Class7.Rmd

Ayse

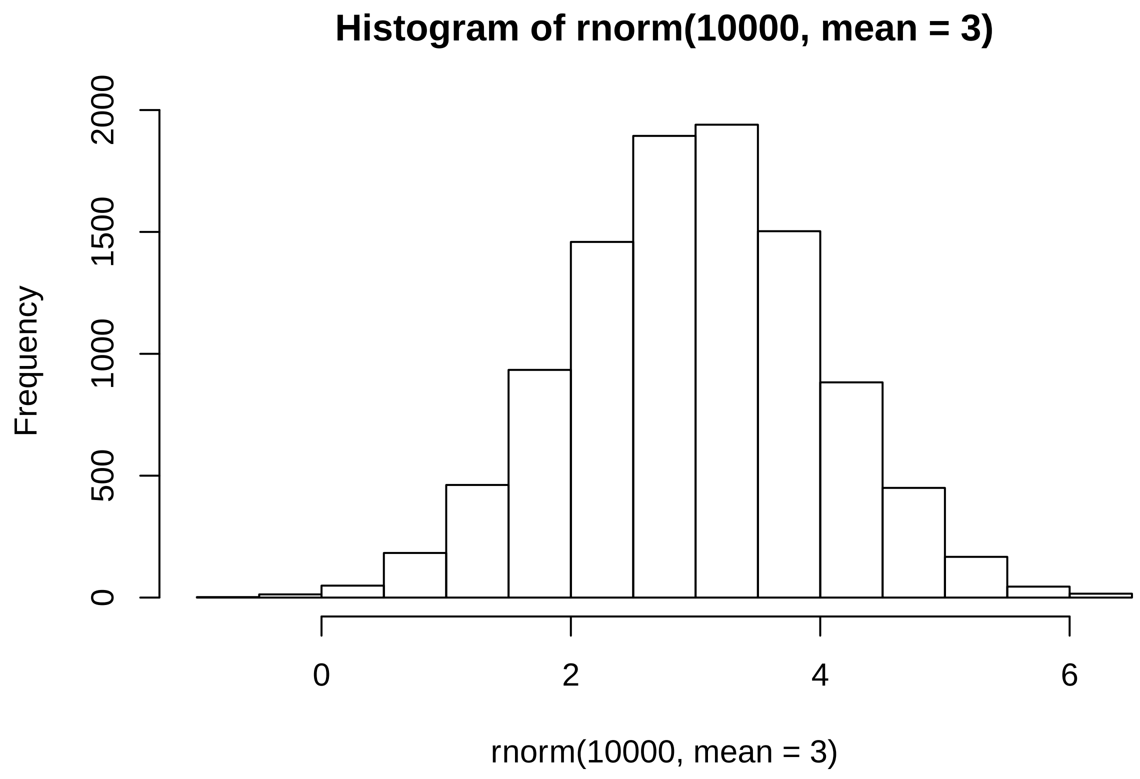
2/9/2022

Introduction to machine learning:

unsupervised, supervised, and reinforcement learning. We will learn 2 clustering methods - K.means and hierarchical clustering. We will also learn how to find patterns in high dimensional data: dimensionality reduction, visualization, and structure analysis (PCA).

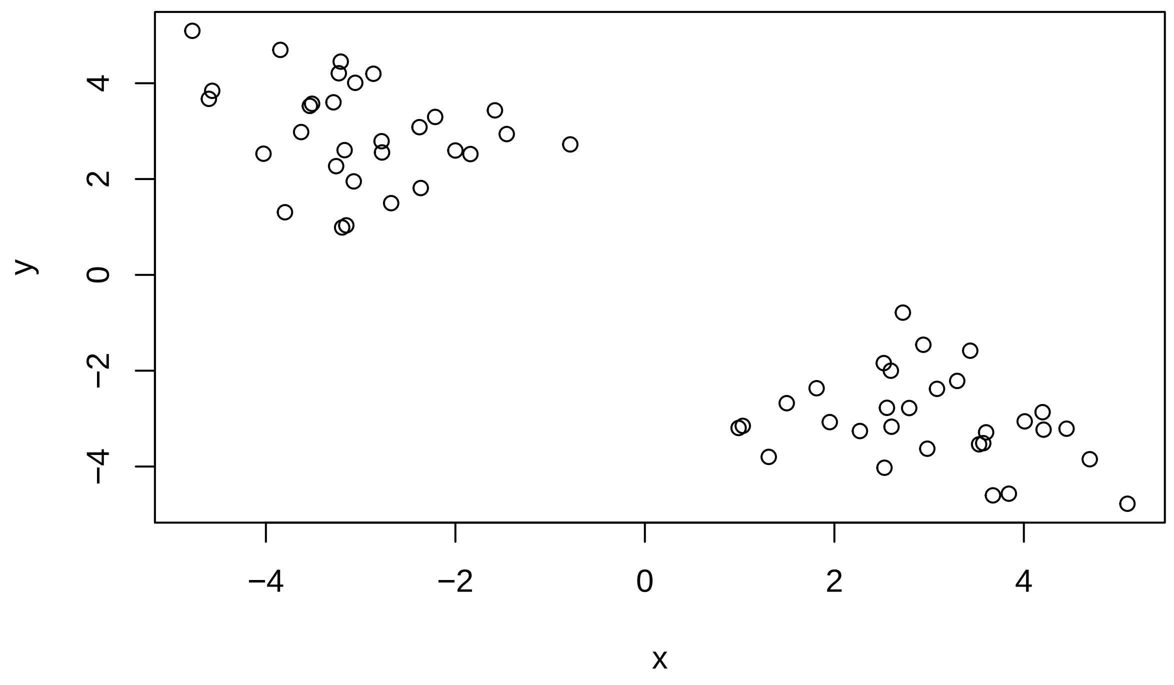
Generating random data in a normal distrubtion

```r
hist(rnorm(10000, mean=3))
```



generating data for K means testing

```
#generate tmp vector from two normal distributions of mean 30, sd -3 or 3
tmp <- c(rnorm(30, -3), rnorm(30, 3))
#use column bind - bind tmp data and reverse of tmp data as 2 columns
n <- cbind(x=tmp, y=rev(tmp))
plot(n)
```



A limitation of kmeans is that we have to tell it how many clusters we want. nstart - point where distance of the points in dataset is calculated

```
k <- kmeans(n, centers=2, nstart = 10)
k
```

```
## K-means clustering with 2 clusters of sizes 30, 30
##
## Cluster means:
##           x         y
## 1  2.992671 -3.021523
## 2 -3.021523  2.992671
##
## Clustering vector:
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
## [39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
##
## Within cluster sum of squares by cluster:
## [1] 57.96134 57.96134
##  (between_SS / total_SS =  90.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

the available components can be called from the list using $. for example, how many points are in each cluster?

