class 07

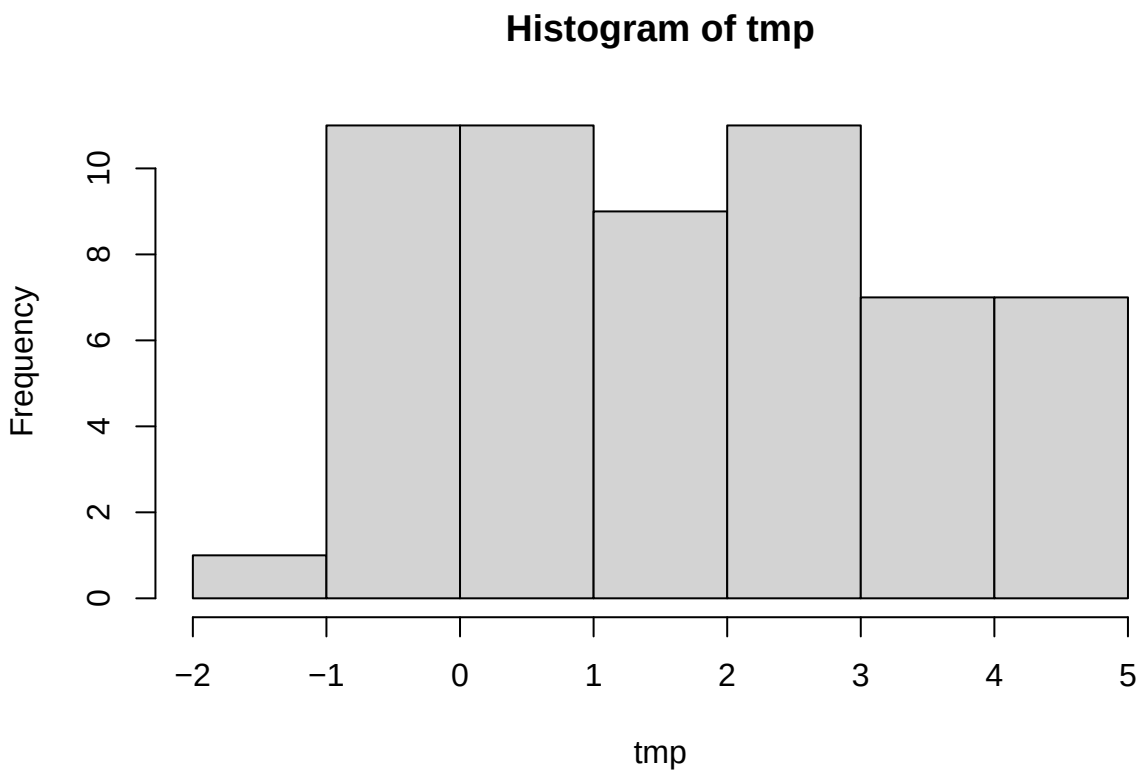
Phoebe LI

2/13/2022

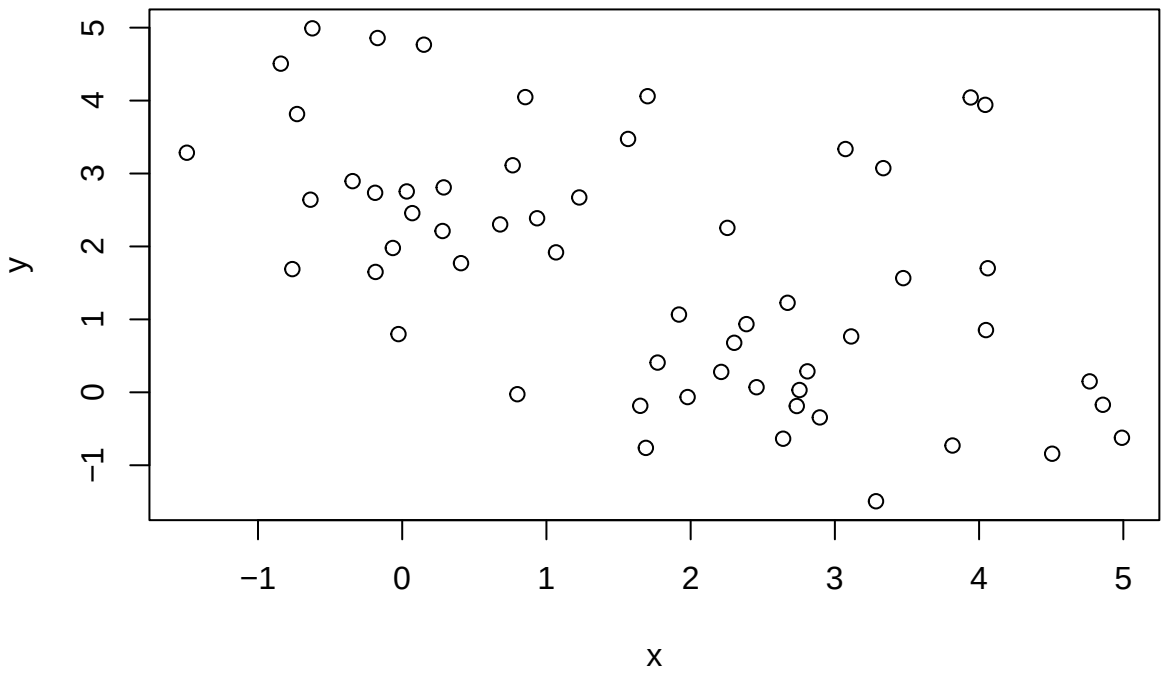
Clusting with kmeans() and hclust()

we will begin by making up some data to cluster.

```
tmp<-c(rnorm(30,3), rnorm(30.-3))
hist(tmp)
```




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




```
x
```

```
##                x          y
##  [1,]  2.64105028 -0.63620027
##  [2,]  4.76650578  0.15024511
##  [3,]  4.05996086  1.70141291
##  [4,]  3.11297388  0.76612795
##  [5,]  4.85797792 -0.17072594
##  [6,]  2.89556389 -0.34407147
##  [7,]  2.45690466  0.06986365
##  [8,]  3.47403883  1.56589116
##  [9,]  4.04755812  0.85341045
## [10,]  3.28517592 -1.49330586
## [11,]  4.50665940 -0.84186748
## [12,]  1.68946342 -0.76178931
## [13,]  1.91889806  1.06646894
## [14,]  2.21190162  0.27968726
## [15,]  2.38678311  0.93516617
## [16,]  4.99049840 -0.62284646
## [17,]  1.65061307 -0.18492497
## [18,]  0.79832798 -0.02593816
## [19,]  3.33574581  3.07354305
## [20,]  2.75459324  0.03146930
## [21,]  3.81550844 -0.72929278
## [22,]  2.30208891  0.67908137
## [23,]  1.77083271  0.40768802
```

```
## [24,]  2.80951780  0.28764241
## [25,]  2.73581713 -0.18790016
## [26,]  1.97886538 -0.06489263
## [27,]  2.67194961  1.22765907
## [28,]  3.94169053  4.04289014
## [29,]  2.25464791  2.25464791
## [30,]  4.04289014  3.94169053
## [31,]  1.22765907  2.67194961
## [32,] -0.06489263  1.97886538
## [33,] -0.18790016  2.73581713
## [34,]  0.28764241  2.80951780
## [35,]  0.40768802  1.77083271
## [36,]  0.67908137  2.30208891
## [37,] -0.72929278  3.81550844
## [38,]  0.03146930  2.75459324
## [39,]  3.07354305  3.33574581
## [40,] -0.02593816  0.79832798
## [41,] -0.18492497  1.65061307
## [42,] -0.62284646  4.99049840
## [43,]  0.93516617  2.38678311
## [44,]  0.27968726  2.21190162
## [45,]  1.06646894  1.91889806
## [46,] -0.76178931  1.68946342
## [47,] -0.84186748  4.50665940
## [48,] -1.49330586  3.28517592
## [49,]  0.85341045  4.04755812
## [50,]  1.56589116  3.47403883
## [51,]  0.06986365  2.45690466
## [52,] -0.34407147  2.89556389
## [53,] -0.17072594  4.85797792
## [54,]  0.76612795  3.11297388
## [55,]  1.70141291  4.05996086
## [56,]  0.15024511  4.76650578
## [57,] -0.63620027  2.64105028
```

