Machine Learning 1

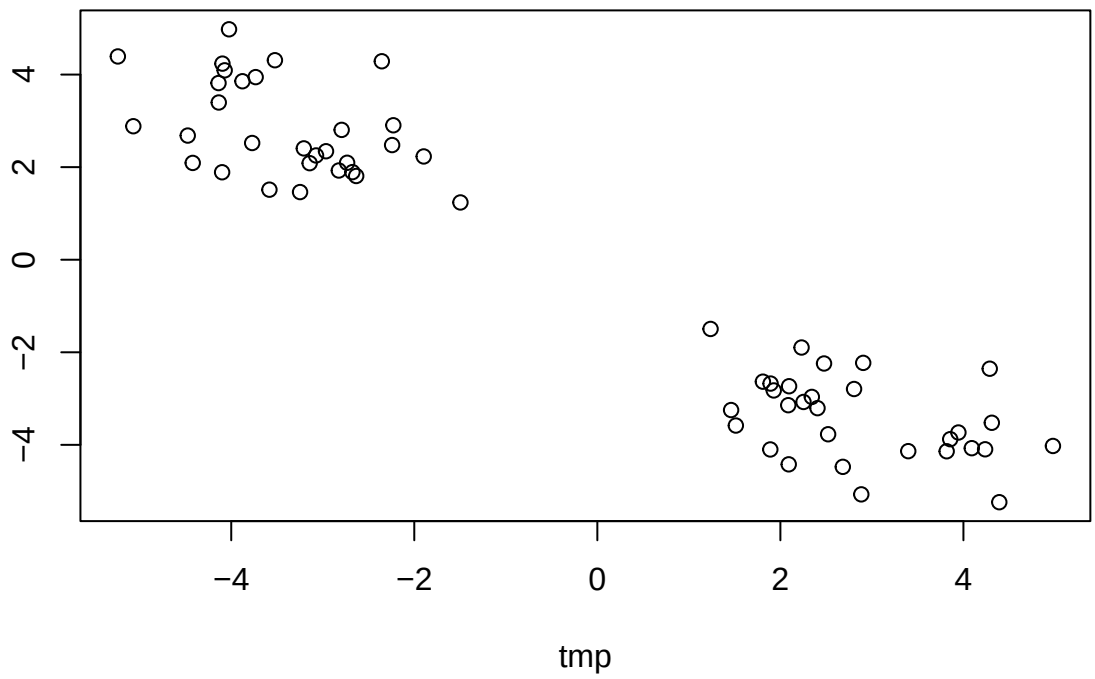
Chantal Rabay A1452864

2/14/2022

Clustering with kmeans() and hclust()

We will begin by making up some data to cluster.

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



Run kmeans()

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

```
## K-means clustering with 2 clusters of sizes 30, 30
##
## Cluster means:
##         tmp
## 1  2.827370 -3.392494
## 2 -3.392494  2.827370
##
## 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 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 2 2
##
## Within cluster sum of squares by cluster:
## [1] 55.45198 55.45198
##  (between_SS / total_SS =  91.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Q. What size is each CLuster?

```
k$size
```

```
## [1] 30 30
```

Q. Cluster centers

```
k$centers
```

```
##         tmp
## 1  2.827370 -3.392494
## 2 -3.392494  2.827370
```

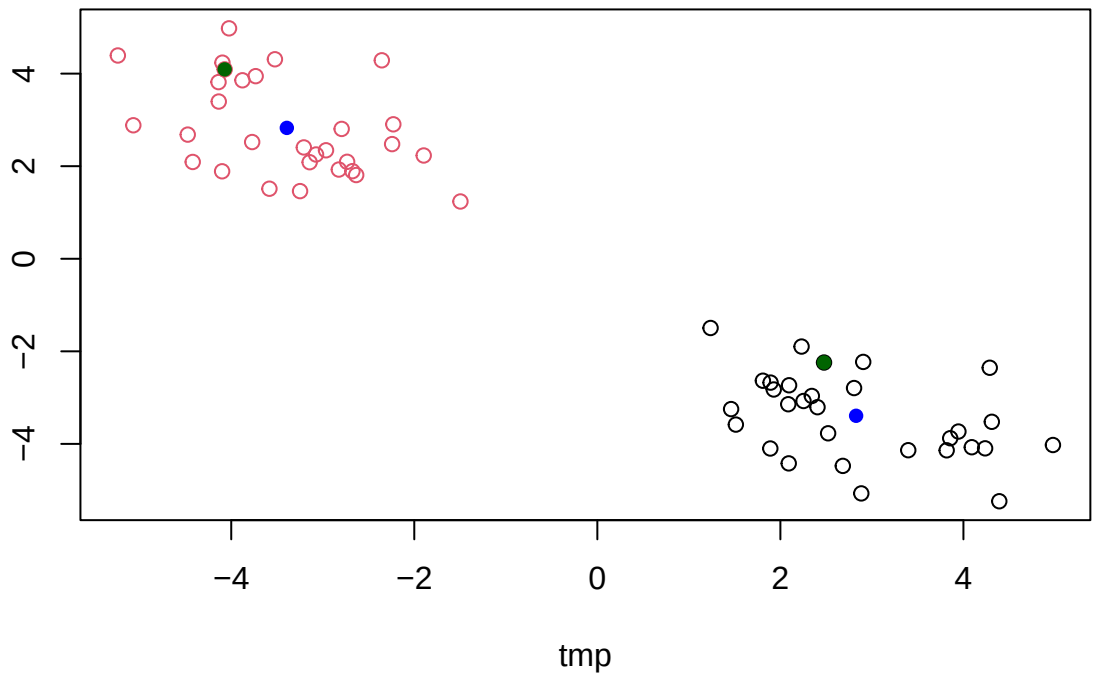
Q. Membership vector

```
k$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 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 2 2
```

Plot our data with the clustering result.

