Class7(MachineLearning)

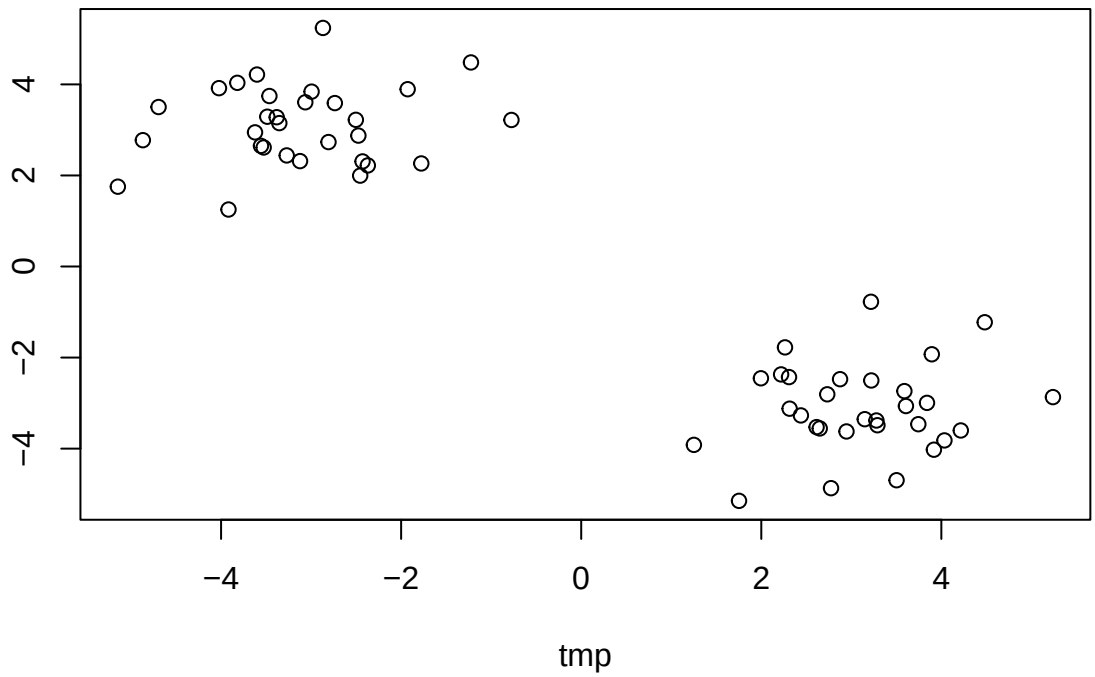
Clustering Methods

Find groups (a.k.a.) clusters in my data

K-means Clustering

```
#Generate some example data for clustering (with 2 clear groups)
tmp <- c(rnorm(30, -3), rnorm(30, 3))
x <- cbind(tmp, rev(tmp))

plot(x)
```




```
#Function: rnorm(number values, mean, st.dev) - generates numbers from norm. dist.
#Function: cbind(dataset, dataset, etc.) - puts vectors into data frame as columns
#Function: rev(dataset) - reverses the dataset (a,b,c) -> (c,b,a)
```

```r
#The 'kmeans()' function does k-means clustering...
k <- kmeans(x, center=4, nstart=20)
k$size    #returns cluster size
```

```
## [1] 14 16 12 18
```