Run Kmeans()

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

```
## K-means clustering with 2 clusters of sizes 26, 31
##
## Cluster means:
##           x        y
## 1 2.9457703 0.152233
## 2 0.6647283 3.007695
##
## Clustering vector:
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 2 2 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] 45.69476 92.86144
##  (between_SS / total_SS =  57.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Cluster membership Q. what size is each cluster

```
K$size
```

```
## [1] 26 31
```

Q. cluster centers

```
K$centers
```

```
##           x        y
## 1 2.9457703 0.152233
## 2 0.6647283 3.007695
```

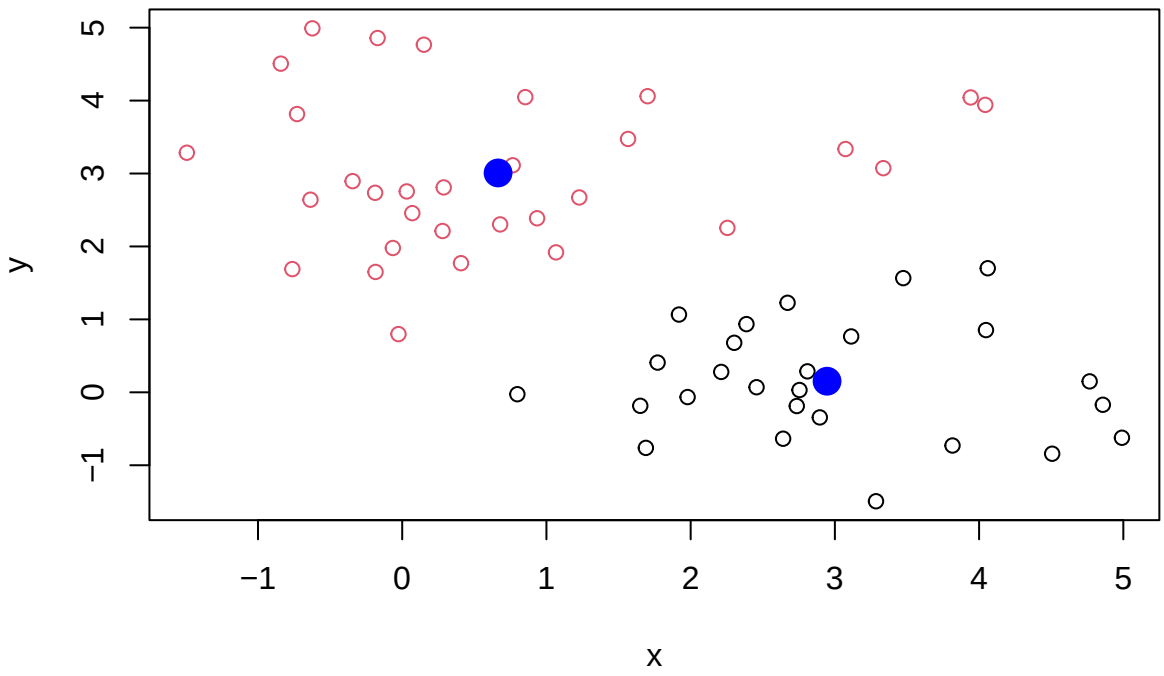
Q. membership vector

```
K$cluster
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 2 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
```

plot our data with the clusting result

```
plot(x, col=K$cluster)
points(K$centers, col="blue", pch=16,cex=2)
```



hclust()

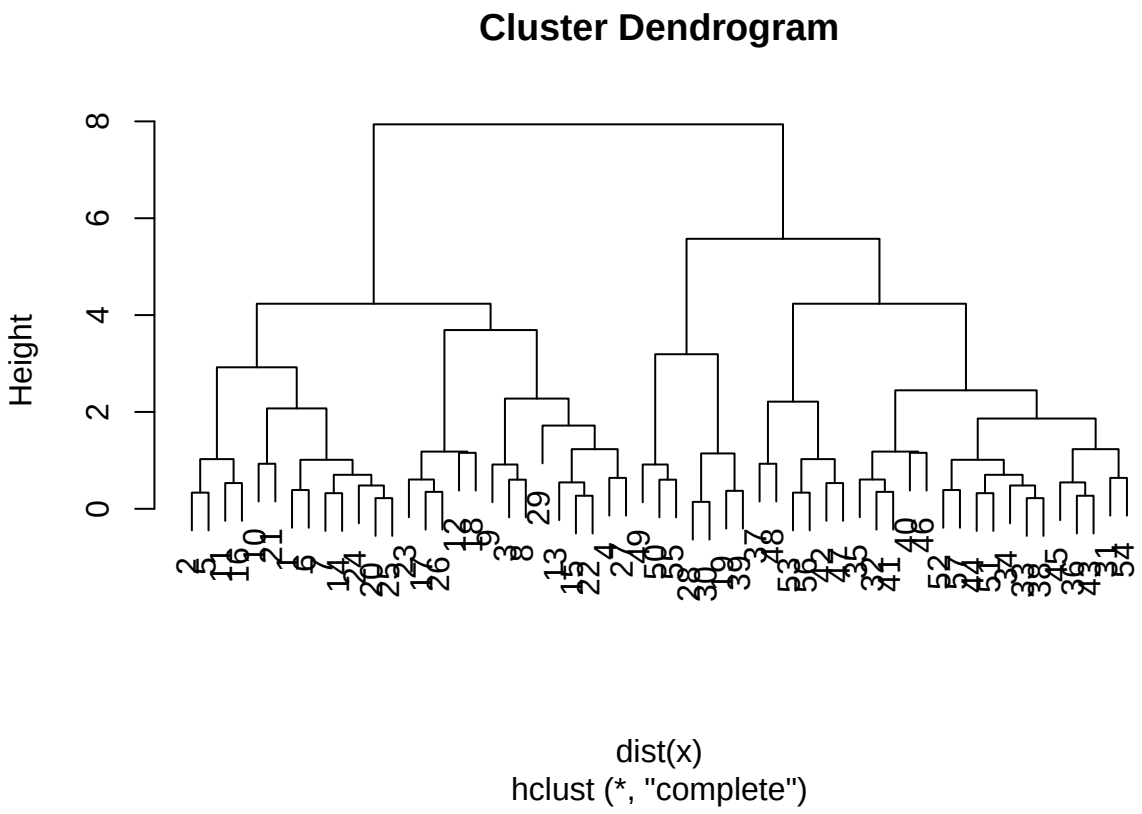
Hierarchical Clustering

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

```
##
## Call:
## hclust(d = dist(x))
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 57
```

There is a cool and useful plot method for hcluster()

```
plot(hc)
```



1. PCA of UK food data

Data import

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

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

```
## Complete the following code to find out how many rows and columns are in x?
dim(x)
```

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

```
nrow(x)
```

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

```
nrow(x)
```

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

```
 ## Preview the first 6 rows
head(x,6)
```

```
##                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
```

```
# Note how the minus indexing works

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

```
##   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
```

```
dim(x)
```

```
## [1] 17  4
```

