

# Machine Learning 1

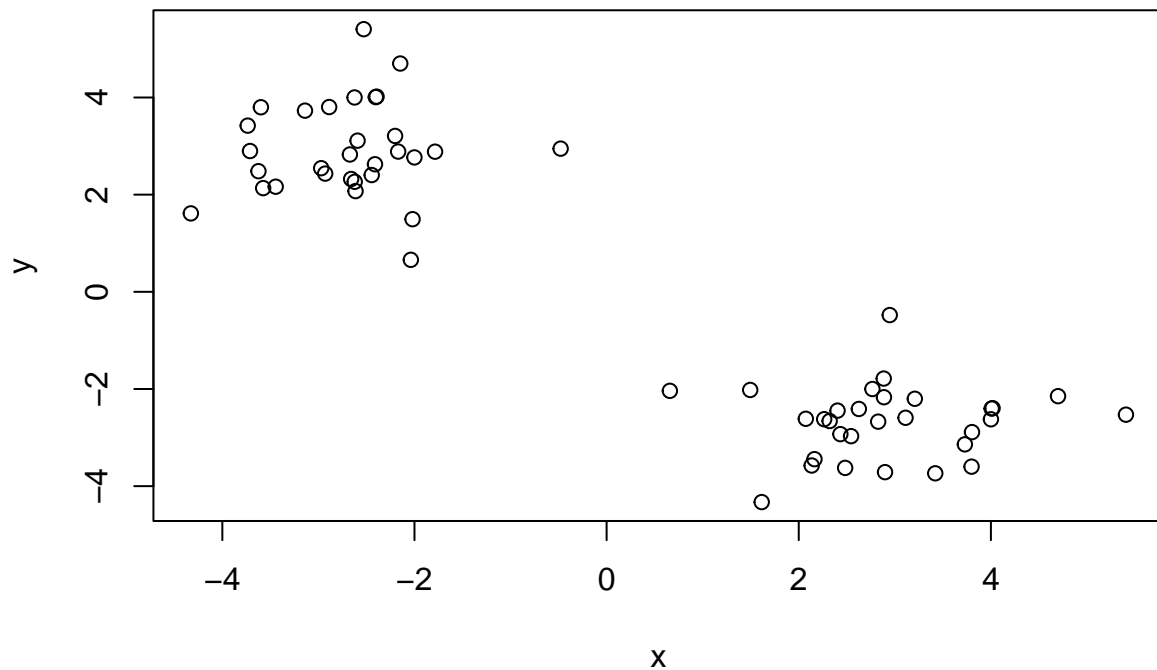
Chloe J. Welch

10/22/2021

## Clustering Methods

Kmeans clustering in R is done with the `kmeans()` function. Here, we make up some data to test and learn with.

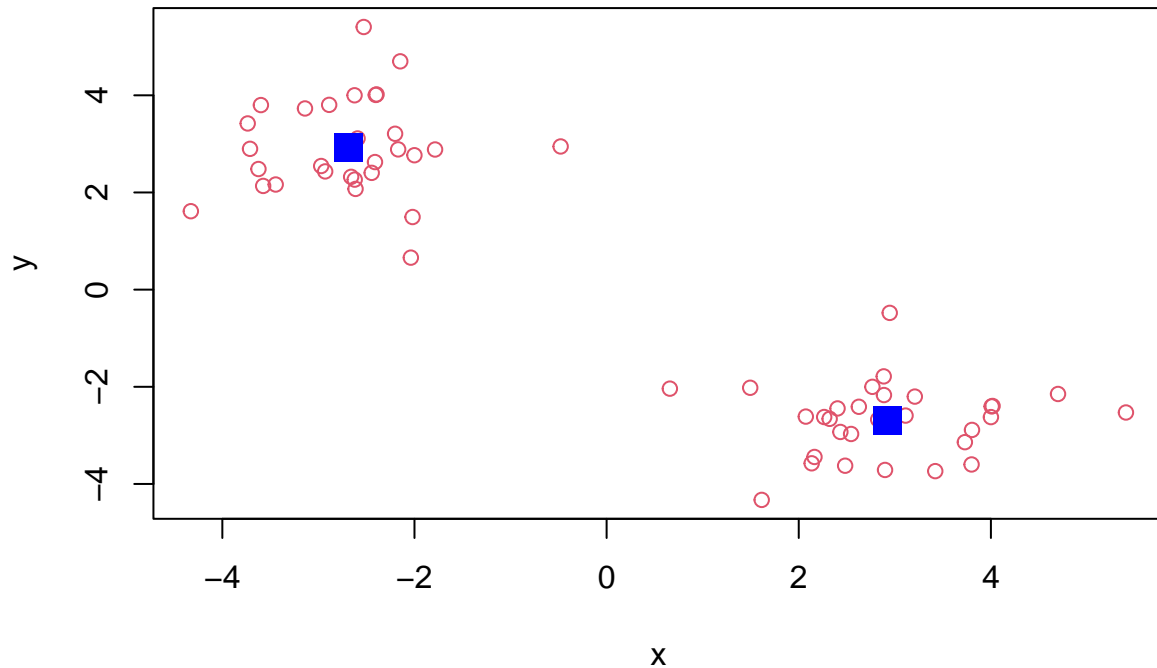
```
tmp <- c(rnorm(30, 3), rnorm(30, -3))  
data <- cbind(x=tmp, y=rev(tmp))  
plot(data)
```