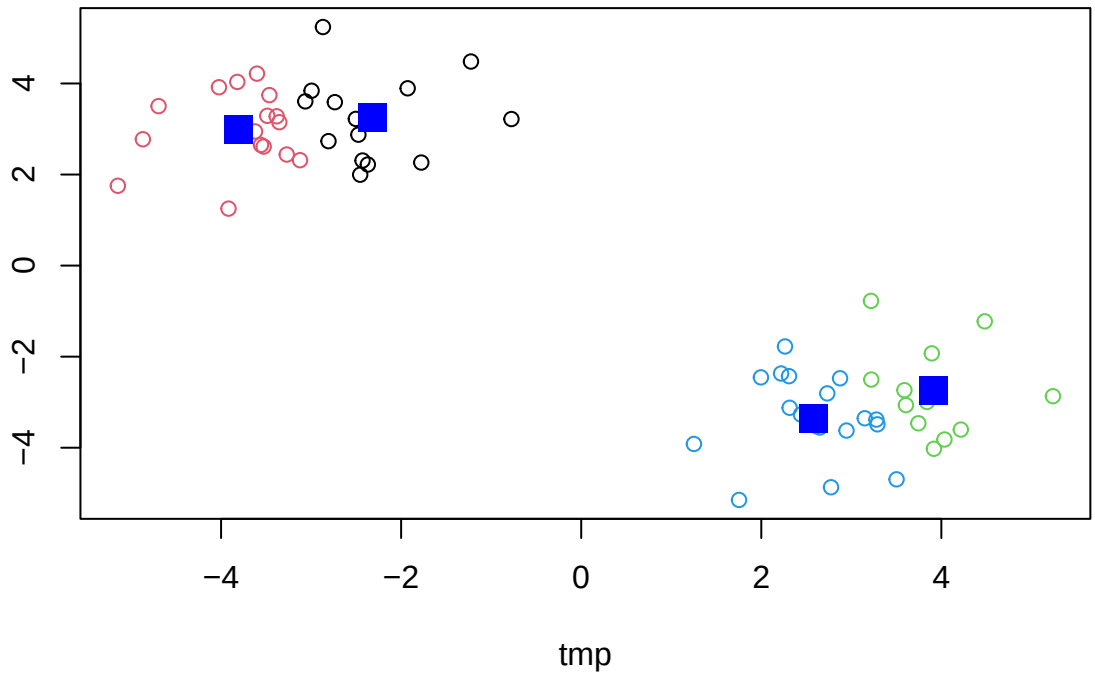
```r
k$cluster #returns cluster assignment (1/2 vector)
```

```
##  [1] 1 2 2 1 2 1 1 1 1 2 1 1 2 1 2 1 1 2 2 2 2 2 2 2 1 2 2 1 2 1 4 4 3 4 3 4 3 4
## [39] 4 3 4 4 4 4 3 4 3 4 3 3 3 4 4 4 3 4 3 4 4 3
```

```r
k$centers #returns coordinates of cluster centers
```

```
##         tmp
## 1 -2.314772  3.249148
## 2 -3.803617  2.992362
## 3  3.917072 -2.750202
## 4  2.575611 -3.347903
```