```
plot(x, col=k$cluster) +
points(k$centers, col="blue", pch=16) +
points(x[50,1], x[50,2], col="darkgreen", pch=16) +
points(x[19,1], x[19,2], col="darkgreen", pch=16)
```




```
## integer(0)
```

hclust()

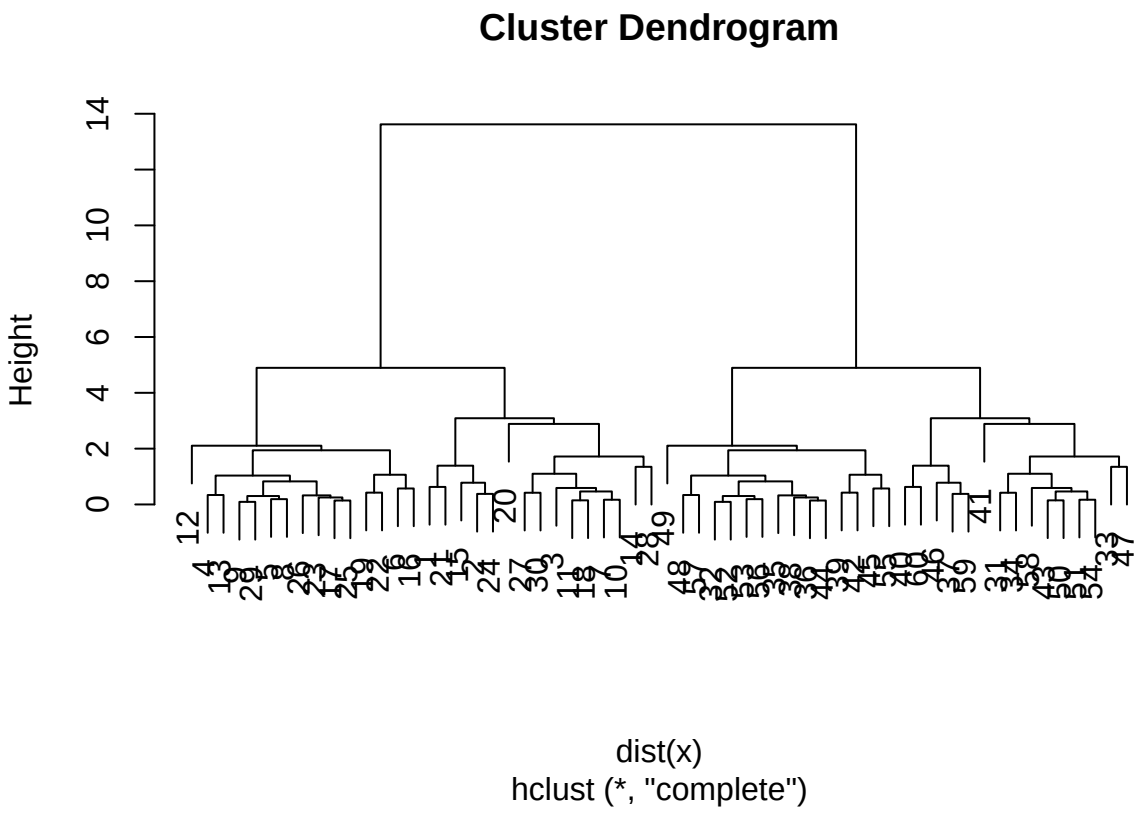
Hierarchical Clustering

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

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

Plot method for hclust()

```
plot(hc)
```



Principal Component Analysis

Data Import

```
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?
>
> > We can use dim() to return the number of rows and columns or we can use ncol() and nrow() together to return seperately, the number of columns and rows.

```
# Using the dim() function to return the number of rows and columns in the data frame.
dim(x)
```

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

There are 17 rows and 5 columns.

Checking your data

```
#Preview the first 6 rows of the data frame
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
```

We want only 4 columns, the first column x needs to be the rownames/index.

```
# Note how the minus indexing works
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
```

Check the dimensions again

```
dim(x)
```

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

Using an alternate way to set the index as the strings in column x

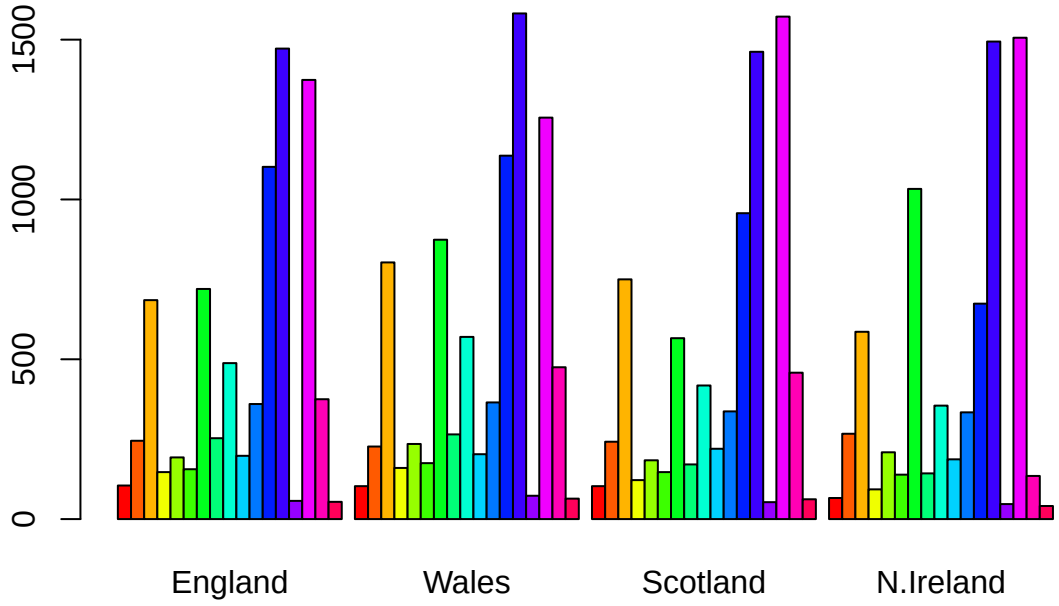
```
x <- read.csv(url, row.names=1)
head(x)
```

```
##                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
```

> [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 preferred the second approach which utilizes the "row.names" argument in "read.csv()". This method is more robust in certain circumstances because if you run the first code block more than once it will continue to move the index to the next column on the right. Additonally, the second method requires less code and is more succint.