Run `kmeans()` set `k(centers)` to 2 (i.e. the number of clusters we want) `nstart` 20 (number of times we run it). Kmeans requires you to tell it how many clusters you want.



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



## hclust = Hierarchical Clustering

We will use the `hclust()` function on the same data as before and see how this method works.

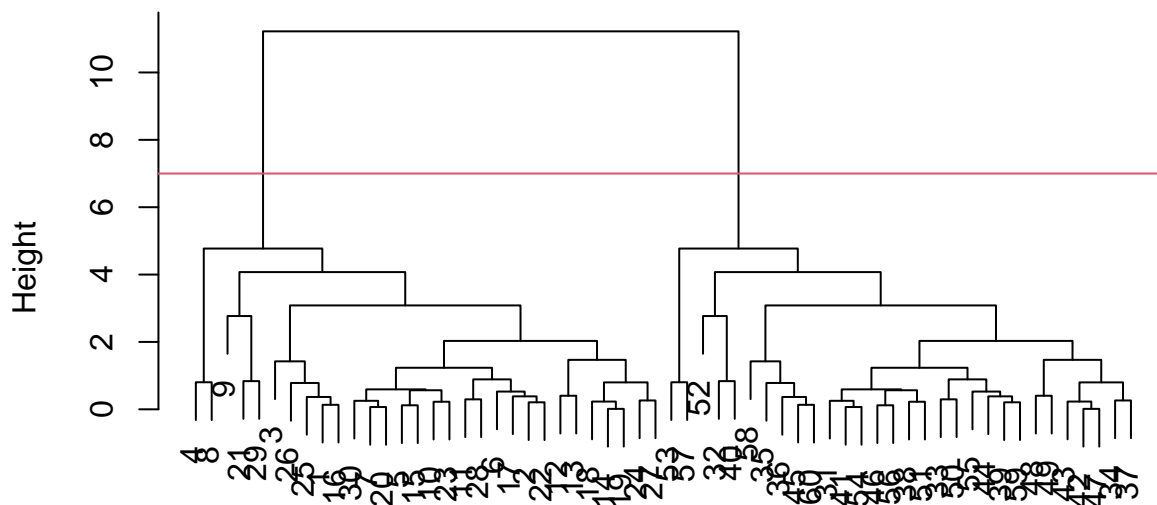
```
hc <- hclust(dist(data))
hc

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

`hclust` also contains a plot method