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

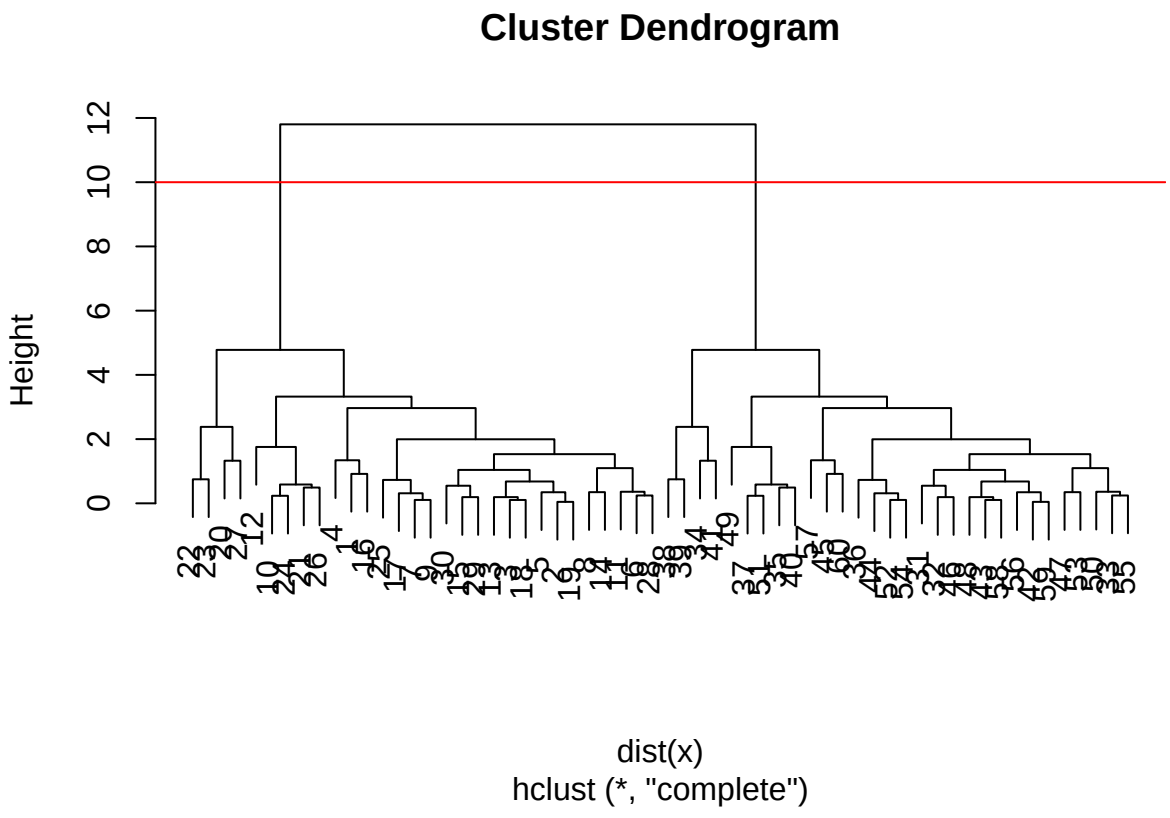


Hierarchical Clustering

```
#The 'huclust()' function needs a distance matrix as input, not a set of the original data. For this, w
hc <- hclust(dist(x))
hc
```

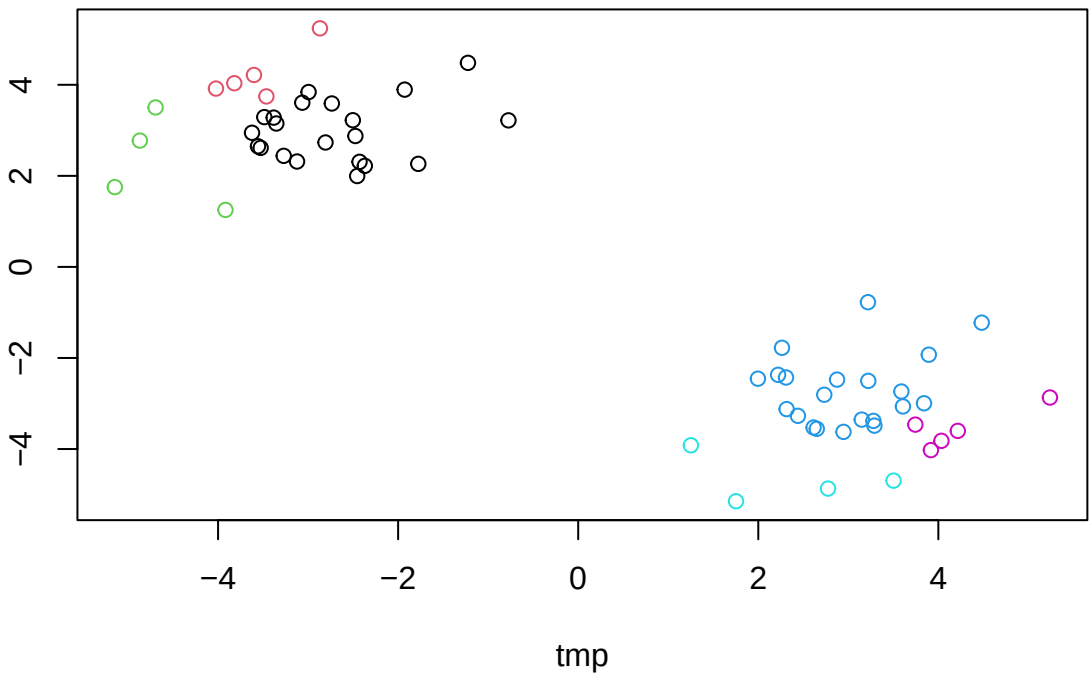
```
##
## Call:
## hclust(d = dist(x))
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 60
```