```
k$size
```

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

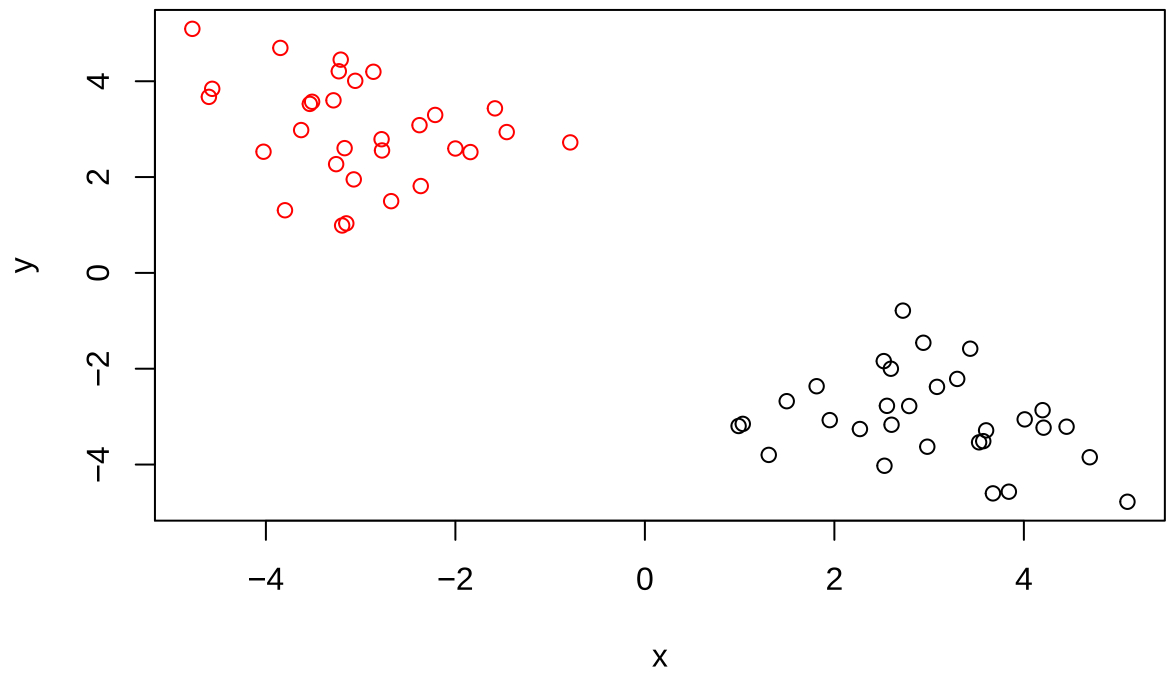
What are the centroids of each cluster?

```
k$centers
```

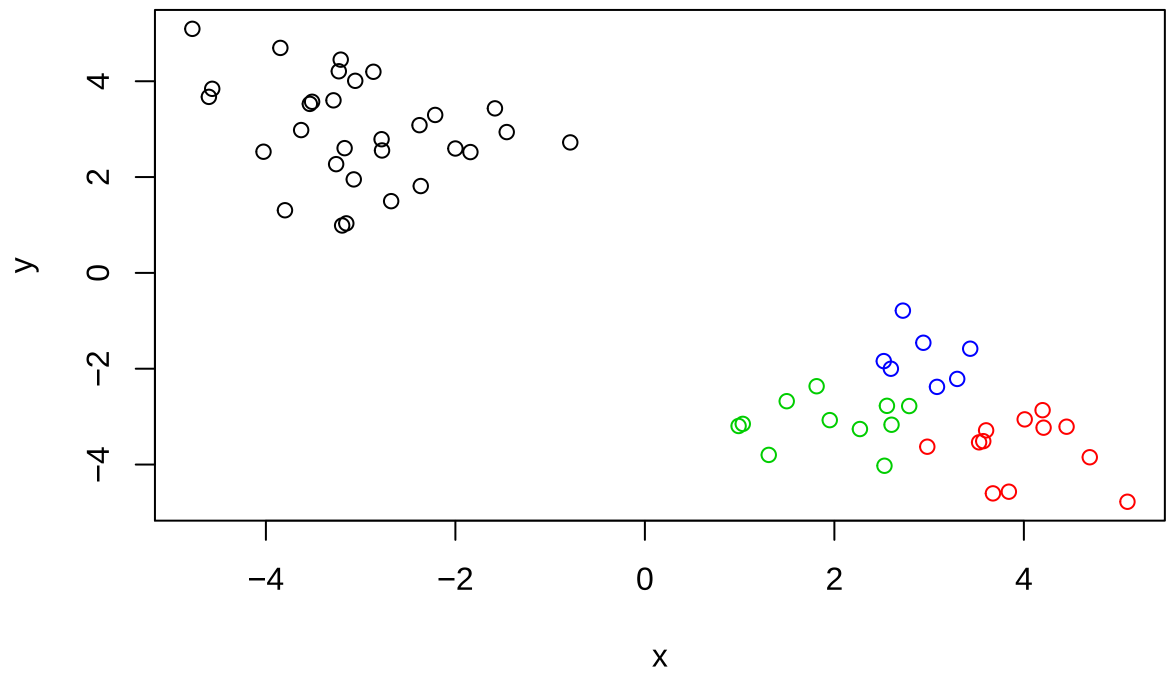
```
##           x         y
## 1  2.992671 -3.021523
## 2 -3.021523  2.992671
```

Determining cluster assignment and plotting x colored by kmeans cluster assignment and cluster centers as blue points