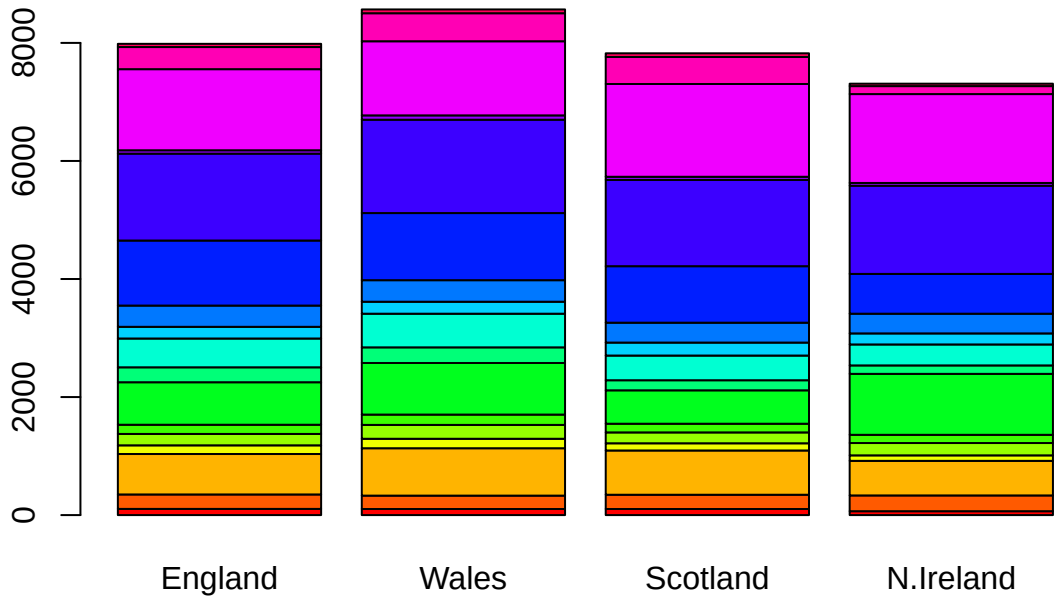
Spotting Major Differences and Trends

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



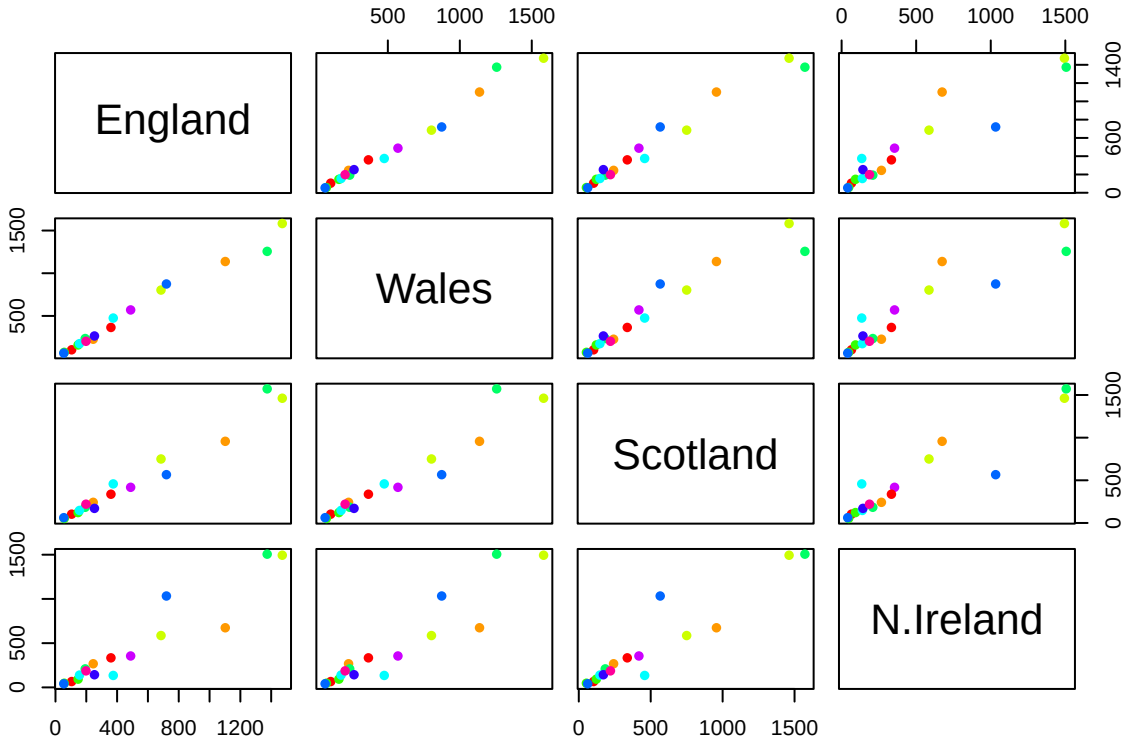
> [Q3] Changing what optional argument in the above barplot() function results in the following plot? » Changing beside to False

```
#Graphing the same barplot as above, with beside set to False
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?

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



» If a given point falls on the diagonal this means that two countries consume the same amount of the type of food that the dot is representing.

> [Q6] What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set? > From this plot we can see that N. Ireland is more different than the other countries of the UK, however, we cannot necessarily see how it is different.

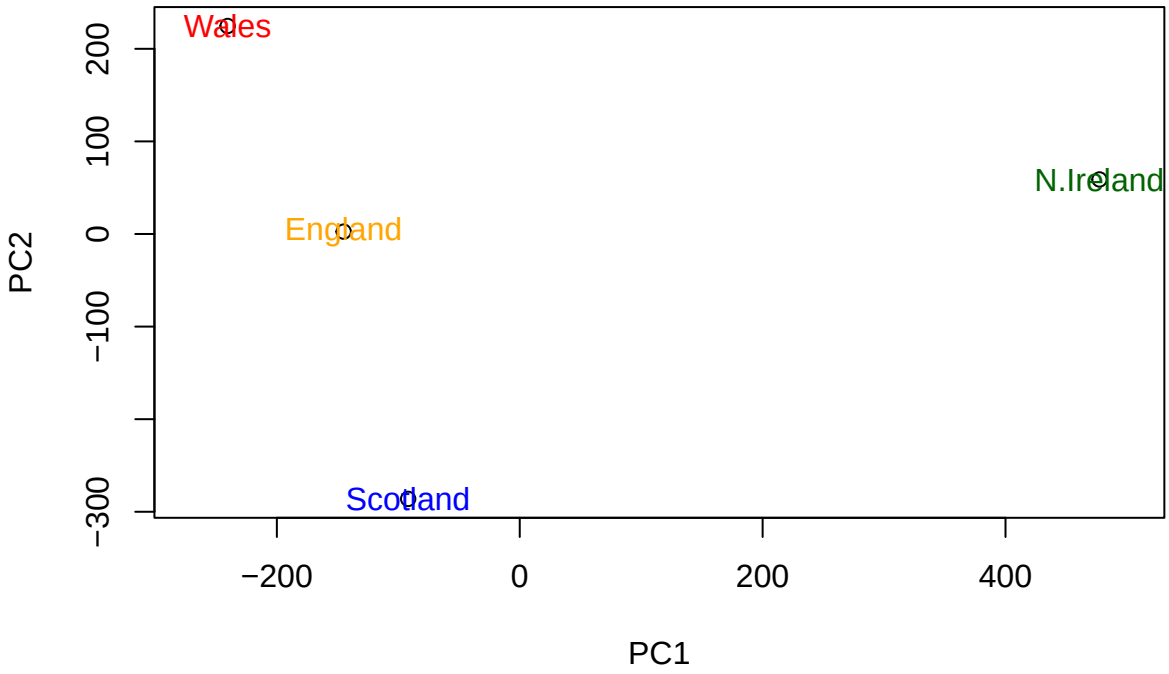
PCA to the Rescue!