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




```
#To get our grouping, we must cut the tree
#To cut by a given height, 'h=' or into given number of clusters, 'k='
#Function: cutree(hclust result vector, h or k argument)
cut <- cutree(hc, h=3)

plot(x, col=cut)
```



PCA: Principal Component Analysis

```
#Principal components (PCs) are new low dimensional axis (or surfaces) closes to the observations
#The first PC - the first axis - follows a "best fit" line
#The second PC is perpendicular to the first PC

#UK Foods PCA Analysis
foods <- read.csv("C:\\Users\\rodeo\\Documents\\School\\BIMM 143\\R Projects\\Class07\\UK_foods.csv") #
dim(foods) #the file has 17 rows and 5 columns (row, col)
```

```
## [1] 17  5
```

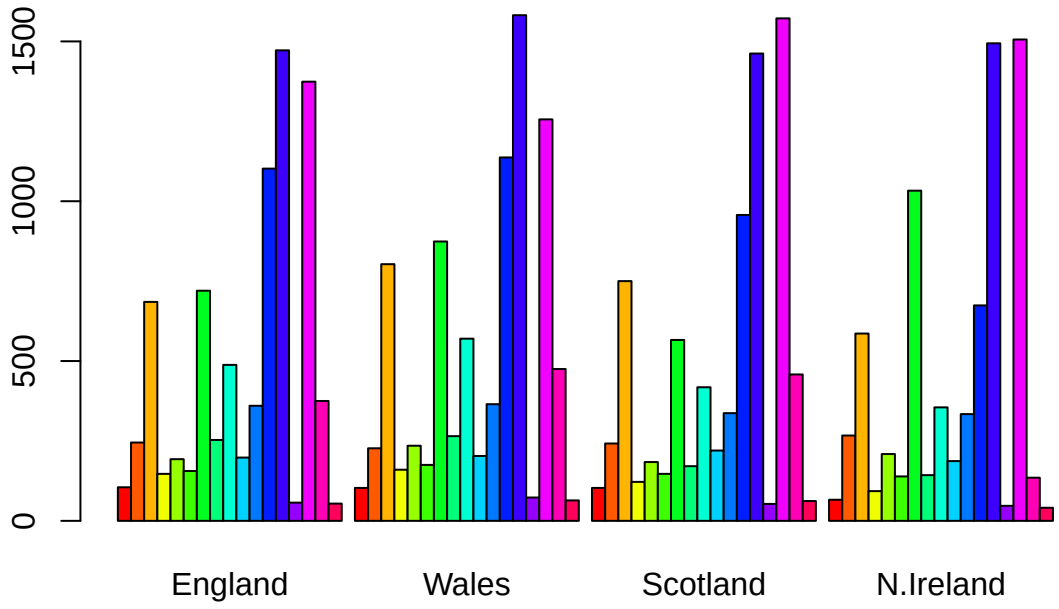
```
head(foods)
```

```
##                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
```