```
plot(hc)
abline(h=7, col=2)
```

## Cluster Dendrogram



```
dist(data)
hclust (*, "complete")
```

To find our membership vector, we need to “cut” the tree and for this, we use the `cutree()` function. We will tell it the height to cut at.

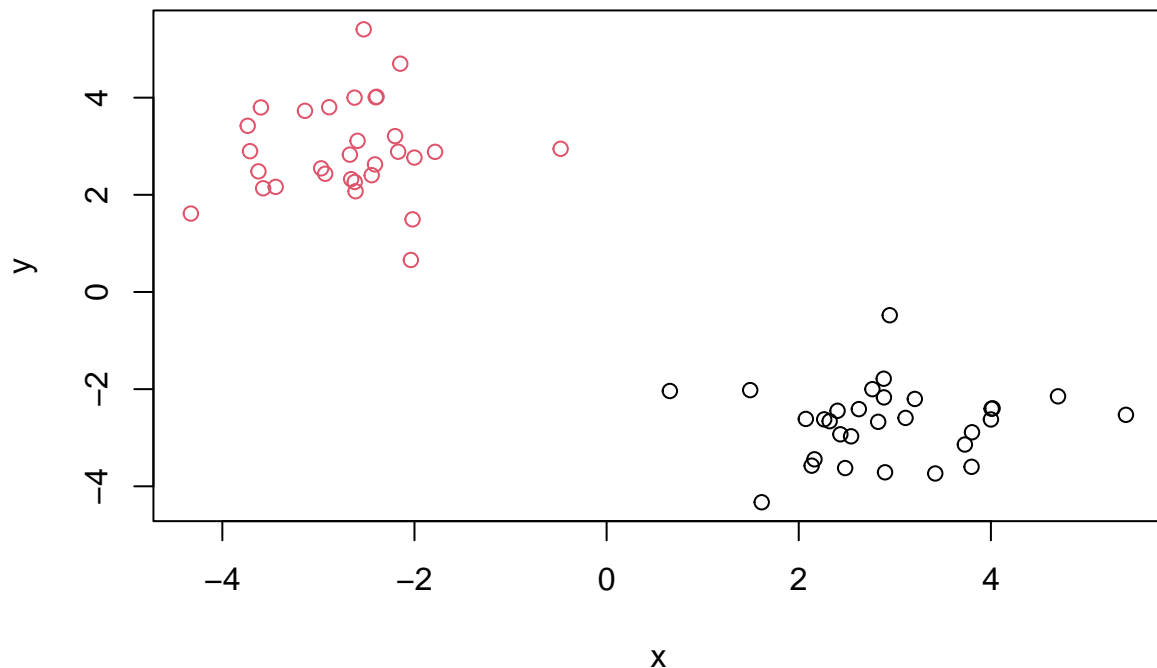
```
cutree(hc, h=7)
```

[illegible]

We can also use `cutree()` and state the number of clusters we want.

```
grps <- cutree(hc, k = 2)
```

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



To recap, we have learned how to use `kmeans()` which is written as `kmeans(c, centers())` and `hclust()` which is written as `hclust(dist(x))` and is more flexible than `kmeans()` and it also shows us something about the nature of our data.

## Principle Component Analysis (PCA)

Starting in the least complex way, we begin with looking at an x and y axis. The PC1 follows a “line of best fit” through the data points. The PC2 describes the variants that are left. These are coordinates that describe the spread of the data.

Objectives of PCA include reducing dimensionality, to visualize multidimensional data, select the most useful variables/features, identify groupings of objects, and to identify outliers.

### PCA of UK food data - lab activity

Import data from a CSV file.

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

How many rows and columns?

```
dim(x)
```

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

```
x[, -1]
```

```
##      England Wales Scotland N.Ireland
## 1      105    103      103        66
## 2      245    227      242        267
## 3      685    803      750        586
## 4      147    160      122         93
## 5      193    235      184        209
## 6      156    175      147        139
## 7      720    874      566       1033
## 8      253    265      171        143
## 9      488    570      418        355
## 10     198    203      220        187
## 11     360    365      337        334
## 12    1102   1137      957        674
## 13    1472   1582     1462       1494
## 14      57     73       53         47
## 15    1374   1256     1572       1506
## 16     375    475      458        135
## 17      54     64       62         41
```