```
#cluster is the membership factor - indicates the cluster to which each point is allocated
a <- k$cluster

plot(n, col=a)
```



```
k <- kmeans(n, 4, 10)
plot(n, col=k$cluster)
```



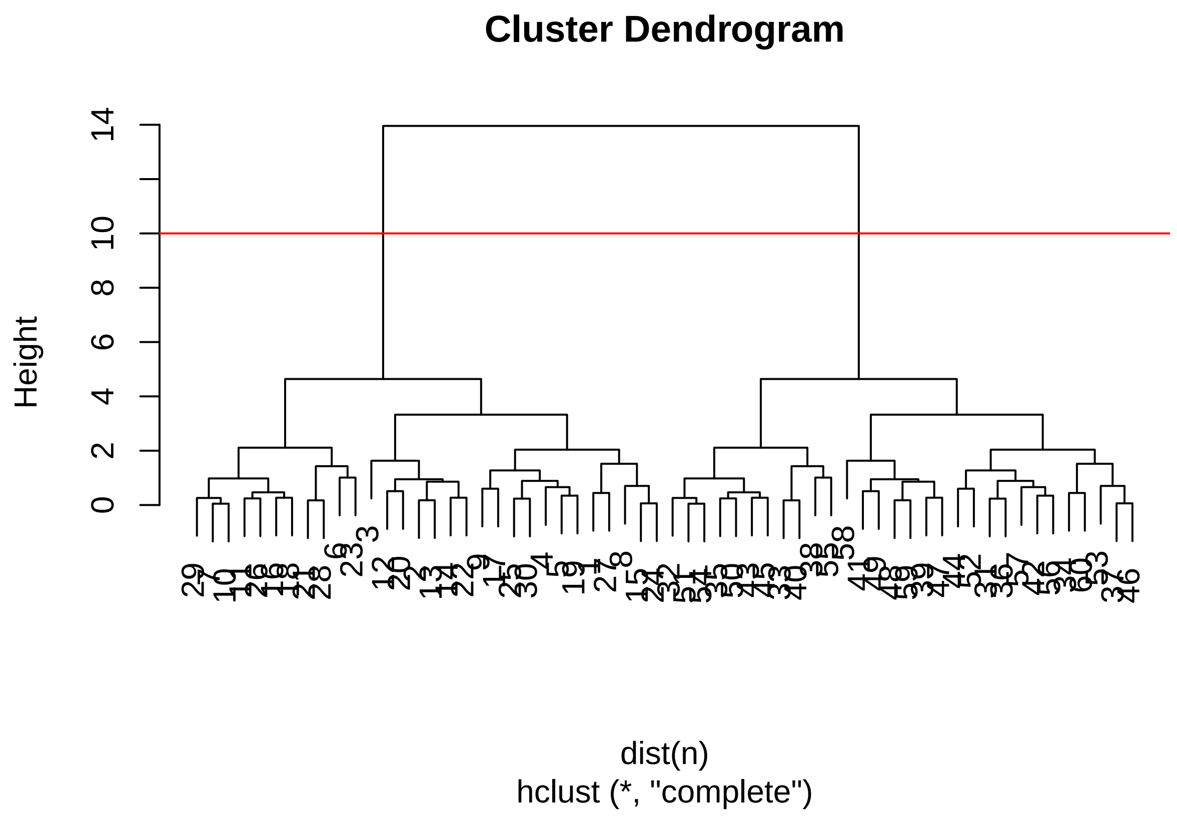
Hierarchical clustering

hclust - must give hclust a distance matrix, not raw data, that can be made using dist(x)

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

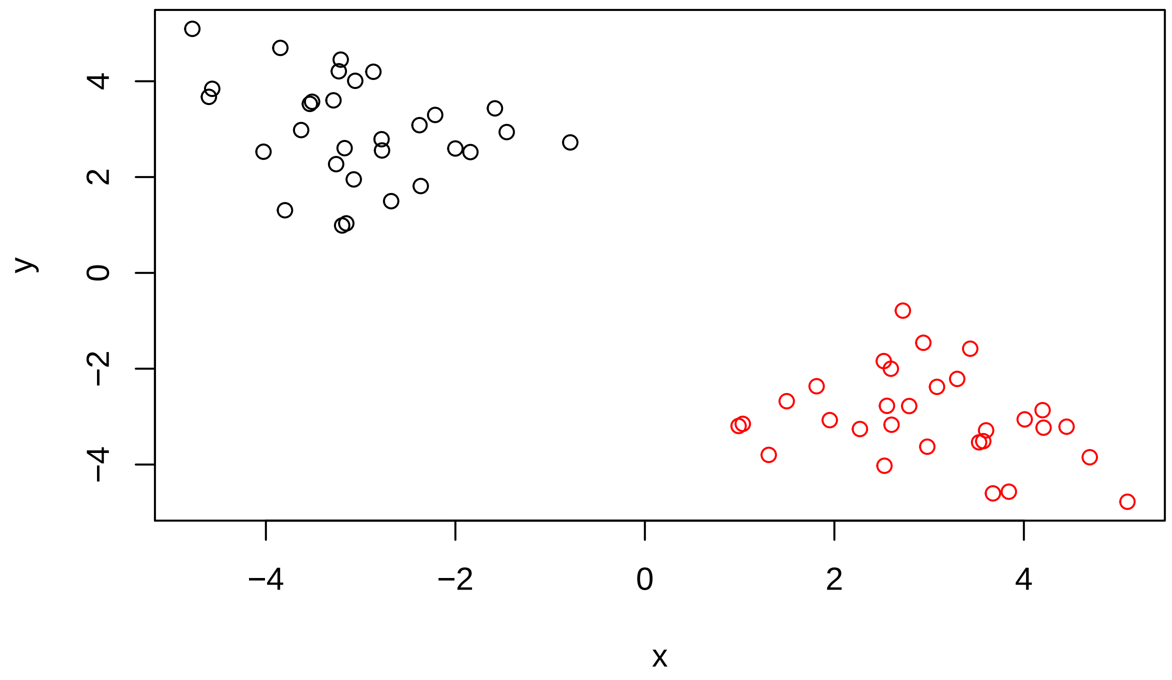
```
##
## Call:
## hclust(d = dist(n))
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 60
```