```
# Use the prcomp() PCA function. This function requires 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 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
```

> [Q7] Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points. [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 PC1 vs PC2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))+
text(pca$x[,1], pca$x[,2], labels=colnames(x), col= c("orange", "red" , "blue", "darkgreen"))
```




```
## integer(0)
```

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

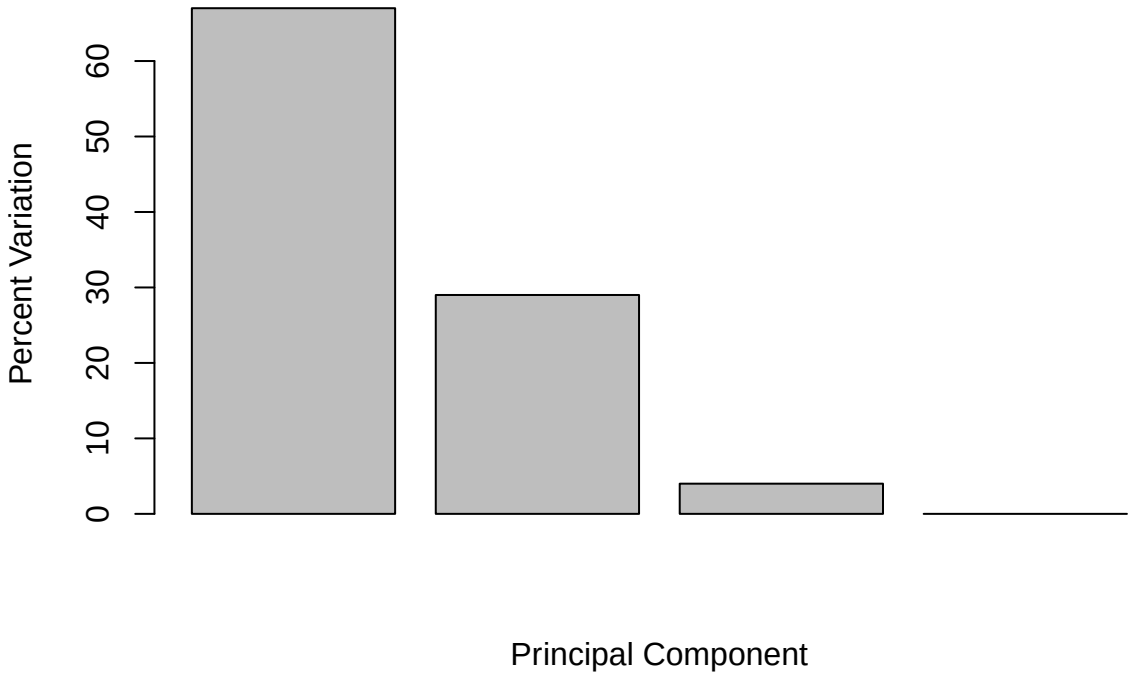
```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

```
## [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
```

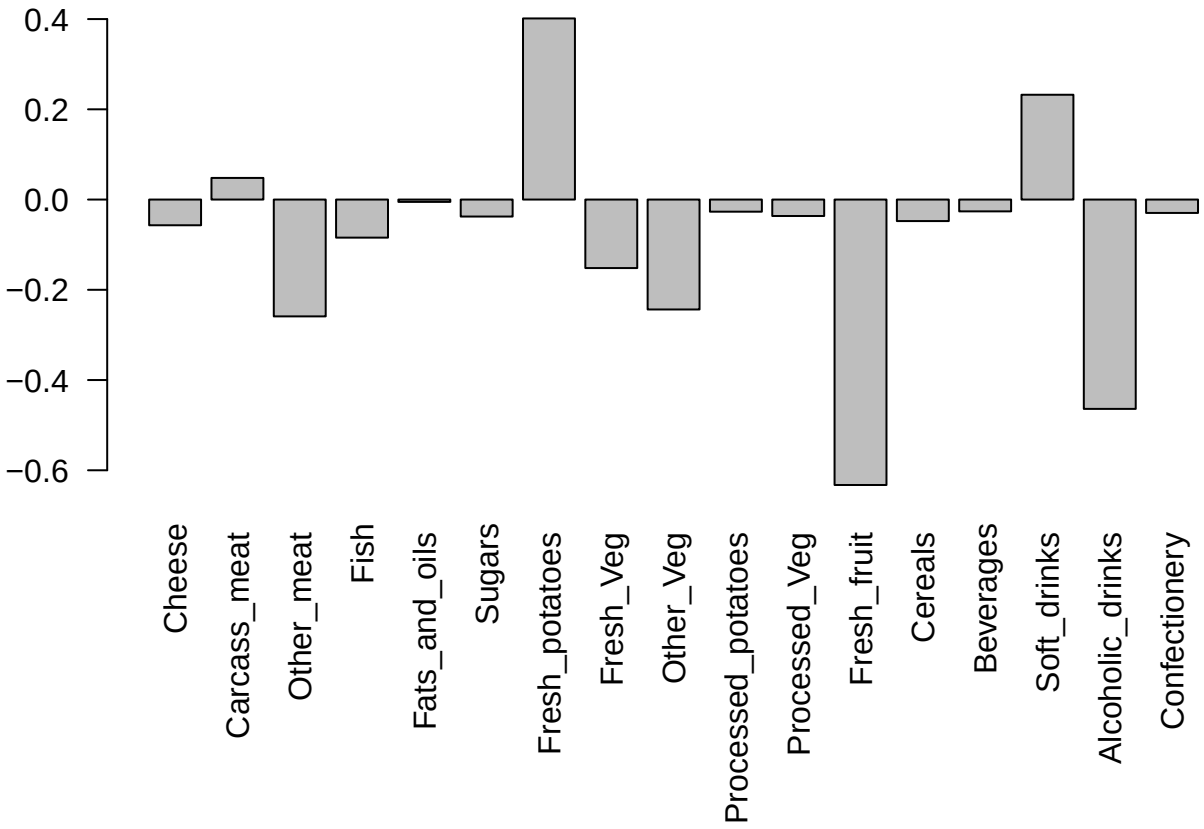
```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```



Digging Deeper (variable loadings)