```
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
## 7      Fresh_potatoes 720    874      566       1033
## 8      Fresh_Veg    253    265      171        143
## 9      Other_Veg    488    570      418        355
## 10 Processed_potatoes 198    203      220        187
## 11 Processed_Veg    360    365      337        334
## 12      Fresh_fruit 1102   1137      957        674
## 13      Cereals     1472   1582     1462       1494
## 14      Beverages    57     73       53         47
## 15      Soft_drinks  1374   1256     1572       1506
## 16      Alcoholic_drinks 375    475      458        135
## 17      Confectionery 54     64       62         41
```

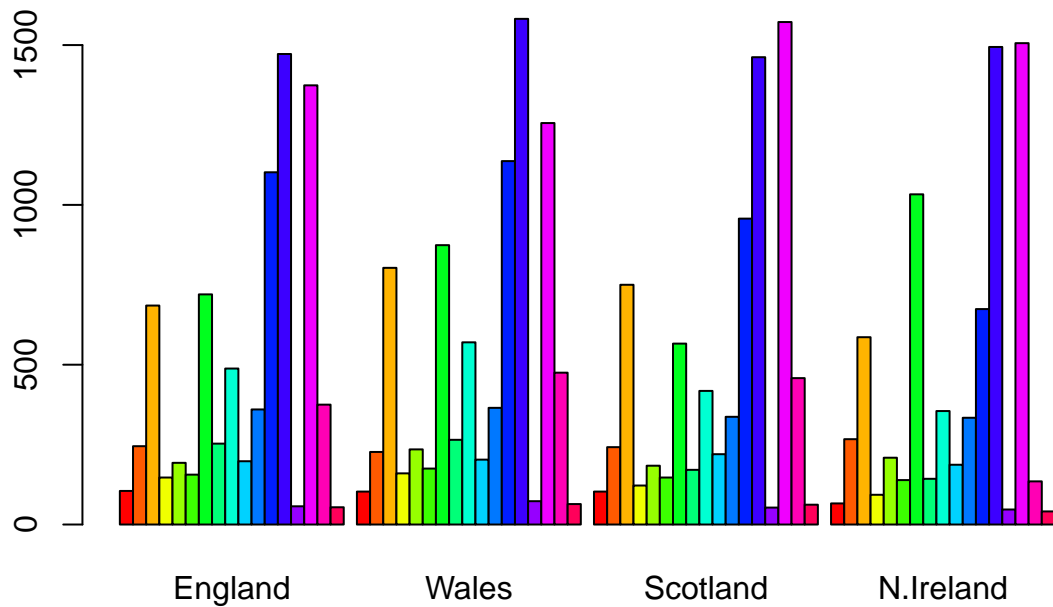
```
rownames(x) <- x[, 1]
```

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

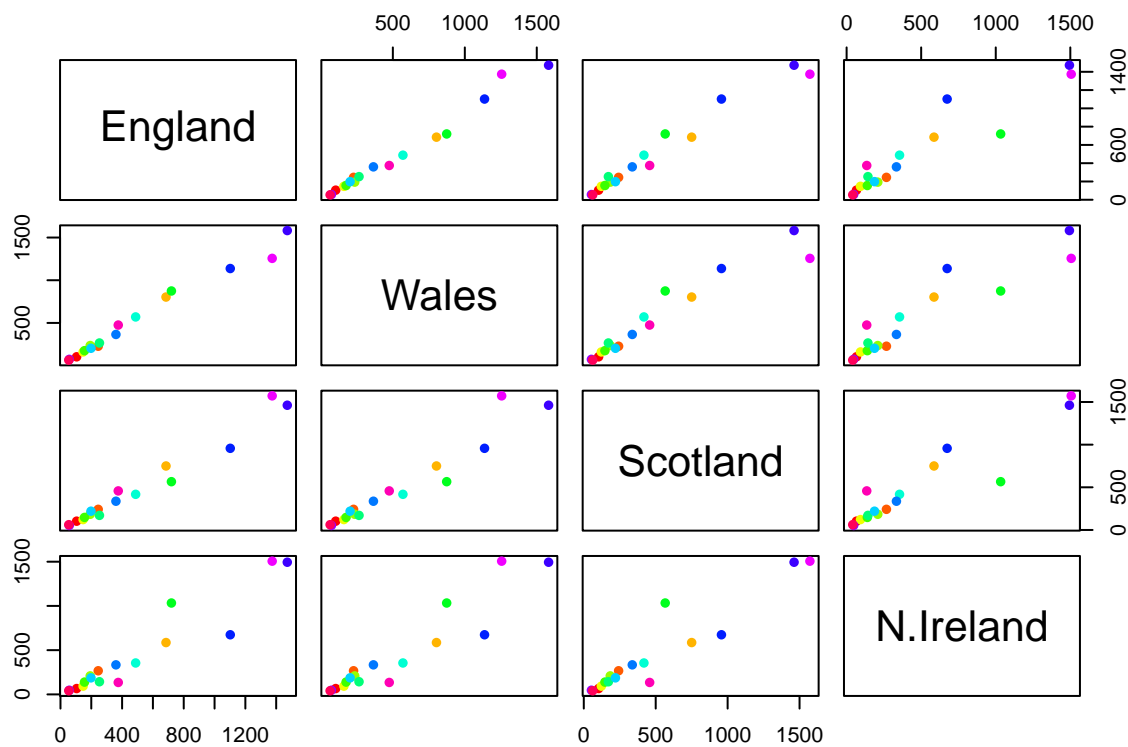
```
##      England Wales Scotland N.Ireland
## Cheese      105    103      103        66
```

## 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

