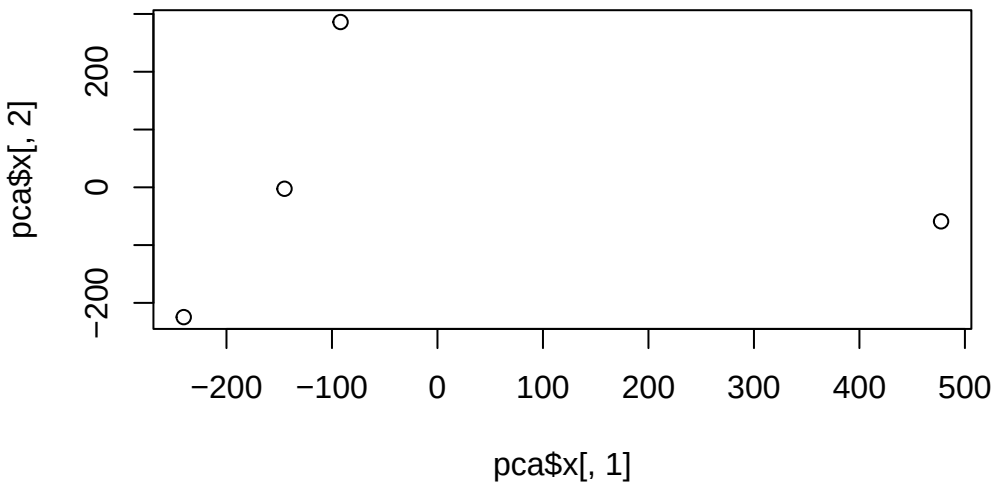
```
$class
[1] "prcomp"
```

```
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
```

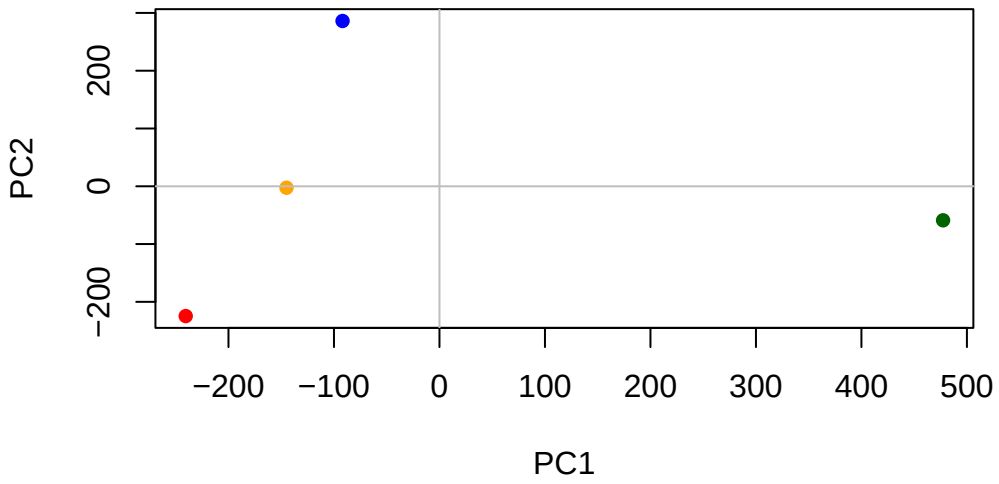
A major PCA result viz is called a "PCA plot" (aka a score plot, a bi-plot, PC1 vs. PC2 plot, ordientation plot)

```
plot(pca$x[,1],pca$x[,2])
```




```
mycols <- c("orange","red","blue","darkgreen")
plot(pca$x[,1],pca$x[,2], col=mycols, pch=16, xlab="PC1", ylab="PC2")
abline(h=0, col="gray")
```

```
abline(v=0, col="gray")
```



Another important output from PCA is called the "loadings" vector or the "rotation" component- this tell us how much the original variables (the foods in this case) contribute to the new PCs.

```
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
```

PCA looks to be a super useful method for gaining some insight into high dimensional data that is difficult to examine in other ways.