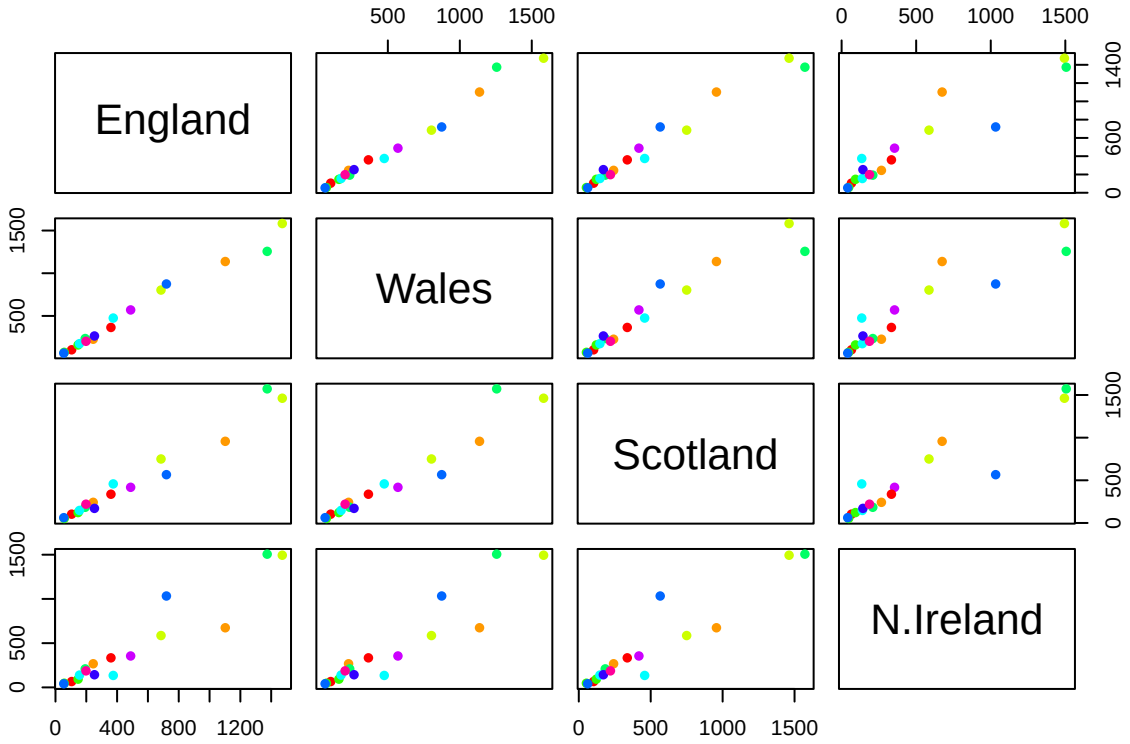
```
#fix row names in first column to be actual row names
#alternative easy approach: 'x <- read.csv(url, row.names=1)'
rownames(foods) <- foods[,1]
foods <- foods[,-1]
head(foods)
```

```
##                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
```

```
#plot the data to see trends
#'barplot()' uses vector or matrix as input -> use 'as.vector()' or 'as.matrix()'
barplot(as.matrix(foods), beside=T, col=rainbow(nrow(foods)))
```




```
#'pairs()' puts variables in omparative charts together
pairs(foods, col=rainbow(10), pch=16)
```