```
barplot(as.matrix(x), col=rainbow(17), beside = TRUE)
```



```
mycols <- rainbow(nrow(x))
pairs(x, col=mycols, pch=16)
```



Now, we will use PCA to become more informed about our data.

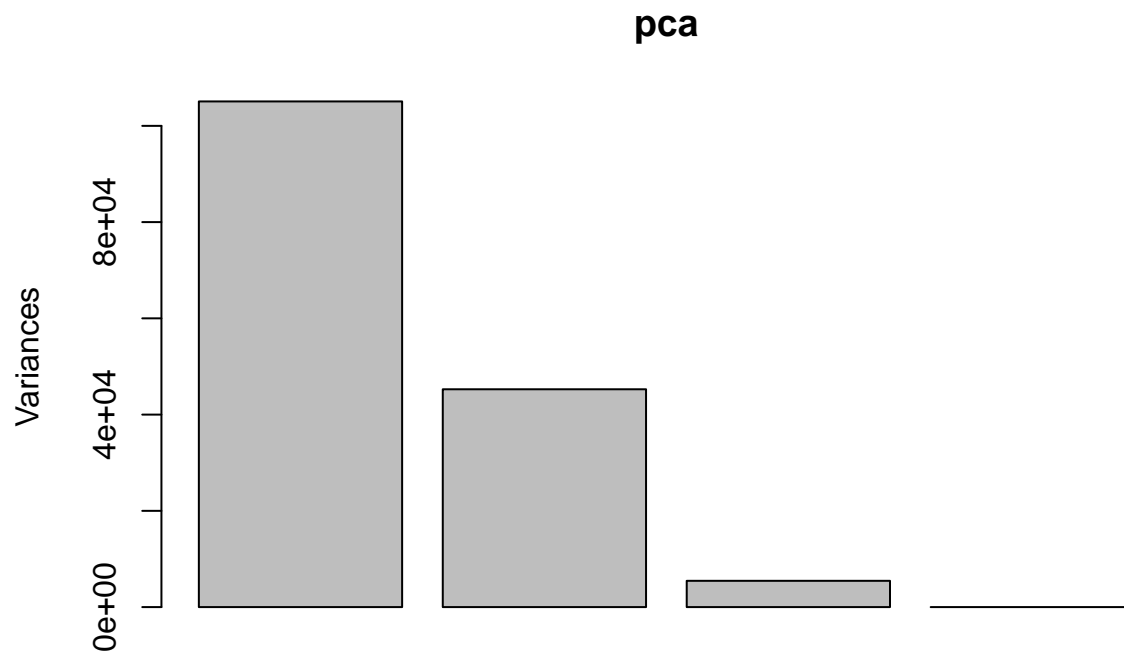
Here, we will use the base R function for PCA, which is called `prcomp()`. This function wants the transpose of data.

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

```
## Importance of components:
##
## Standard deviation      324.1502 212.7478 73.87622 5.552e-14
## Proportion of Variance  0.6744  0.2905  0.03503 0.000e+00
## Cumulative Proportion  0.6744  0.9650  1.00000 1.000e+00
```