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

```
dim(x)
```

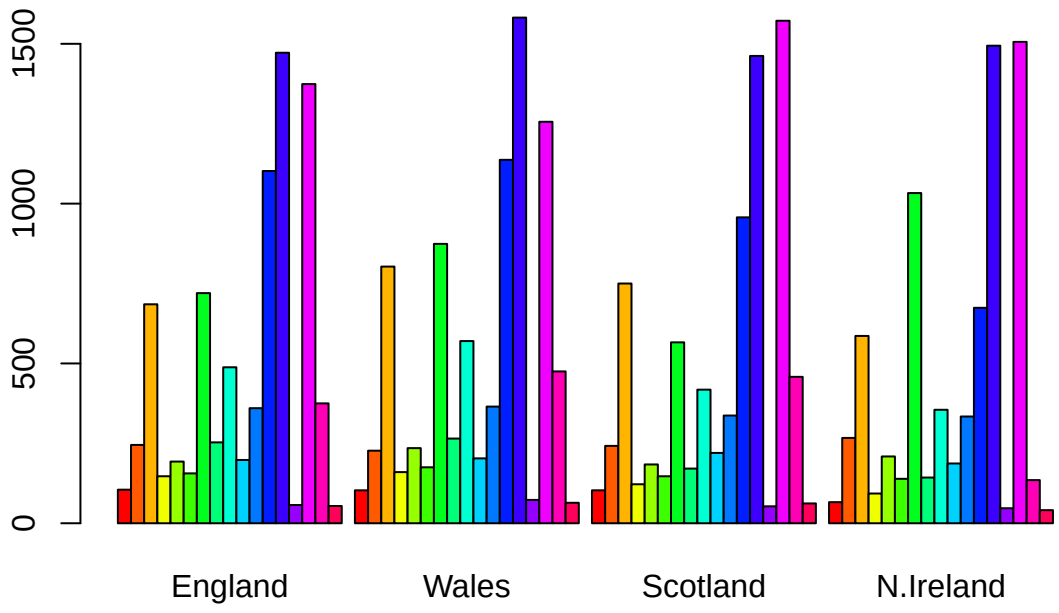
```
## [1] 17  4
```

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?

I prefer the second one, becuase it makes it clear the function of the first row is name.

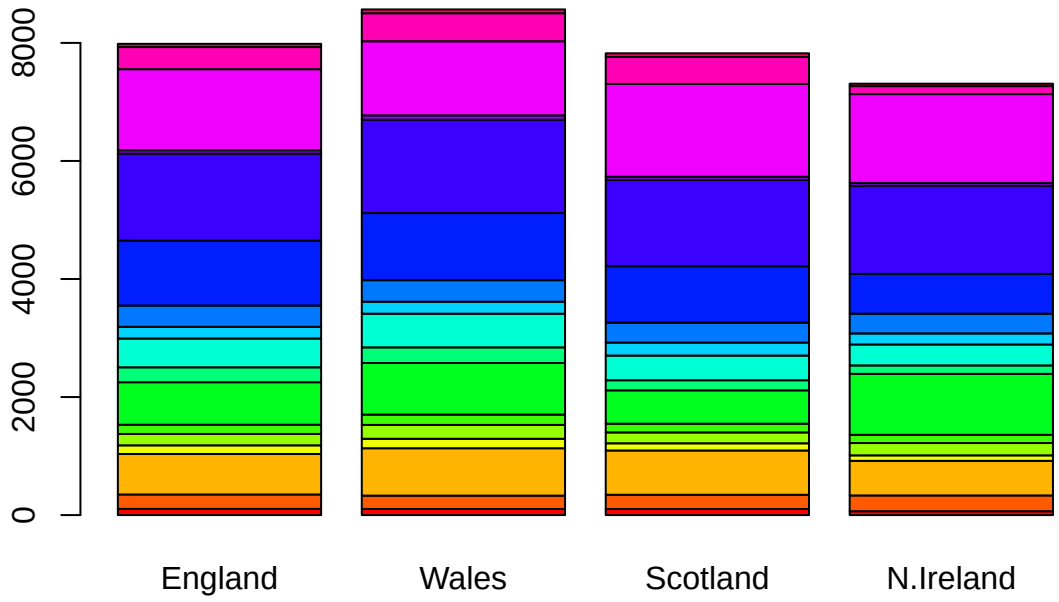
Spotting major differences and trends

```
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?

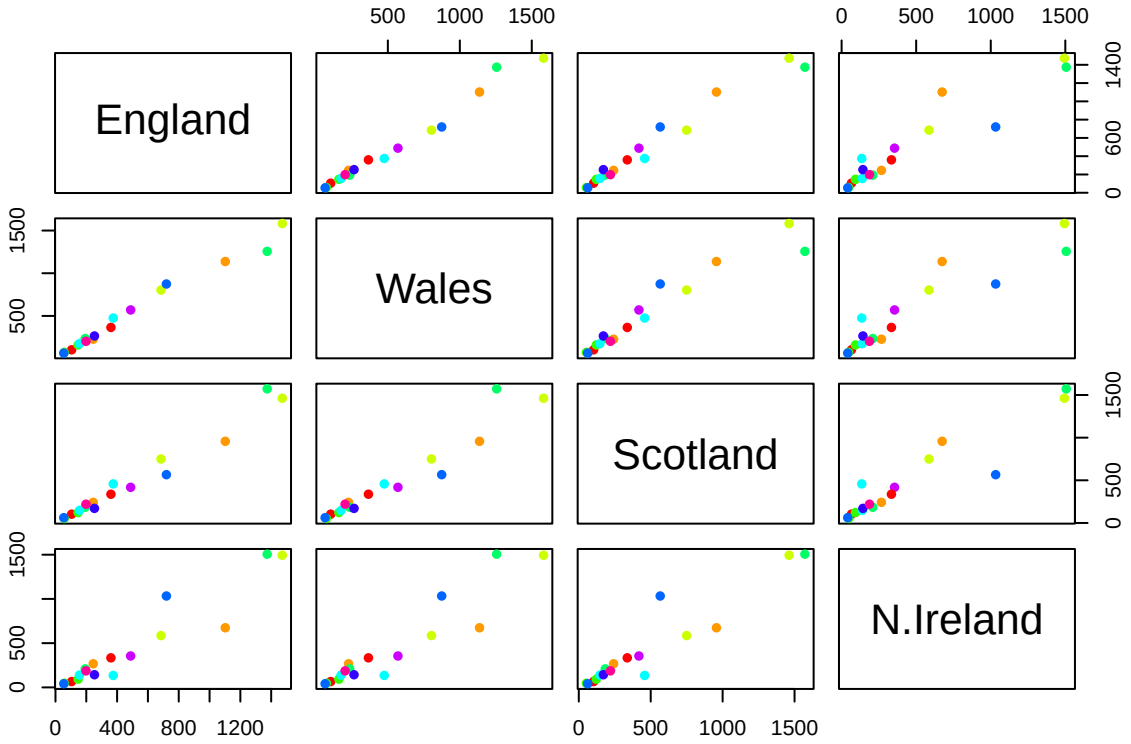
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



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?

The figure means the country you see the name on the vertical vs harizontal.

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



> Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set?

The data is more spread compare to others. They intake higher fresh potato than others. Other countries food consume are similar, but N.Ireland is more different than others.

PCA to the rescue!

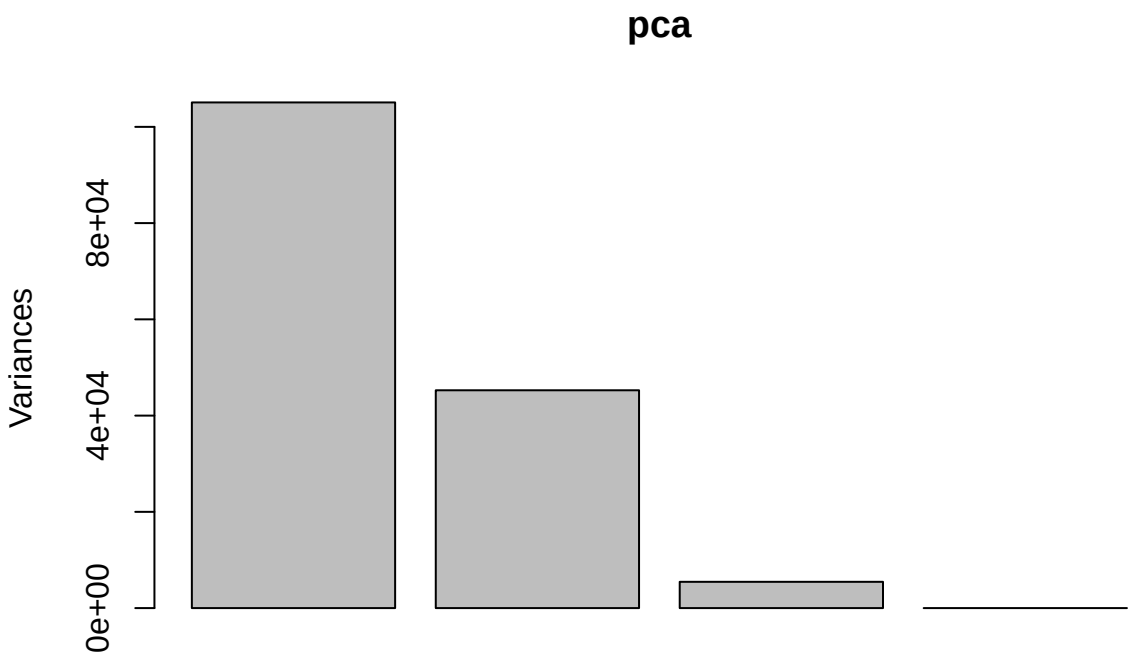
Do PCA of 17D food data. The main function in base R is called prcomp()