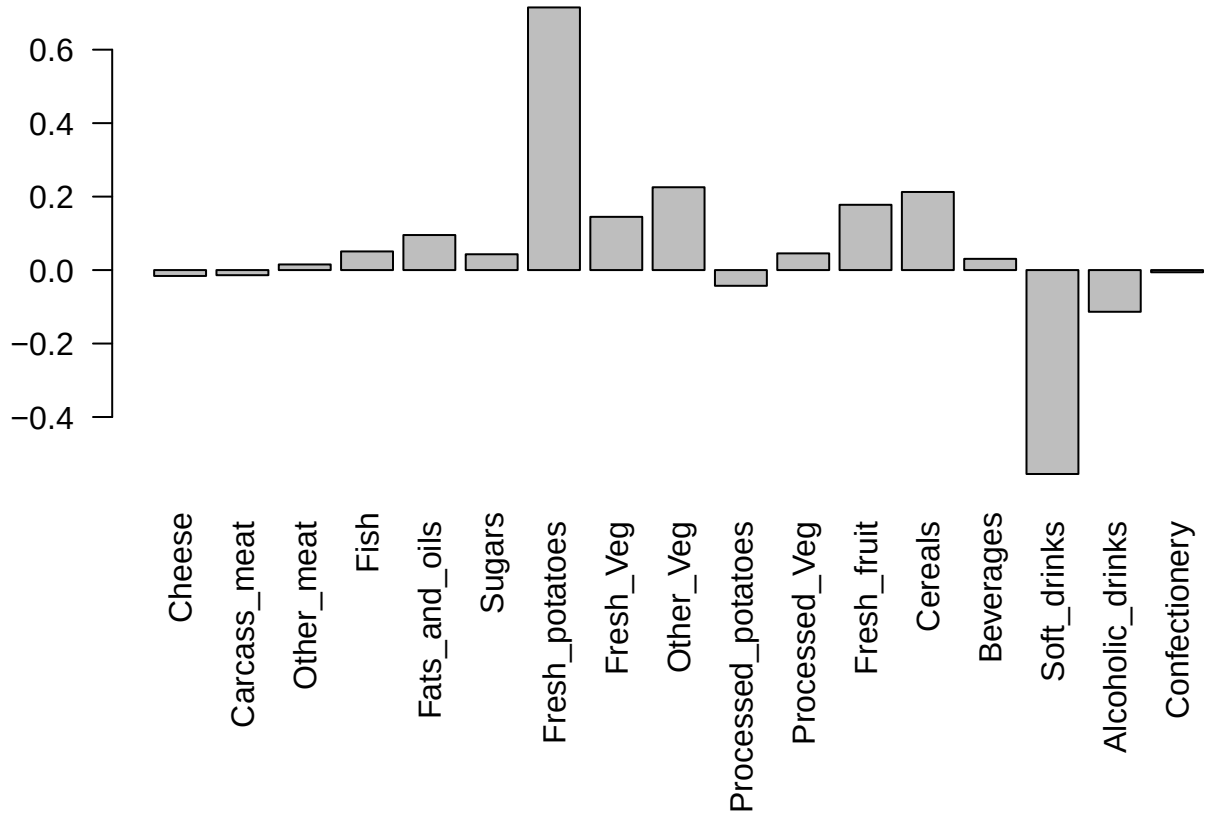
Lets focus on PC1 as it accounts for > 90% of variance

```
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?

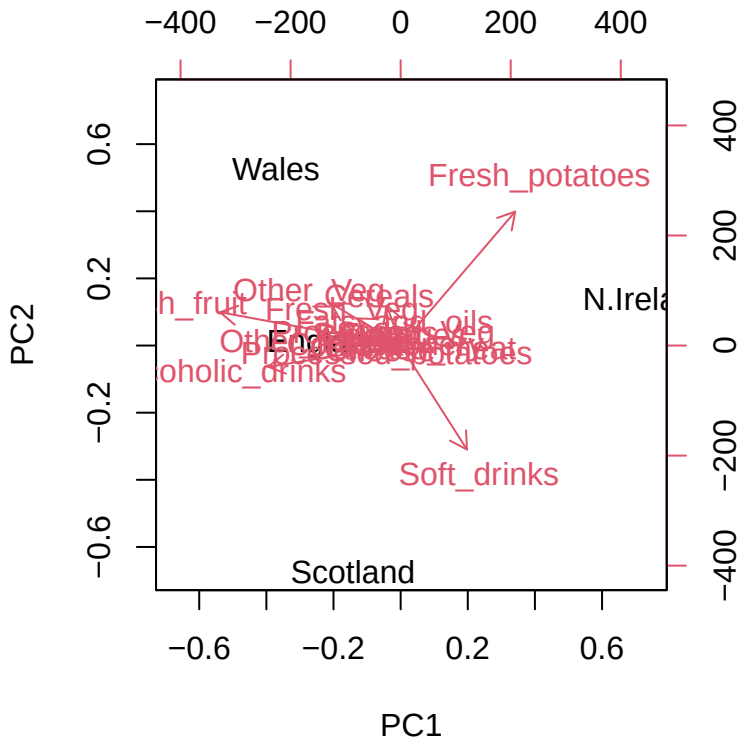
```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```



PC2 tells us more about the differences between Scotland and Wales. Soft_drinks and sugars are the two predominate groups here with soft_drinks on the negative, therefore Scotland consumes more soft drinks compared to the other UK countries. However, Wales and possibly N. Ireland consume sugars more than Scotland.

Bigplots

```
## 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
```

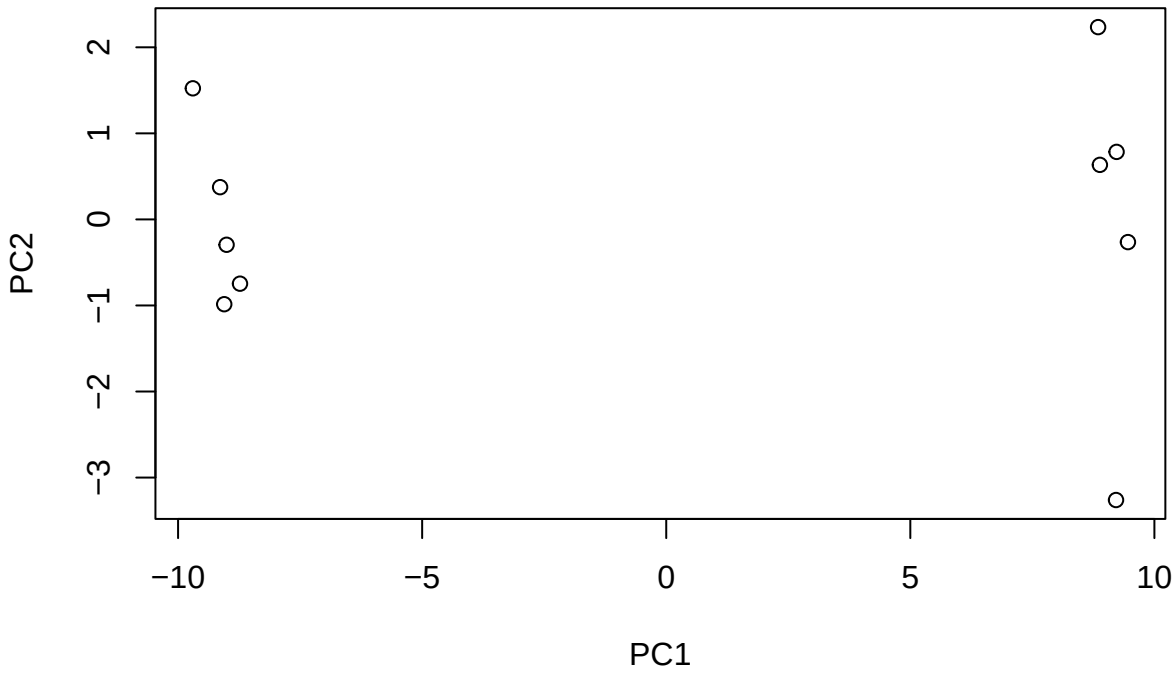
[Q10] How many genes and samples are in this data set?

```
#Find the number of rows and columns of the dataframe
dim(rna.data)
```