```
#will make a dendogram showing clusters
plot(hc)
abline(h=10, col="red")
```



cut tree to get cluster membership vector

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



Lab class7

import data Q1: there are 5 columns and 17 rows.

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

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

```r
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
```

Fixing row names (should have 4 variables, but x has been counted as a variable)

```r
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
```

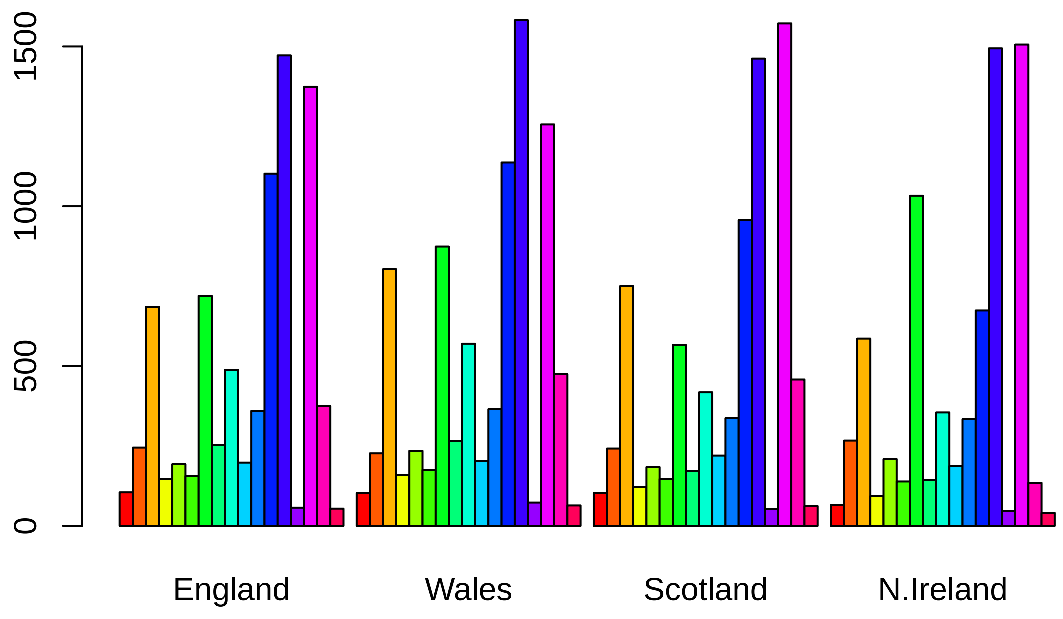
```r
dim(x)
```

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

Q2: I would prefer the row.names() argument being used, this will be more robust.

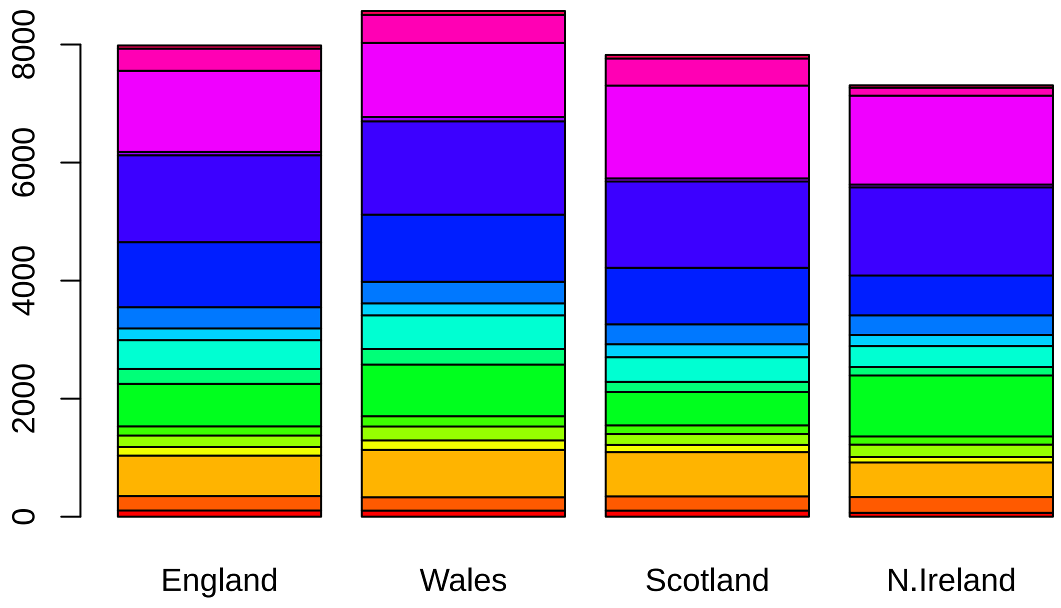
Visualize the data in a bar plot

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



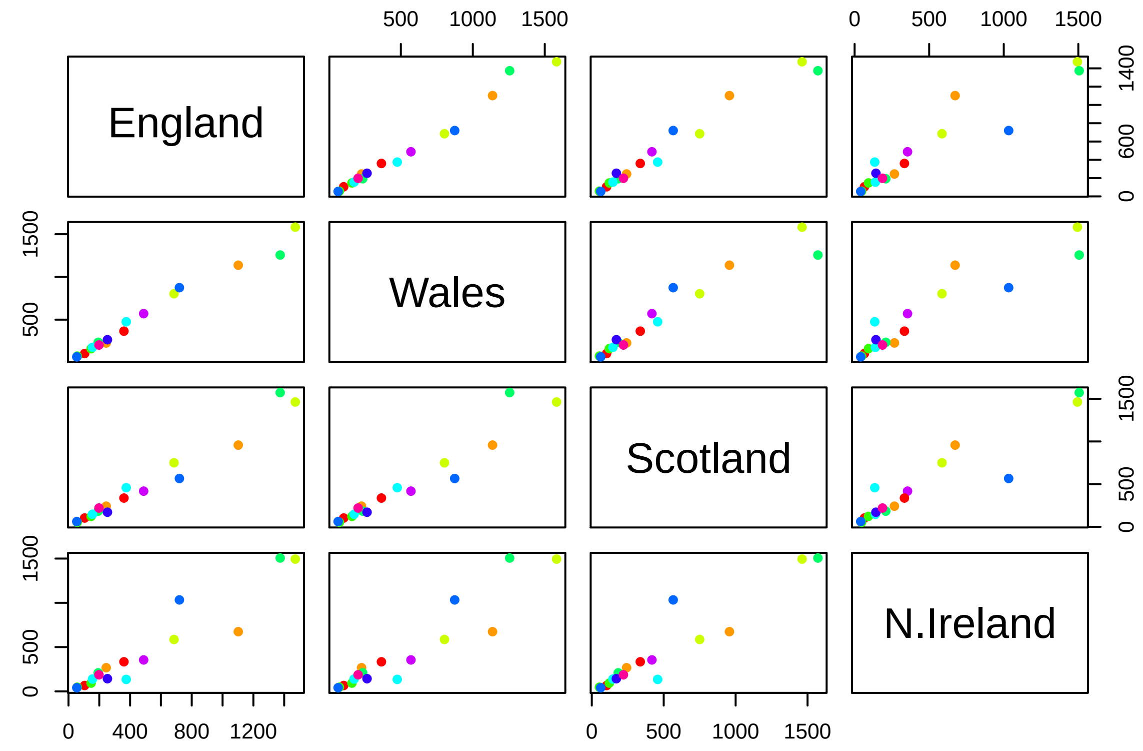
Q3: For a stacked plot, we can set the argument beside to FALSE