```
plot(pca)
```





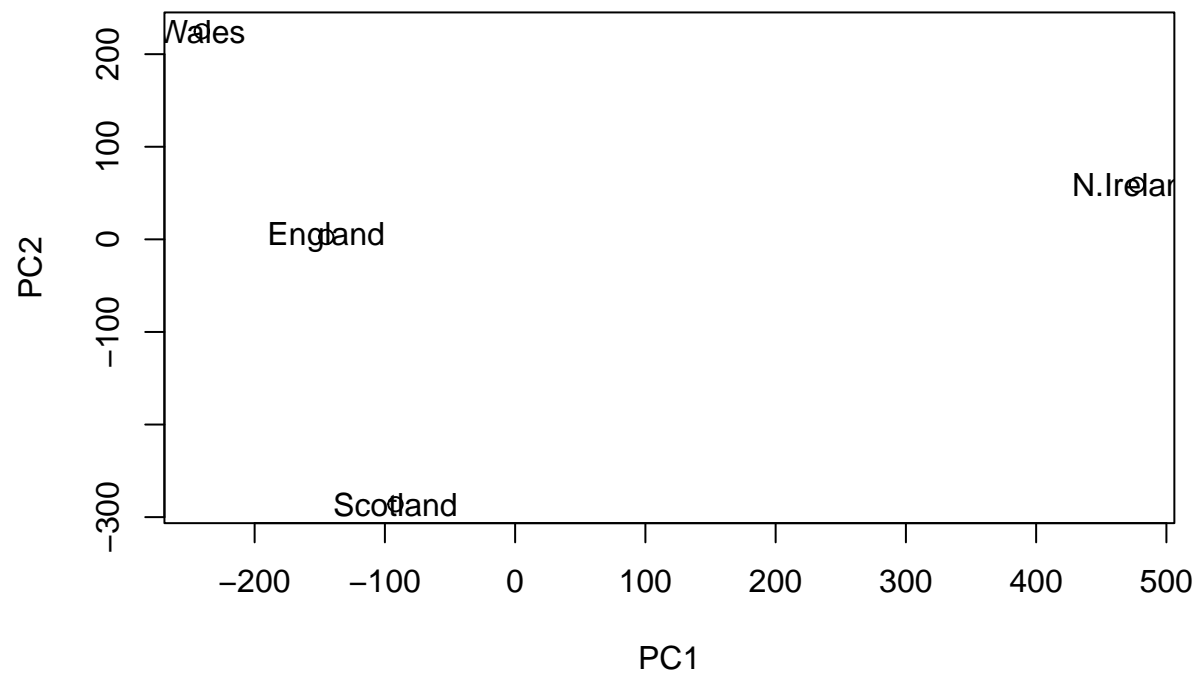
The plot we want is a score plot (a.k.a. PCA plot). This is a plot of PC1 vs. PC2.

```
attributes(pca)
```

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

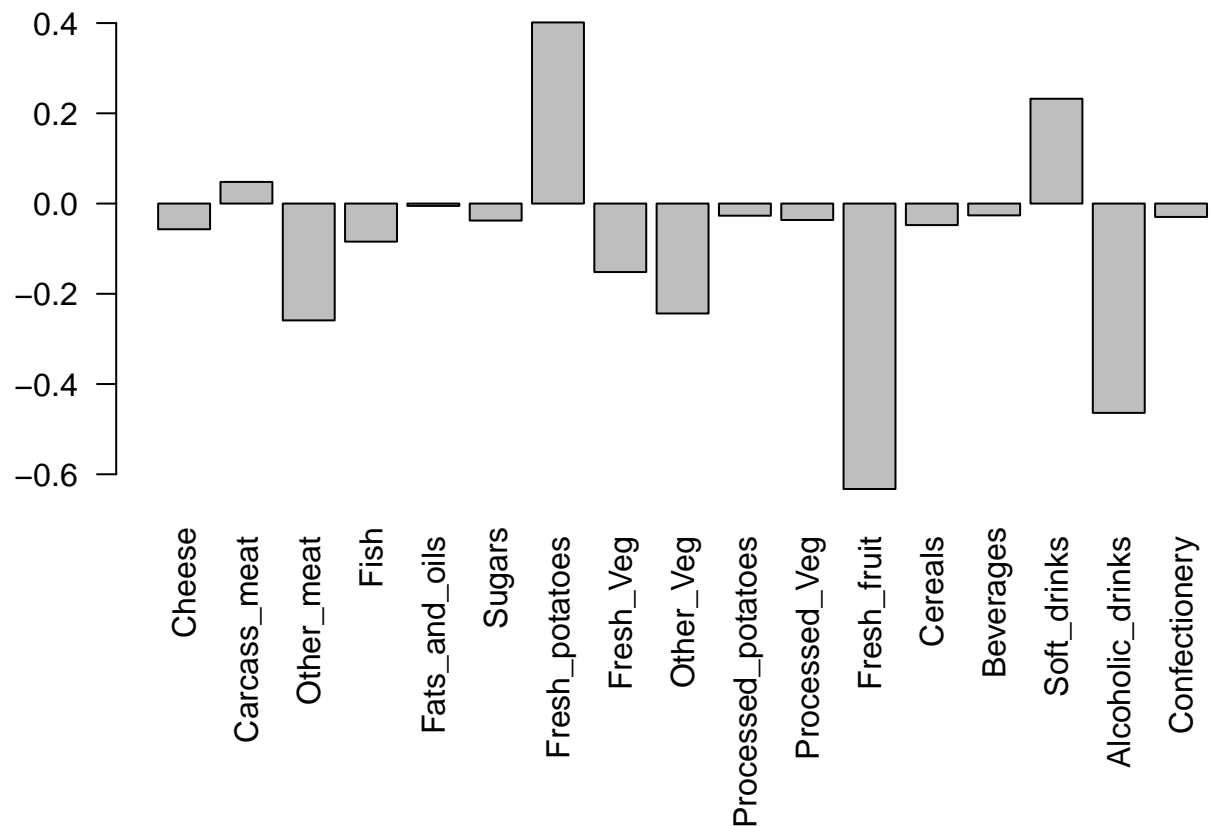
We are after the `pca$x` component for this plot.