```
#this function require tranpom of the data t(x)
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
```

The 'prcomp()' function return a list of object.

```
plot(pca)
```



The "PCA plot" aka a pca score plot is a plot of PC1 vs PC2. Basically using the new PCA axis to view our data.

```
attributes(pca)
```

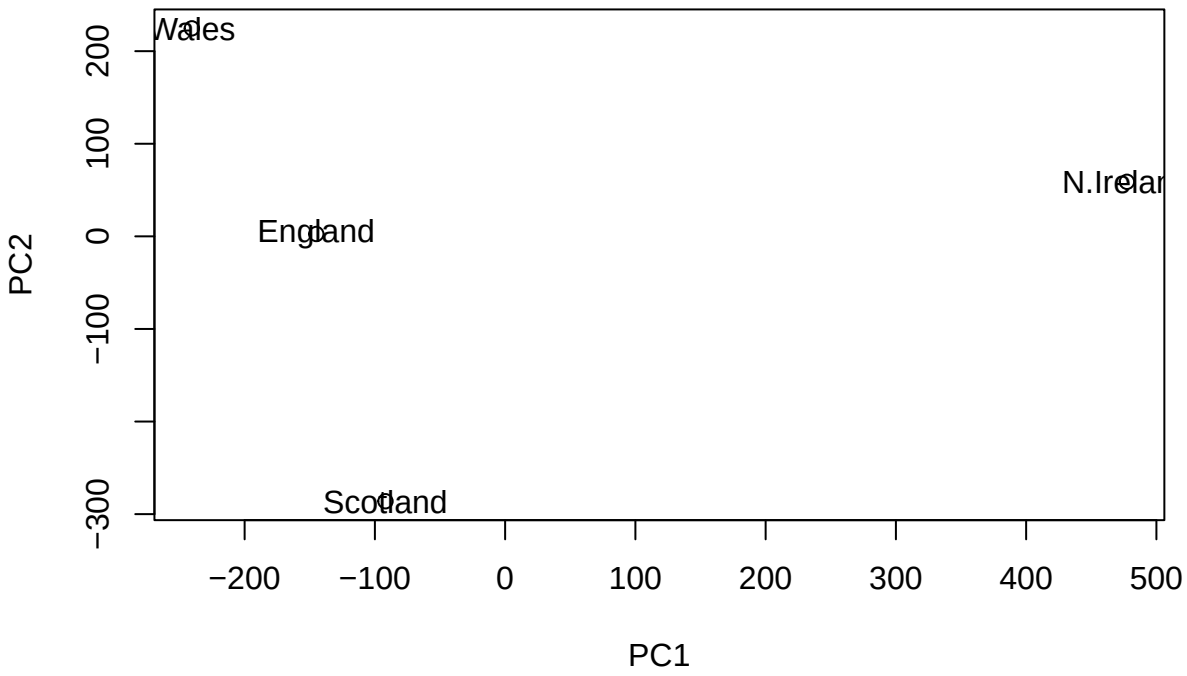
```
## $names
## [1] "sdev"     "rotation" "center"   "scale"    "x"
##
## $class
## [1] "prcomp"
```

```
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
```

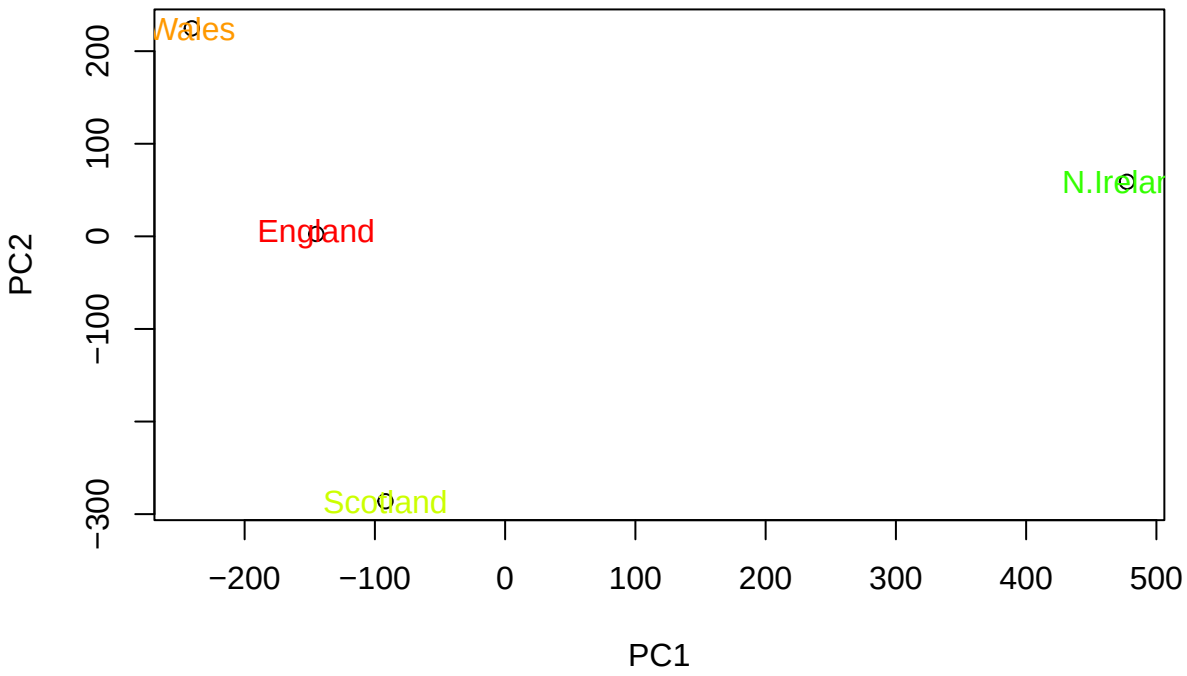
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
plot(pca$x[,1], pca$x[,2],xlab="PC1", ylab="PC2")
text(pca$x[,1], pca$x[,2], labels = colnames(x))
```



>Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
plot(pca$x[,1], pca$x[,2],xlab="PC1", ylab="PC2")
text(pca$x[,1], pca$x[,2], labels = colnames(x), col=rainbow(10))
```



Below we can use the square of pca$sdev , which stands for "standard deviation", to calculate how much variation in the original data each PC accounts for.

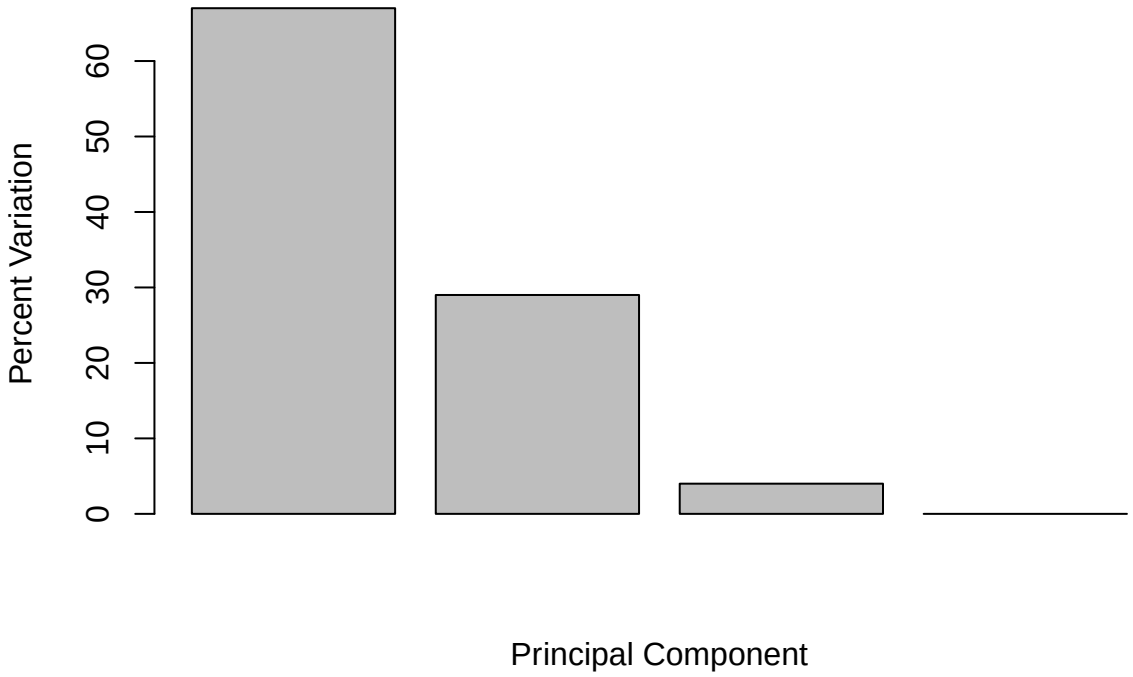
```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

```
## [1] 67 29  4  0
```

```
## or the second row here...
z <- summary(pca)
z$importance
```

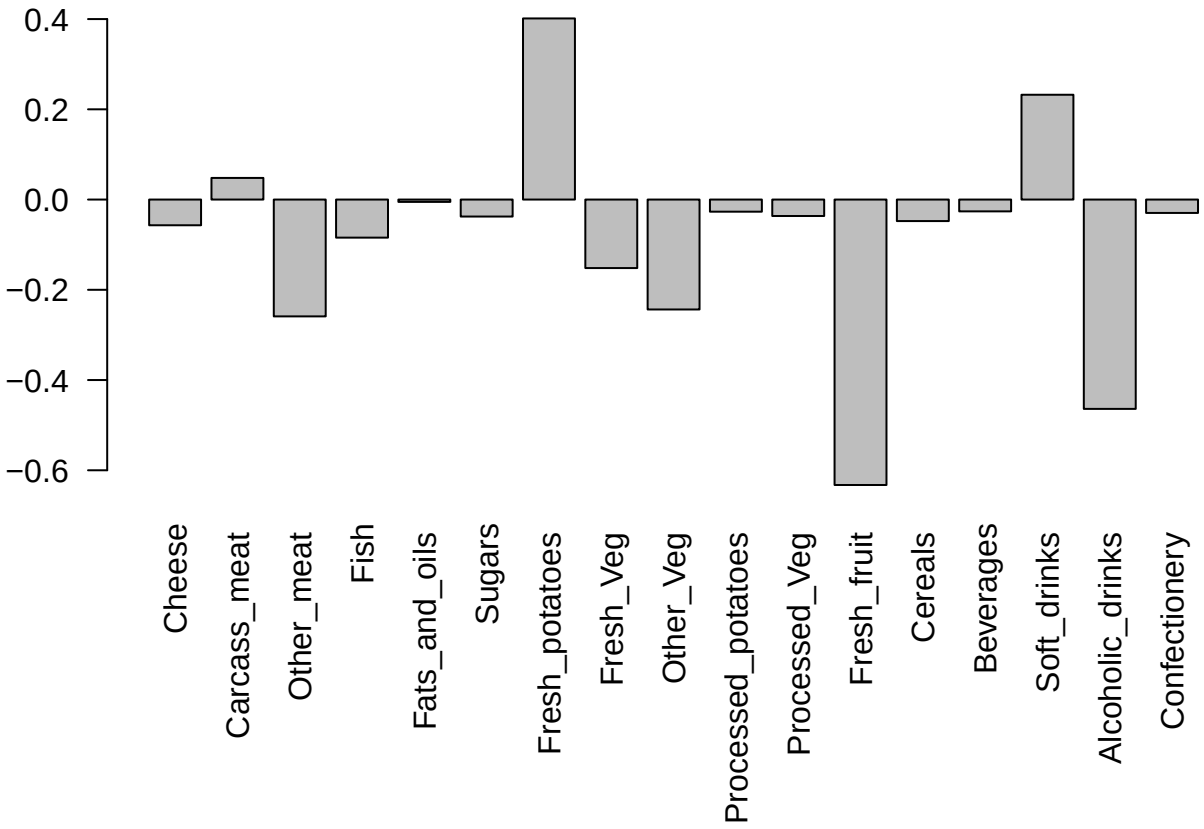
```
##                             PC1       PC2      PC3          PC4
## Standard deviation     324.15019 212.74780 73.87622 4.188568e-14
## Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
## Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```



Digging deeper (variable loadings)

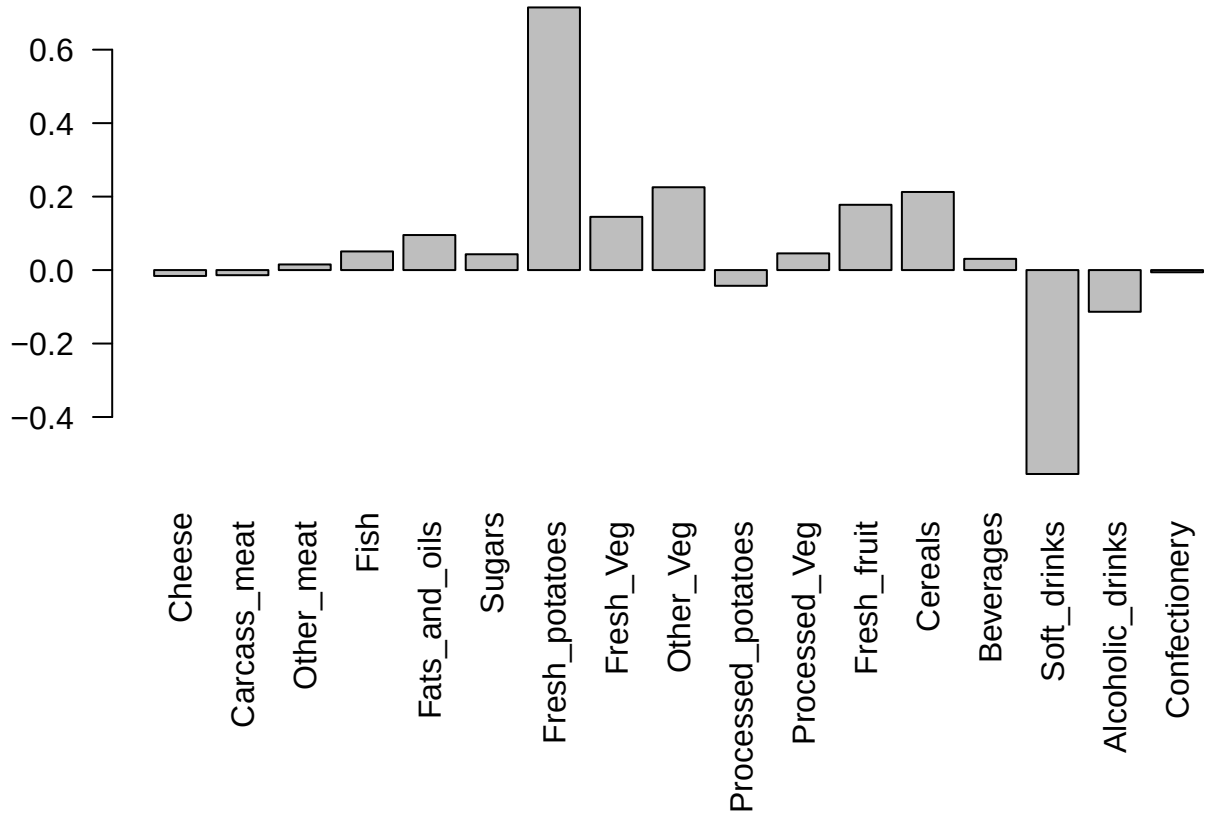
```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



> Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about?