```r
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?

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



For the plot above, it is a matrix of scatterplots. It is plotting row values (different colored points) with values for one country on the x axis and the other on the y axis. Points on the diagonal will indicate nearly identical values for each country - that particular food is consumed in similar amounts in the two countries being plotted. Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set? I can already see that any plot including values from N. Ireland has fewer points close to or on the diagonal than the other pairs of countries. The consumptoin of various food in N. Ireland seems to be different from the other countries (compared to when those countries are compared to one another)

PCA

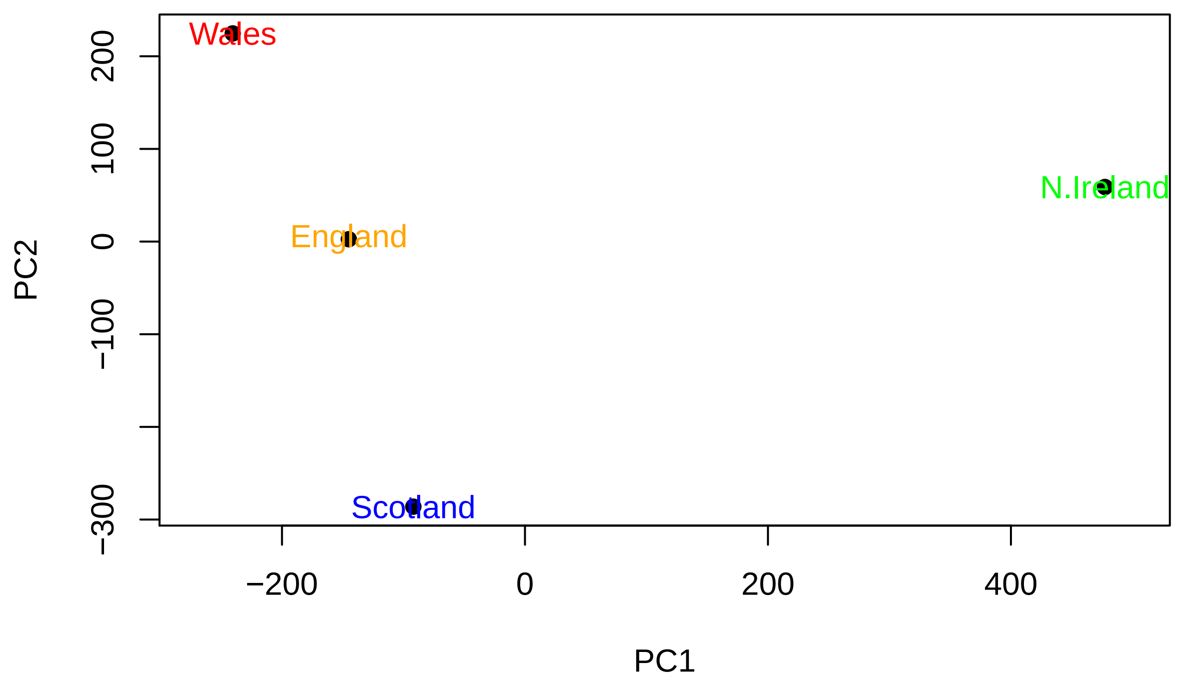
prcomp() is a base R function that performs pca - observations are rows and variables are columns. We need to transpose our data.

```
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 principal components are listed in order of importance above. PC1 accounts for more than 67% of the variance in the data, PC2 for 29%, and PC1 and PC2 together to >96%. #score plot The x in pca is what we want to plot tbe score plot below.

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500), pch=19)

text(pca$x[,1], pca$x[,2], colnames(x), col=c("orange", "red", "blue", "green"))
```