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

There are 100 genes (rows) and 10 samples (columns) in this data set.

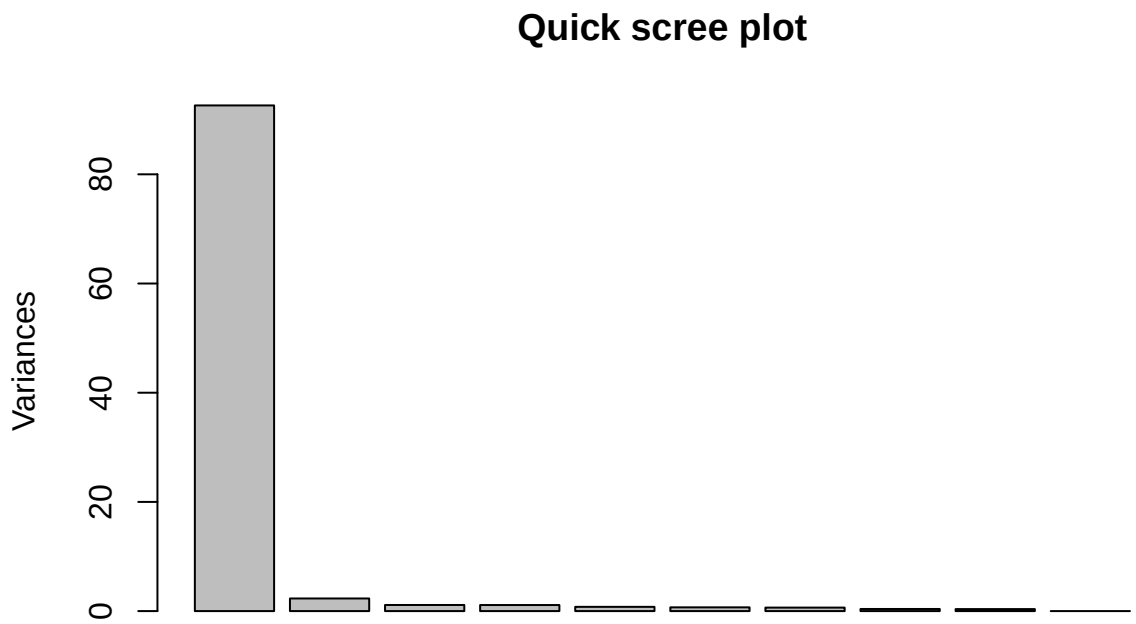
```
## Again we have to take the transpose of our data
pca <- prcomp(t(rna.data), scale=TRUE)

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




```
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
```

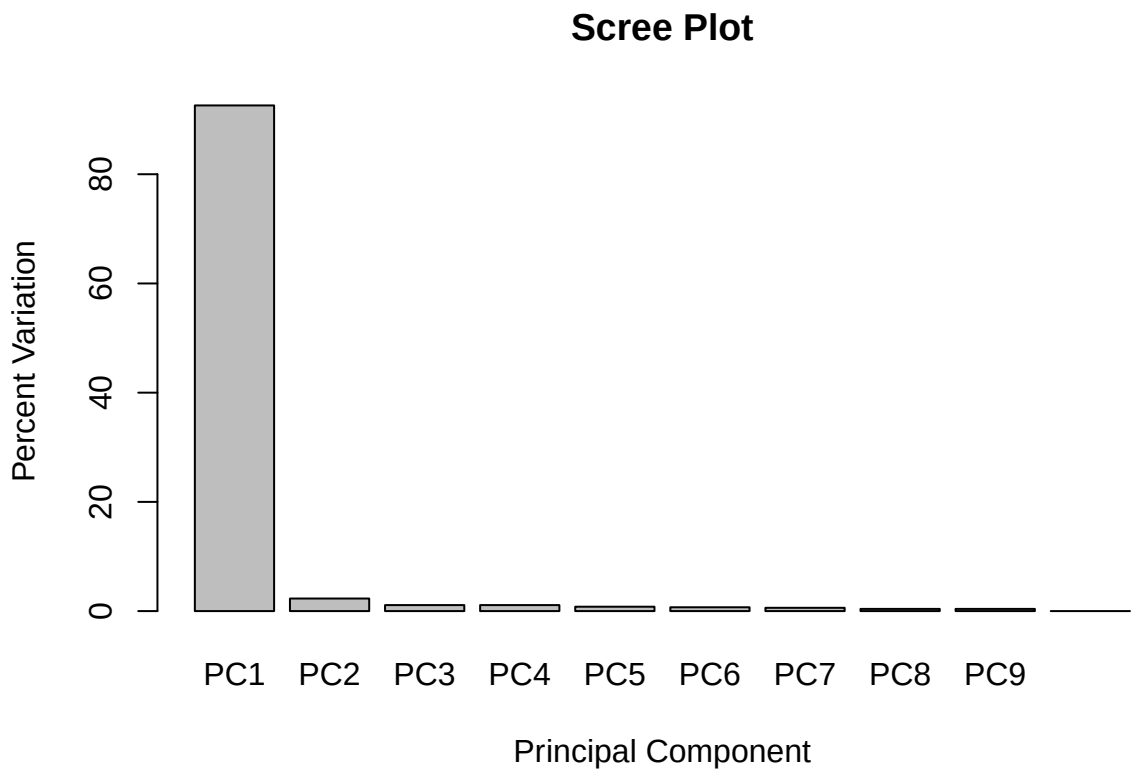
```
plot(pca, main="Quick scree plot")
```



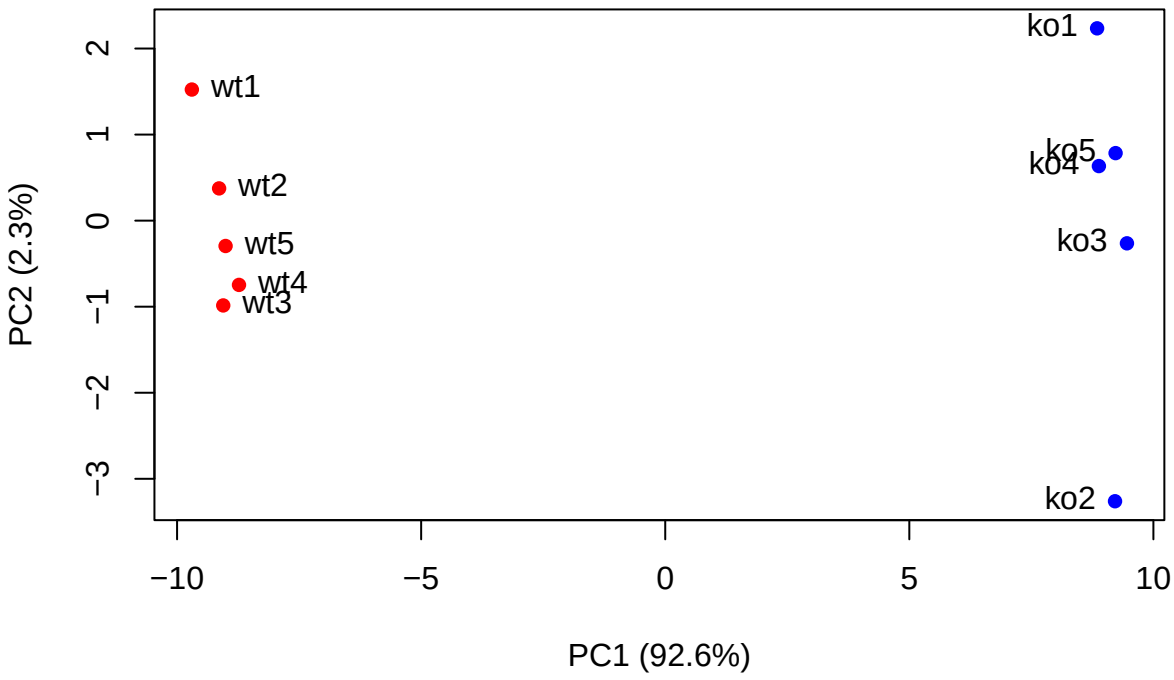