Actual PCA Now

```
#The PCA function that comes pre-built with R is 'prcomp()', which needs an input with variables in the
#Function: t(dataset) - switches the axes of the data table
pca <- prcomp( t(foods) )
pca
```

```
## Standard deviations (1, .., p=4):
## [1] 3.241502e+02 2.127478e+02 7.387622e+01 4.188568e-14
##
## Rotation (n x k) = (17 x 4):
##                              PC1          PC2         PC3          PC4
## Cheese               -0.056955380 -0.016012850 -0.02394295 -0.691718038
## Carcass_meat          0.047927628 -0.013915823 -0.06367111  0.635384915
## Other_meat           -0.258916658  0.015331138  0.55384854  0.198175921
## Fish                 -0.084414983  0.050754947 -0.03906481 -0.015824630
## Fats_and_oils        -0.005193623  0.095388656  0.12522257  0.052347444
## Sugars               -0.037620983  0.043021699  0.03605745  0.014481347
## Fresh_potatoes        0.401402060  0.715017078  0.20668248 -0.151706089
## Fresh_Veg            -0.151849942  0.144900268 -0.21382237  0.056182433
## Other_Veg            -0.243593729  0.225450923  0.05332841 -0.080722623
## Processed_potatoes   -0.026886233 -0.042850761  0.07364902 -0.022618707
## Processed_Veg        -0.036488269  0.045451802 -0.05289191  0.009235001
```

```
## Fresh_fruit         -0.632640898  0.177740743 -0.40012865 -0.021899087
## Cereals             -0.047702858  0.212599678  0.35884921  0.084667257
## Beverages           -0.026187756  0.030560542  0.04135860 -0.011880823
## Soft_drinks          0.232244140 -0.555124311  0.16942648 -0.144367046
## Alcoholic_drinks    -0.463968168 -0.113536523  0.49858320 -0.115797605
## Confectionery       -0.029650201 -0.005949921  0.05232164 -0.003695024
```

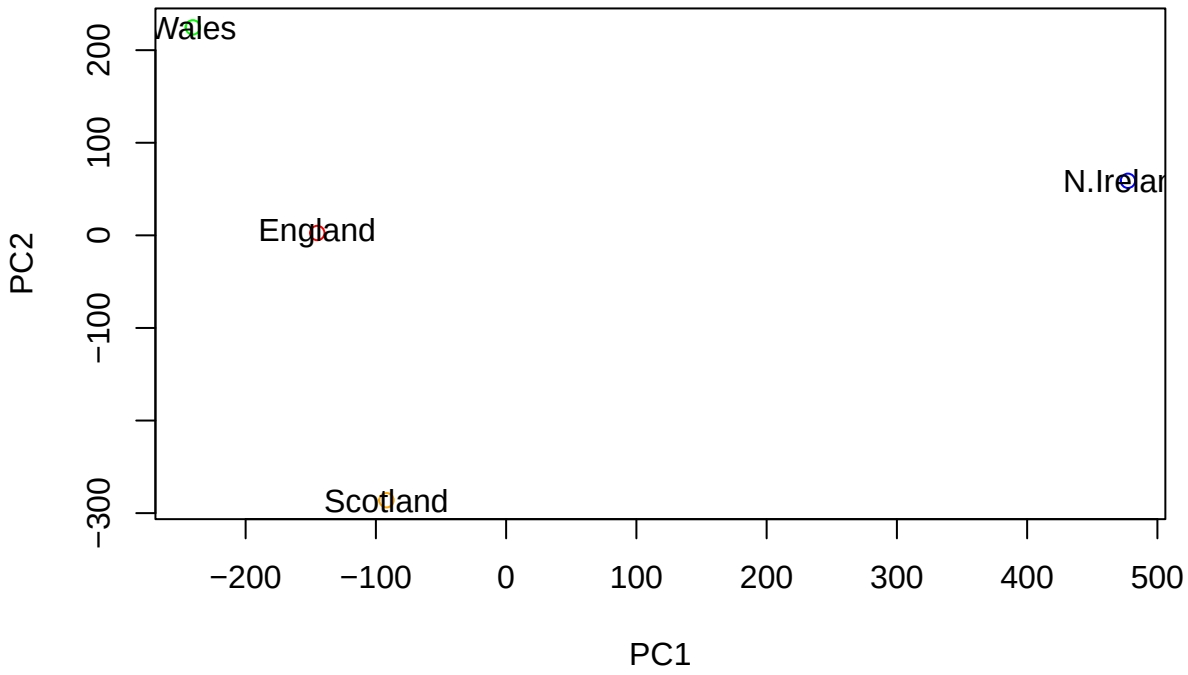
```
summary(pca)
```

```
## 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
```