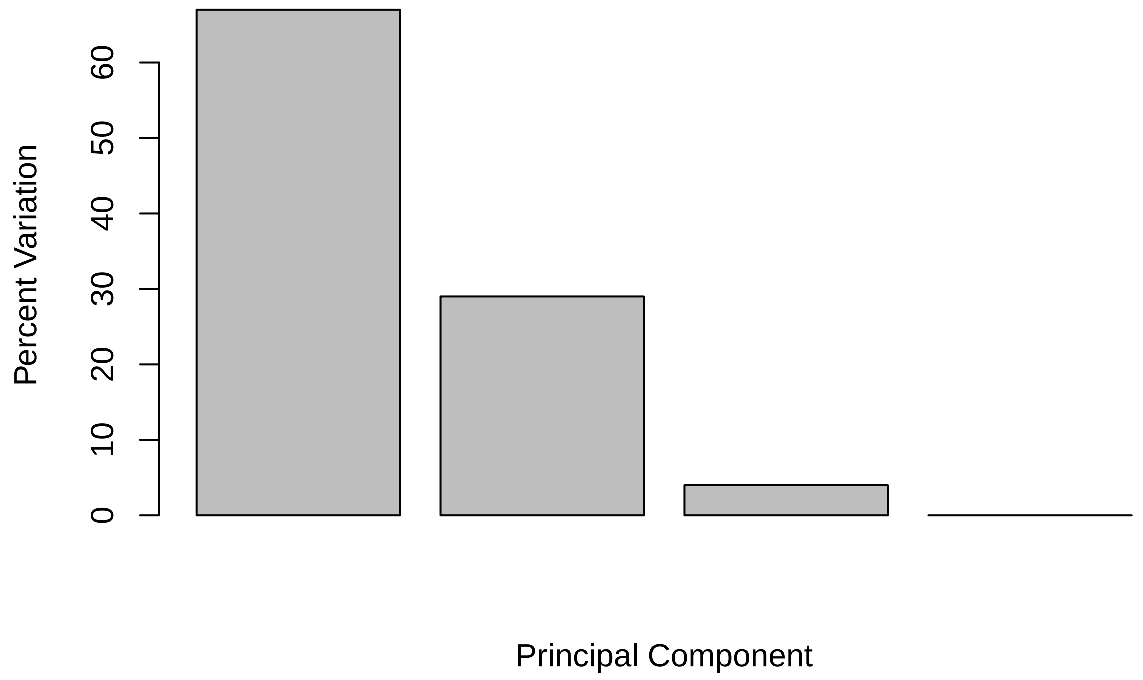


looking at how much variation each PC accounts for and summarizing it in a bar plot

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

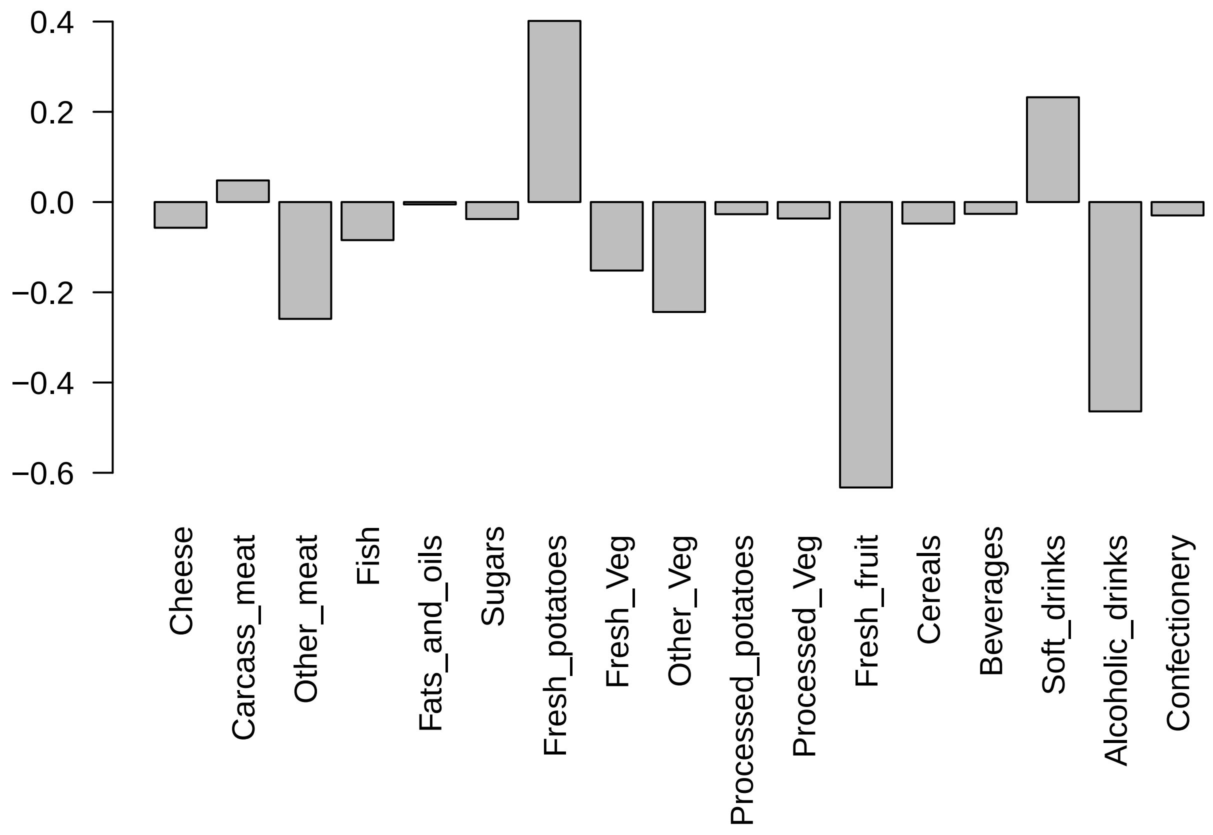
```
## [1] 67 29  4  0
```

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



Next, we can check the influence of each row on a principal component.