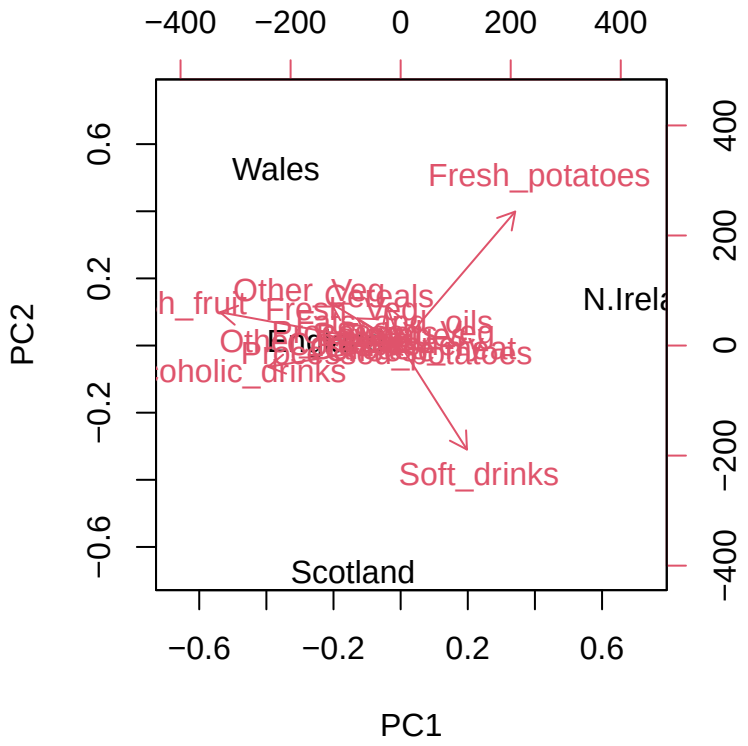
It shows that fresh potato and soft drinks are two food groups feature prominantely.

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



Biplots

```
## The inbuilt biplot() can be useful for small datasets
biplot(pca)
```



PCA of RNA-seq data

```
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
```

Q10: How many genes and samples are in this data set?

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

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

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

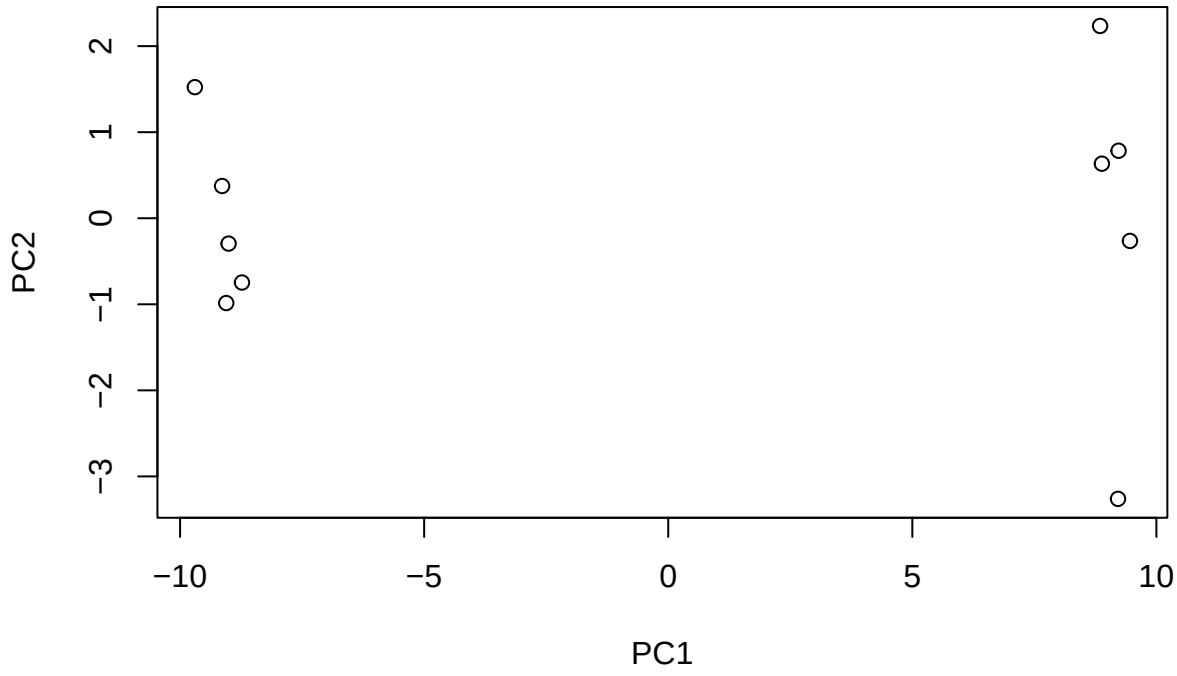
```
## [1] 100
```

```
dim(rna.data)
```

```
## [1] 100  10
```

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

## Simple un polished plot of pc1 and pc2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```



This quick plot looks interesting with a nice separation of samples into two groups of 5 samples each. Before delving into the details of this grouping let's first examine a summary of how much variation in the original data each PC accounts for:

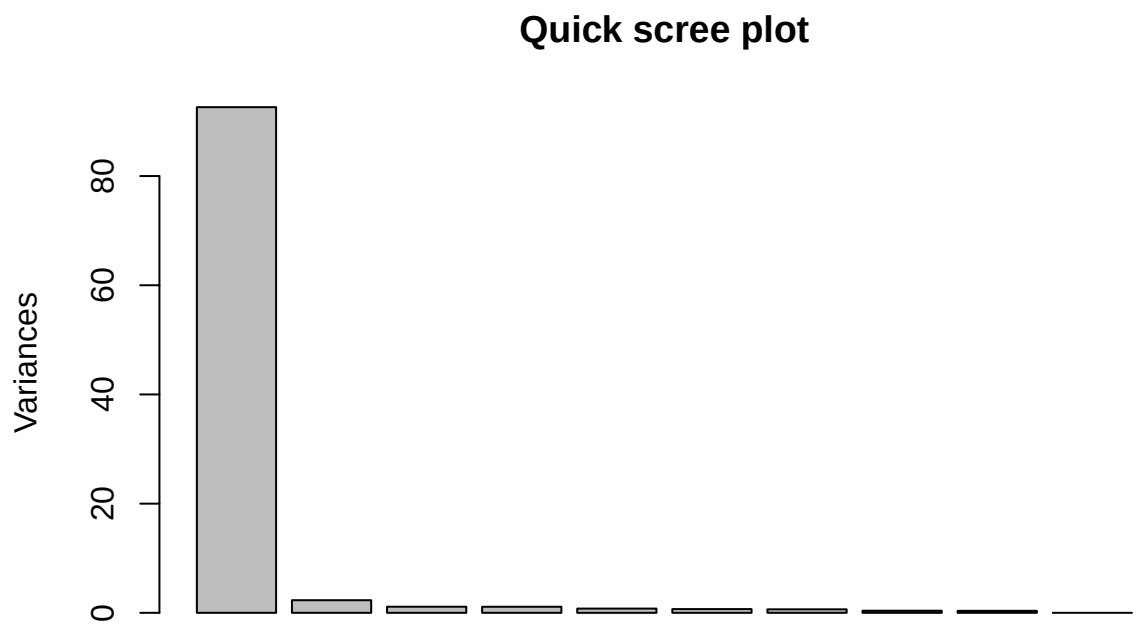
```
summary(pca)
```

```
## 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
```