```
## Variance captured per PC
pca.var <- pca$sdev^2

## 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
```

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

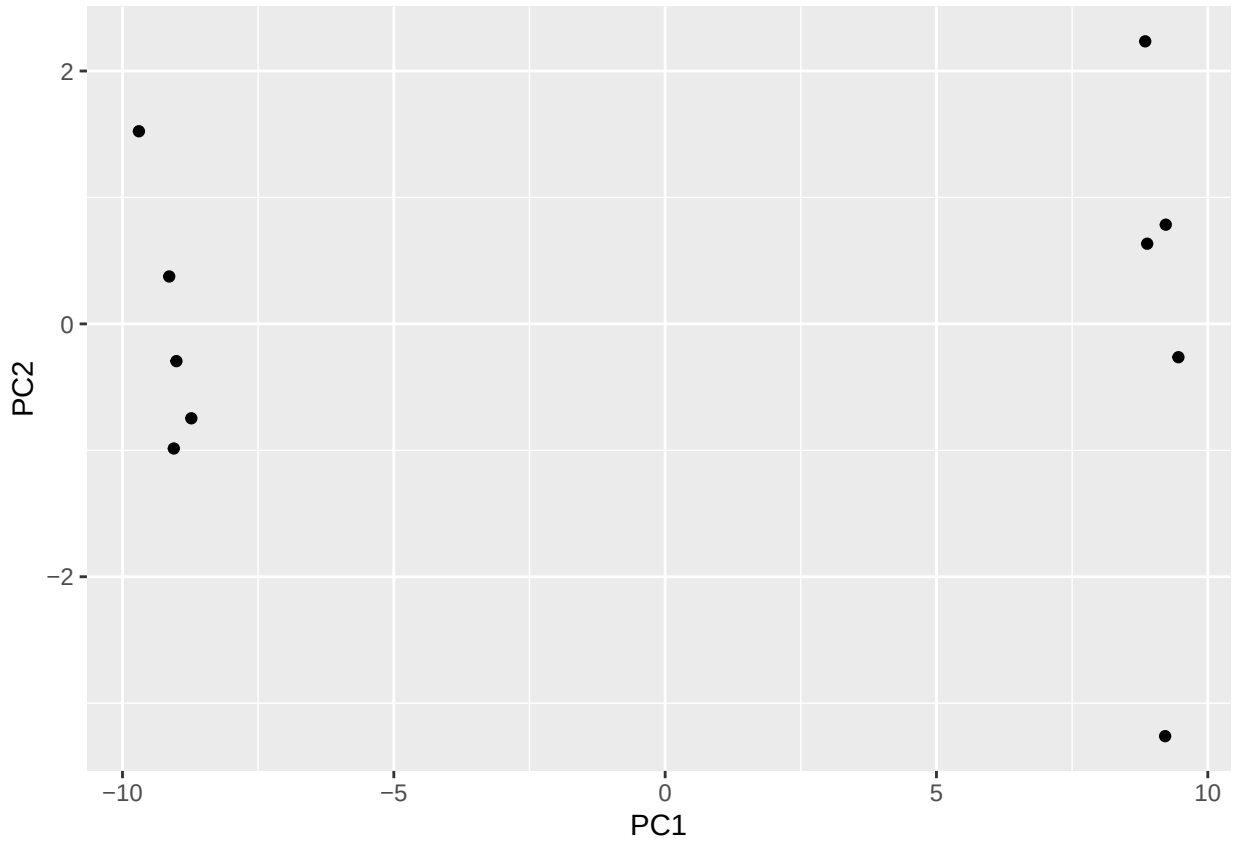



```
## A vector of colors for wt and ko samples
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