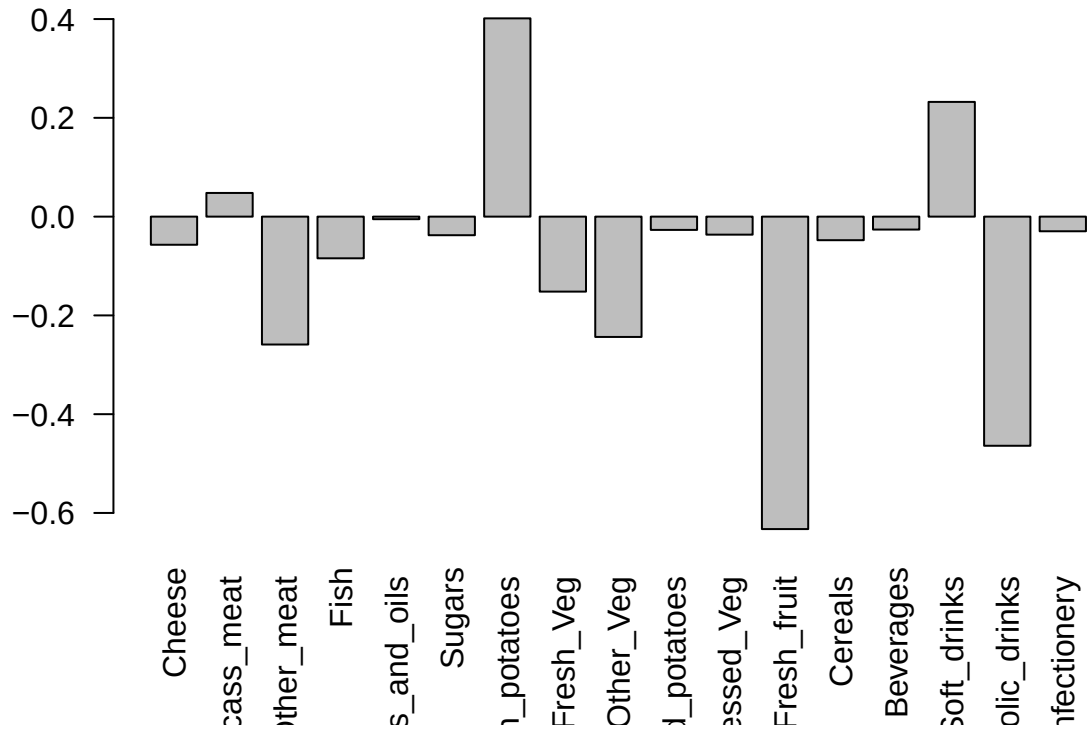
```
#What is in this returned pca object?
attributes(pca)
```

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

```
plot(pca$x[,1:2], col=c("red", "green", "orange", "blue", pch=18))
text( pca$x[,1], pca$x[,2], labels=colnames(foods))
```



```
#las = label rotation (accepts 0 thru 3)
barplot(pca$rotation[,1], las=2)
```



Other PCA Visualizations (RNA_seq biplot)

```
#Get the data (gene expression this time)
url2 <- "https://tinyurl.com/expression-CSV"
rna.data <- read.csv(url2, row.names=1)
head(rna.data)
```

```
##        wt1 wt2  wt3  wt4 wt5 ko1 ko2 ko3 ko4 ko5
## gene1  439 458  408  429 420  90  88  86  90  93
## gene2  219 200  204  210 187 427 423 434 433 426
## gene3 1006 989 1030 1017 973 252 237 238 226 210
## gene4  783 792  829  856 760 849 856 835 885 894
## gene5  181 249  204  244 225 277 305 272 270 279
## gene6  460 502  491  491 493 612 594 577 618 638
```

```
#How many genes are there?
nrow(rna.data) #100 genes
```