A quick barplot summary of this Proportion of Variance for each PC can be obtained by calling the plot() function directly on our prcomp result object.

```
plot(pca, main="Quick scree plot")
```



Let's make the above scree plot ourselves and in so doing explore the object returned from prcomp() a little further. We can use the square of pca$sdev, which stands for "standard deviation", to calculate how much variation in the original data each PC accounts for:

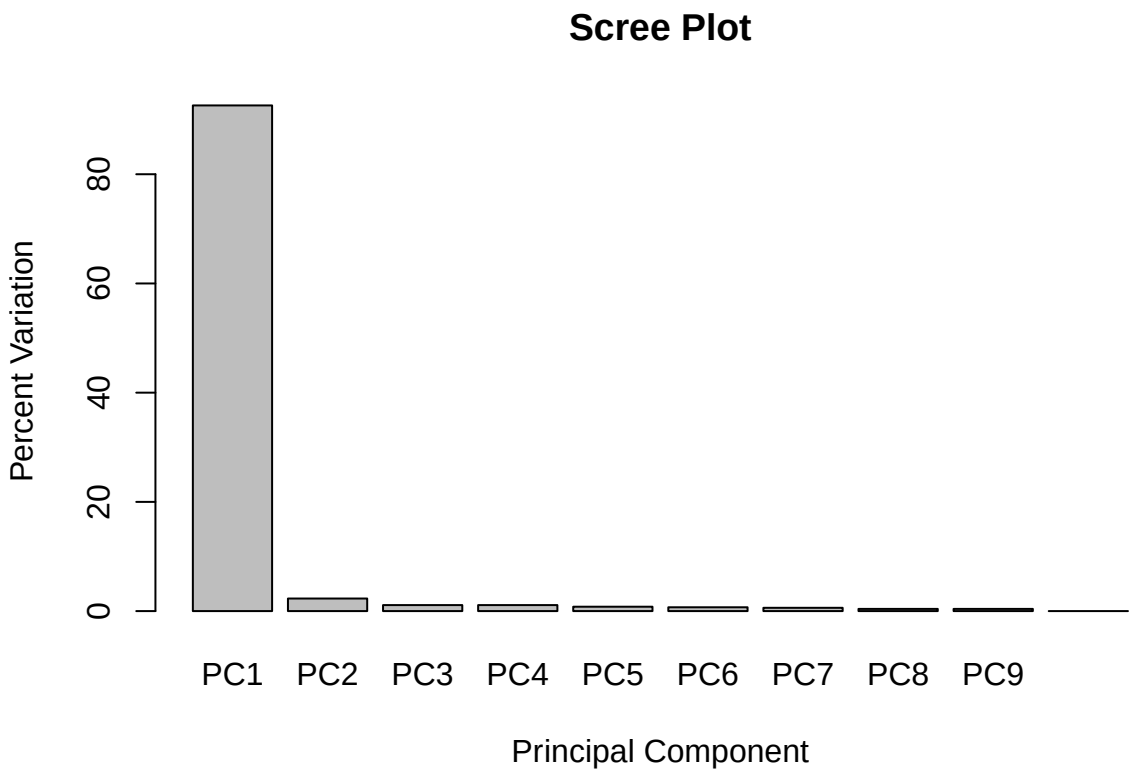
```
## Variance captured per PC
pca.var <- pca$sdev^2

## Percent variance is often more informative to look at
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)
pca.var.per
```

```
##  [1] 92.6  2.3  1.1  1.1  0.8  0.7  0.6  0.4  0.4  0.0
```

We can use this to generate our own scree-plot like this

```
barplot(pca.var.per, main="Scree Plot",
        names.arg = paste0("PC", 1:10),
        xlab="Principal Component", ylab="Percent Variation")
```

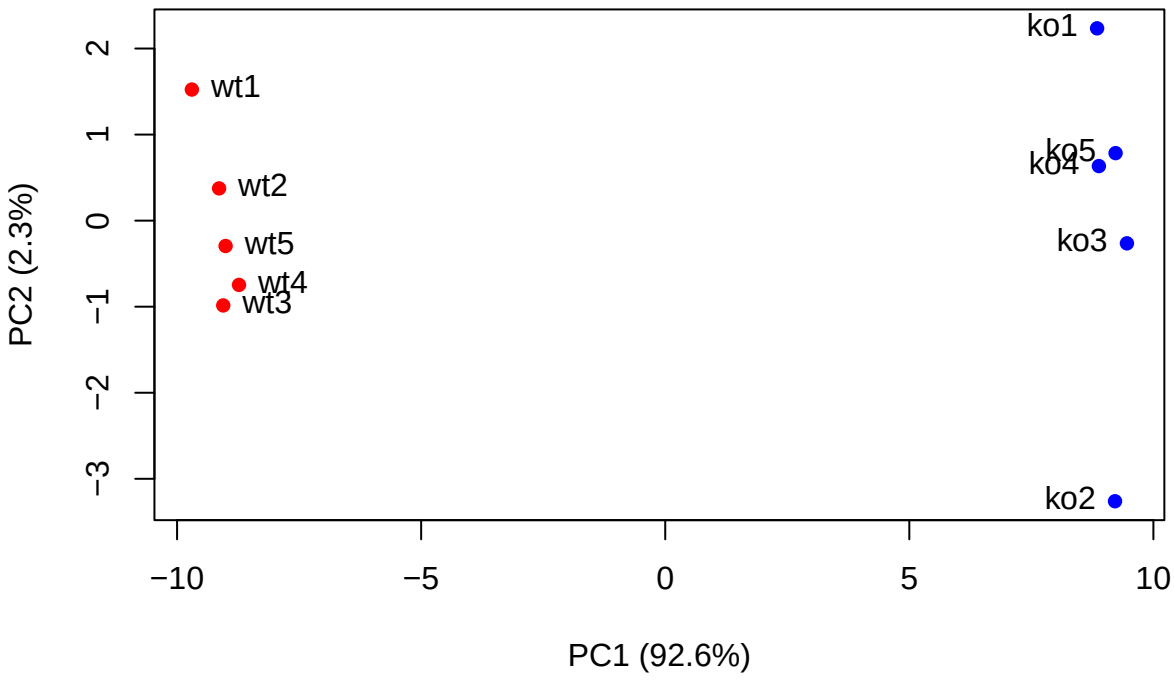


Now lets make our main PCA plot a bit more attractive and useful...

```
## A vector of colors for wt and ko samples
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

plot(pca$x[,1], pca$x[,2], col=colvec, pch=16,
     xlab=paste0("PC1 (", pca.var.per[1], "%)"),
     ylab=paste0("PC2 (", pca.var.per[2], "%)"))

text(pca$x[,1], pca$x[,2], labels = colnames(rna.data), pos=c(rep(4,5), rep(2,5)))
```