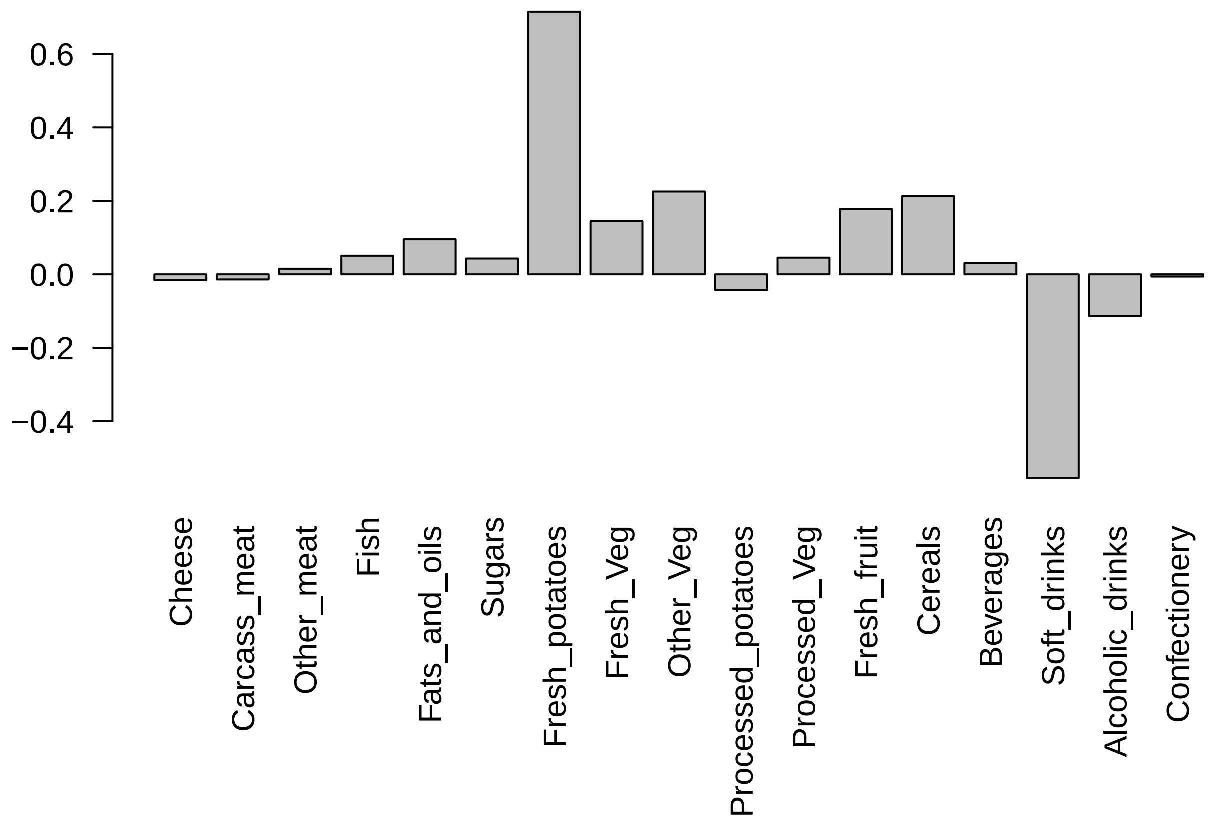
```r
#to specify margins, without this the row names are cutoff
par(mar=c(10, 3, 0.35, 0))
#make barplot of of the influence of each row on pc1
barplot( pca$rotation[,1], las=2 )
```



fresh potatoes, soft drinks, fresh fruit ,and alcoholic drinks are influencing pc1 the most

loadings plot for pc2:

```r
#to specify margins, without this the row names are cutoff
par(mar=c(10, 3, 0.35, 0))
#make barplot of of the influence of each row on pc1
barplot( pca$rotation[,2], las=2 )
```

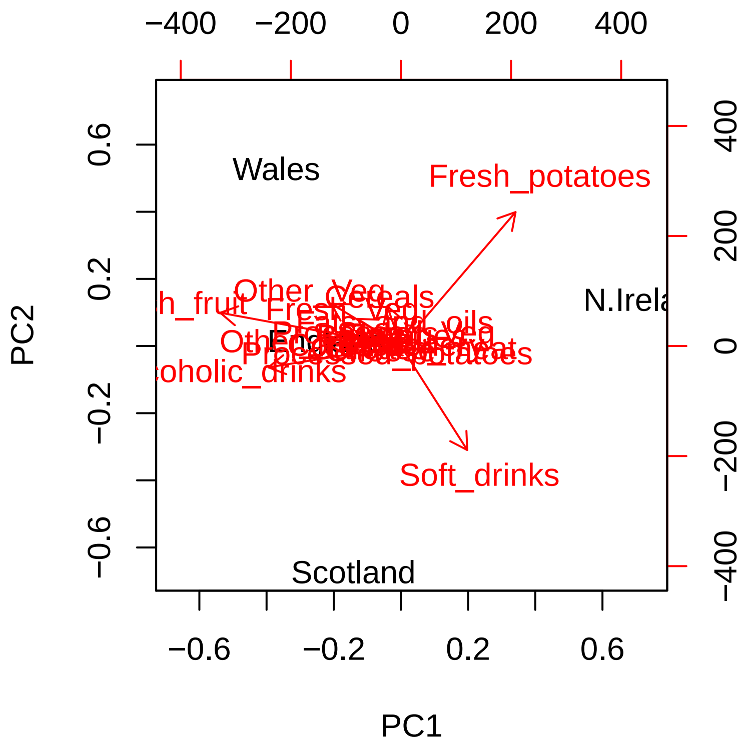


Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about? The most prominent foods are soft drinks and fresh potatoes, meaning that these are the rows that are most prominently influencing PC2. Based on this information, PC2 is telling us that the fresh potatoes characteristic is contributing to moving N. Ireland away from the other countries (more consumption in Ireland vs. other countries), and soft drinks is moving the other countries away from N. Ireland (less consumption in N. Ireland vs. other countries). This accounts for the second greatest amount of variance in the dataset.