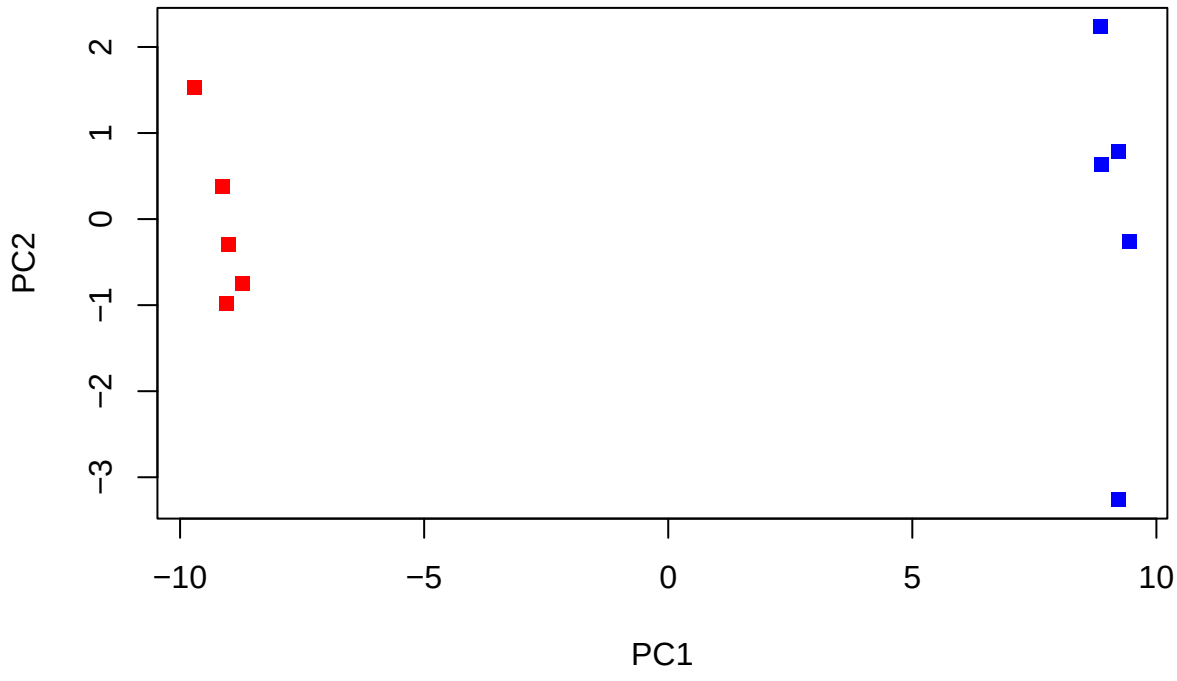
```
## [1] 100
```

```
# Again we have to take the transpose of our data
pca <- prcomp(t(rna.data), scale=TRUE)
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     9.6237 1.5198 1.05787 1.05203 0.88062 0.82545 0.80111
## Proportion of Variance 0.9262 0.0231 0.01119 0.01107 0.00775 0.00681 0.00642
## Cumulative Proportion  0.9262 0.9493 0.96045 0.97152 0.97928 0.98609 0.99251
##                           PC8     PC9      PC10
## Standard deviation     0.62065 0.60342 3.348e-15
## Proportion of Variance 0.00385 0.00364 0.000e+00
## Cumulative Proportion  0.99636 1.00000 1.000e+00
```

```
#Make the score plot...

## Simple unpolished plot of pc1 and pc2 (plot PC1 vs. PC2)
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", pch=15, col=c(rep("red", 5), rep("blue", 5))) # must
```




```
#OR: kmeans(pca$x[,1], centers=2) -> for color
```

```
#Color by k-means cluster
url2 <- "https://tinyurl.com/expression-CSV"
rna.data <- read.csv(url2, row.names=1)
```

```
pca <- prcomp( t(rna.data))
k <- kmeans(pca$x[,1:2], center=2)

plot(pca$x[,1], pca$x[,2], col=k$cluster, pch=15, xlab="PC1", ylab="PC2")
```