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)))
```



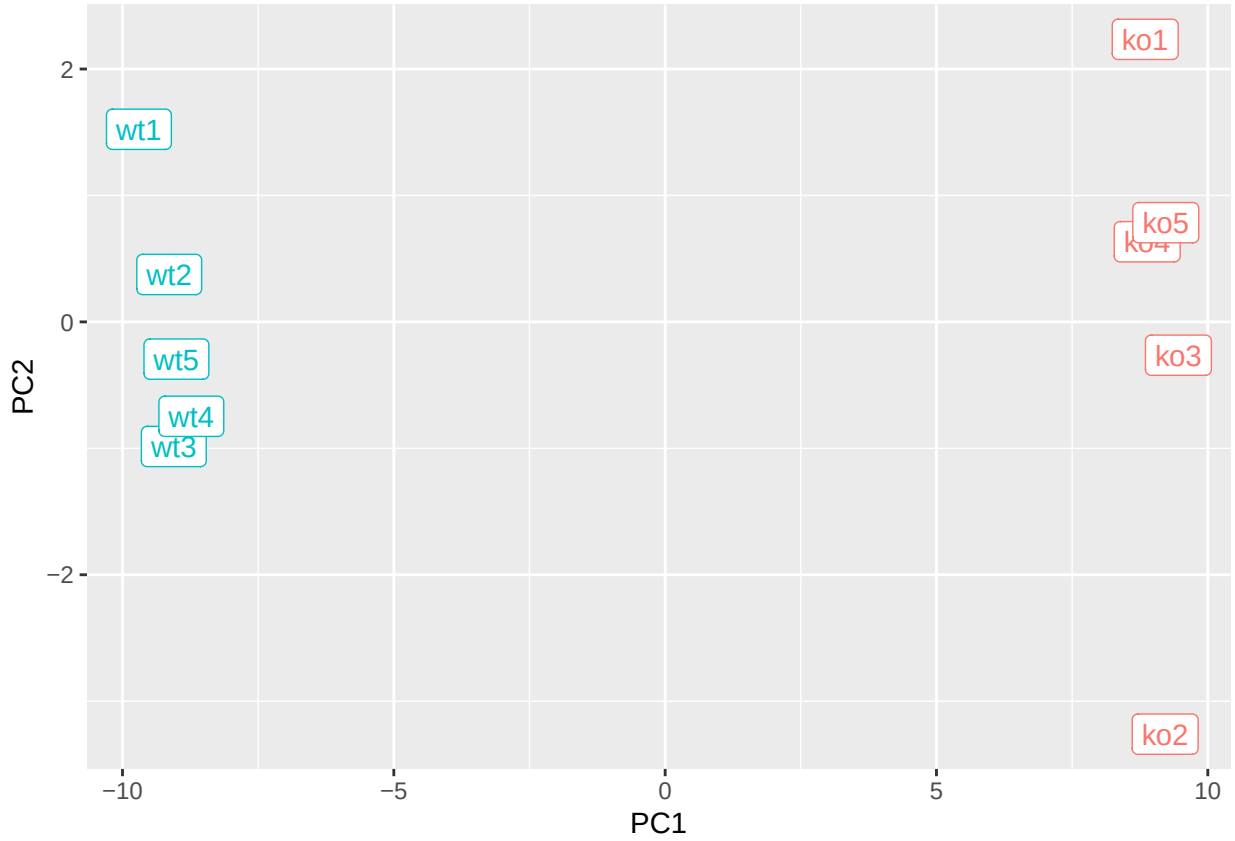
Using ggplot

```
library(ggplot2)

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

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



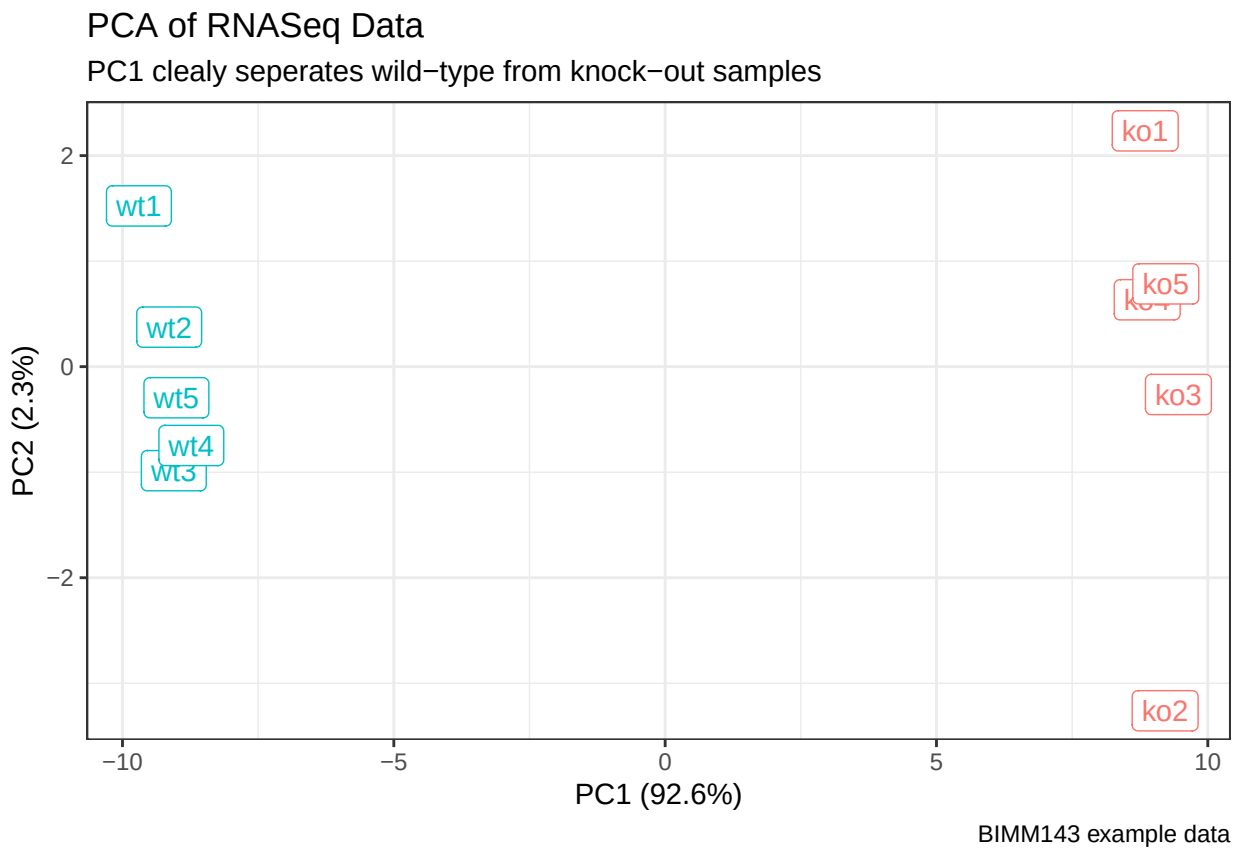
add column specific color and labels

```
# 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)
p
```



#Adding titles and captions

```
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()
```



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"
```