```
plot(pca$x[,1:2])
text(pca$x[,1:2], labels=colnames(x))
```



We can also examine the PCA “loadings”, which tell us how much the original variables contribute to each new PC...

```
par(mar=c(10, 3, 0.35, 0))
barplot(pca$rotation[,1], las=2)
```



One more PCA for the day...

```
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
```

```
nrow(rna.data)
```

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

```
ncol(rna.data)
```

```
## [1] 10
```

```
colnames(rna.data)
```

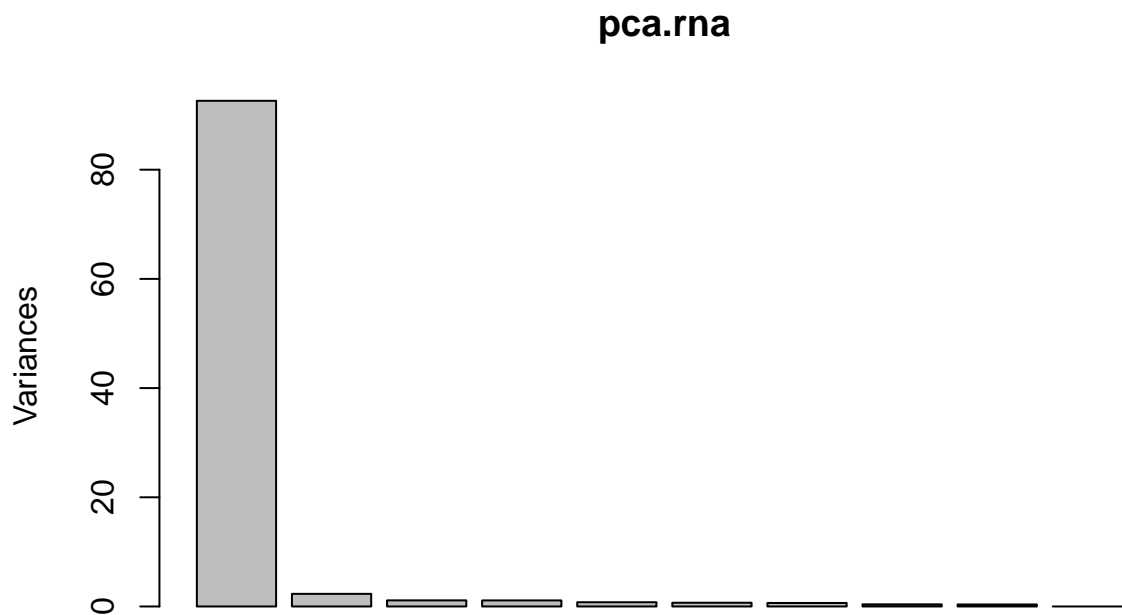
```
## [1] "wt1" "wt2" "wt3" "wt4" "wt5" "ko1" "ko2" "ko3" "ko4" "ko5"
```

```
pca.rna <- prcomp(t(rna.data), scale=TRUE)  
summary(pca.rna)
```

```
## 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 3.327e-15  
## Proportion of Variance 0.00385 0.00364 0.000e+00  
## Cumulative Proportion 0.99636 1.00000 1.000e+00
```

```
plot(pca.rna)
```



```
plot(pca.rna$x[,1:2])  
text(pca.rna$x[,1:2], labels = colnames(rna.data))
```