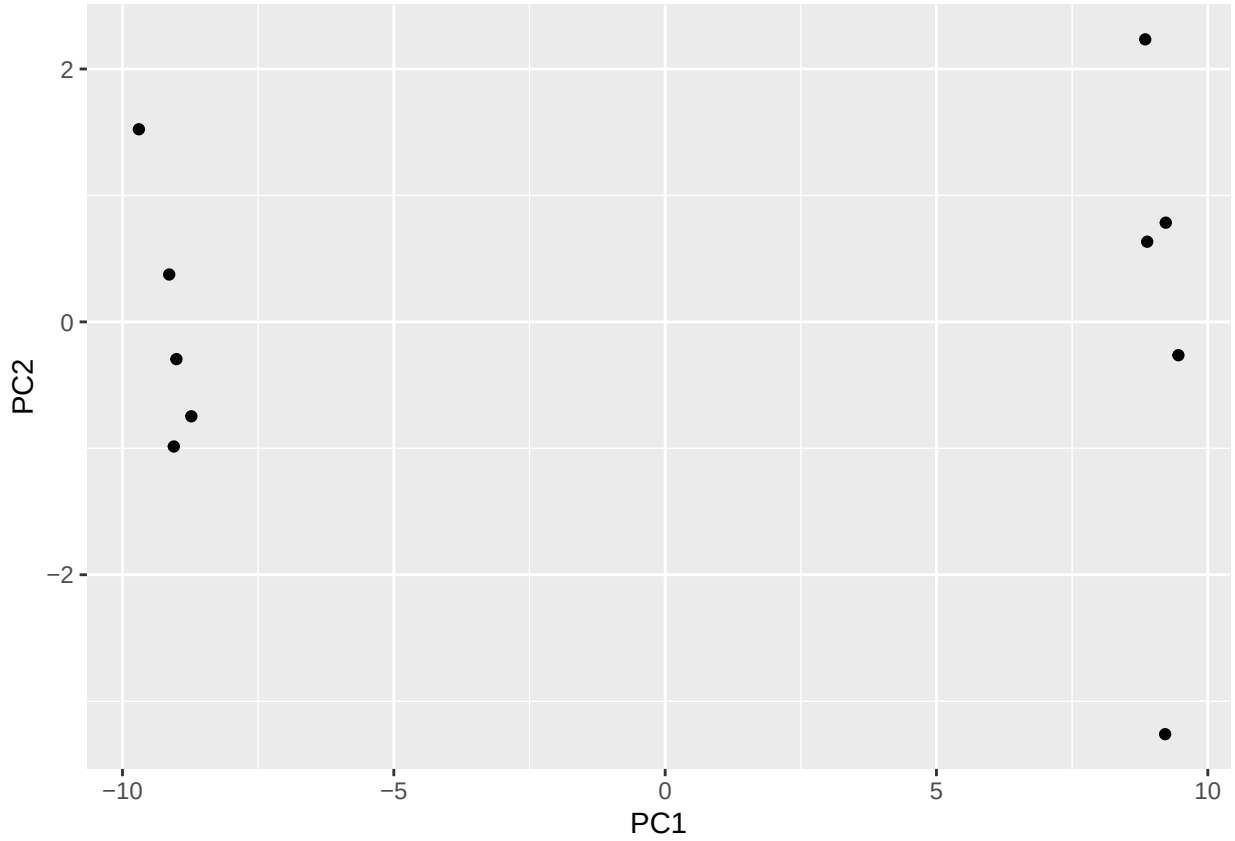


Using ggplot

```
library(ggplot2)

df <- as.data.frame(pca$x)

# Our first basic plot
ggplot(df) +
  aes(PC1, PC2) +
  geom_point()
```



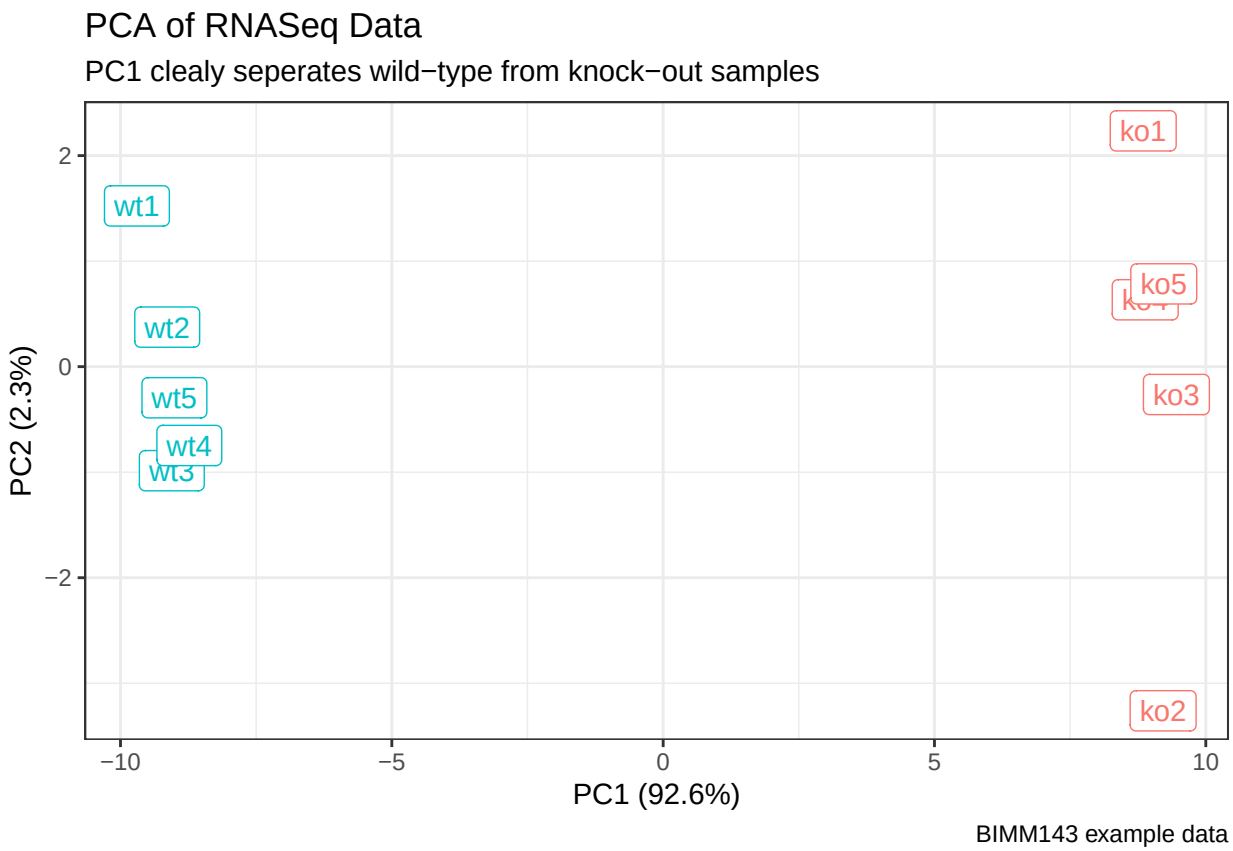
If we want to add a condition specific color and perhaps sample label aesthetics for wild-type and knock-out samples we will need to have this information added to our data.frame:

```
# Add a 'wt' and 'ko' "condition" column
df$samples <- colnames(rna.data)
df$condition <- substr(colnames(rna.data),1,2)

p <- ggplot(df) +
        aes(PC1, PC2, label=samples, col=condition) +
        geom_label(show.legend = FALSE)
```

some polish

```
p + labs(title="PCA of RNASeq Data",
       subtitle = "PC1 clealy seperates wild-type from knock-out samples",
       x=paste0("PC1 (", pca.var.per[1], "%)"),
       y=paste0("PC2 (", pca.var.per[2], "%)"),
caption="BIMM143 example data") +
     theme_bw()
```



Optional: Gene loadings

```
loading_scores <- pca$rotation[,1]

## Find the top 10 measurements (genes) that contribute
## most to PC1 in either direction (+ or -)
gene_scores <- abs(loading_scores)
gene_score_ranked <- sort(gene_scores, decreasing=TRUE)

## show the names of the top 10 genes
top_10_genes <- names(gene_score_ranked[1:10])
top_10_genes
```

```
##  [1] "gene100" "gene66"  "gene45"  "gene68"  "gene98"  "gene60"  "gene21"
##  [8] "gene56"  "gene10"  "gene90"
```