biplots

This combines the loading plot and pca plot, is mainly useful for small datasets

```
biplot(pca)
```



PCA of RNAseq

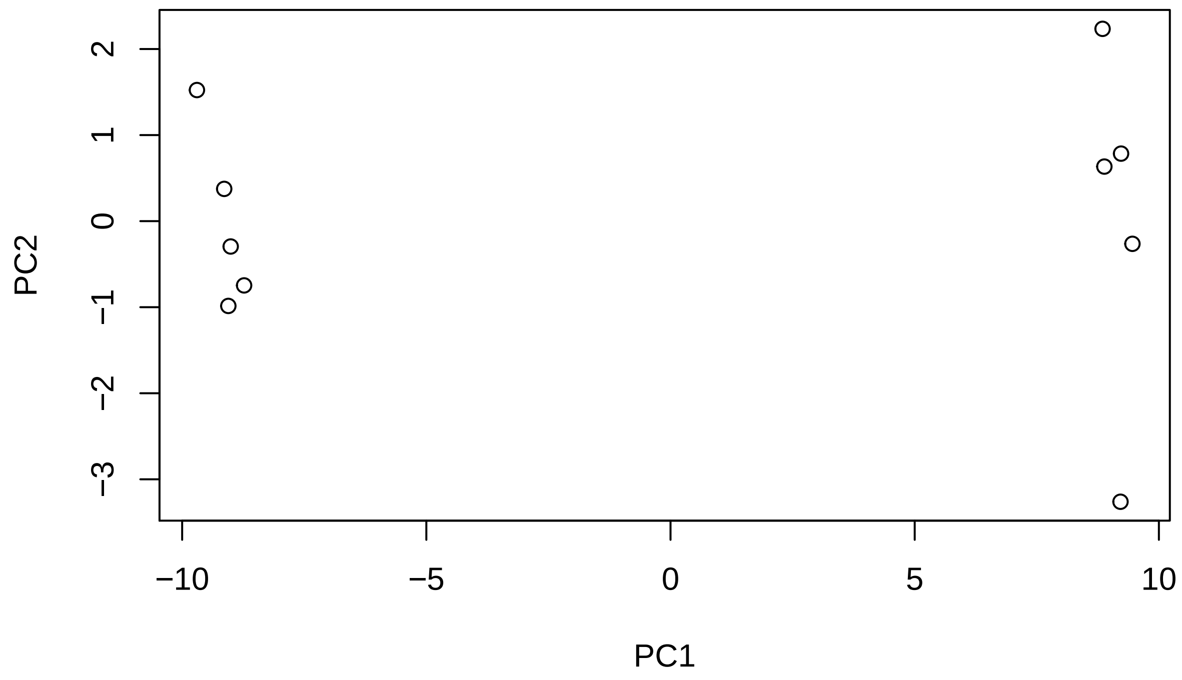
Load data

```
url2 <- "https://tinyurl.com/expression-CSV"
rna.data <- read.csv(url2, row.names=1)
dim(rna.data)
```

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

Q10: How many genes and samples are in this data set? There are 10 samples and 100 genes.

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

## score plot
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```

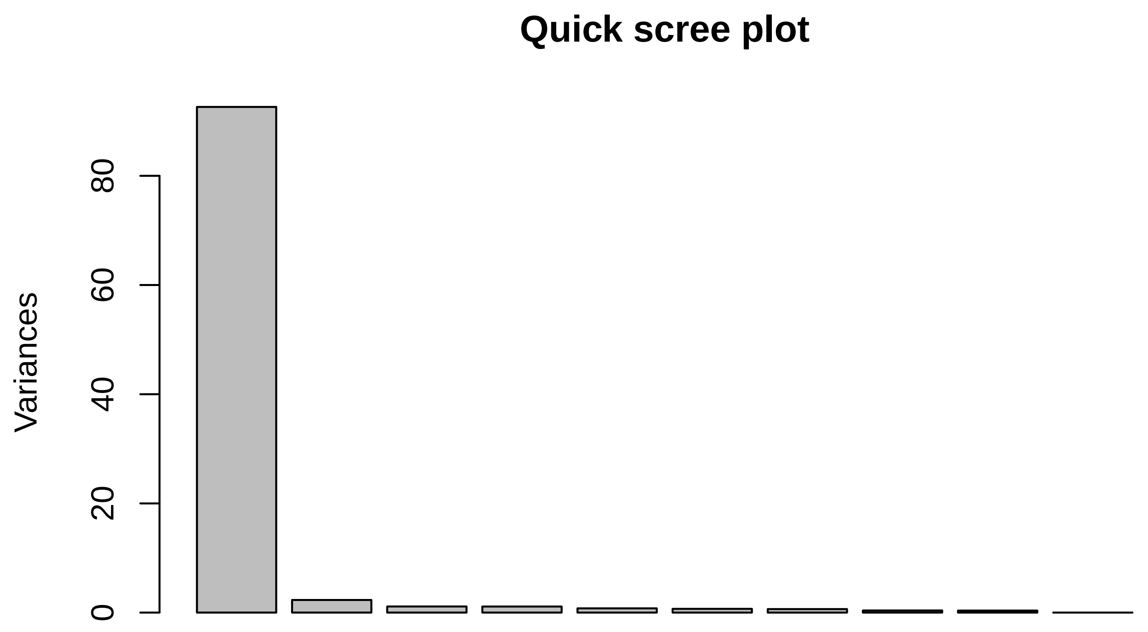


```r
summary(pca)
```

```
## 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
```

We can see that PC1 is accounting for >92% of the variance, and PC1 and PC2 together to greater than >94%. Lets make scree plot for this

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

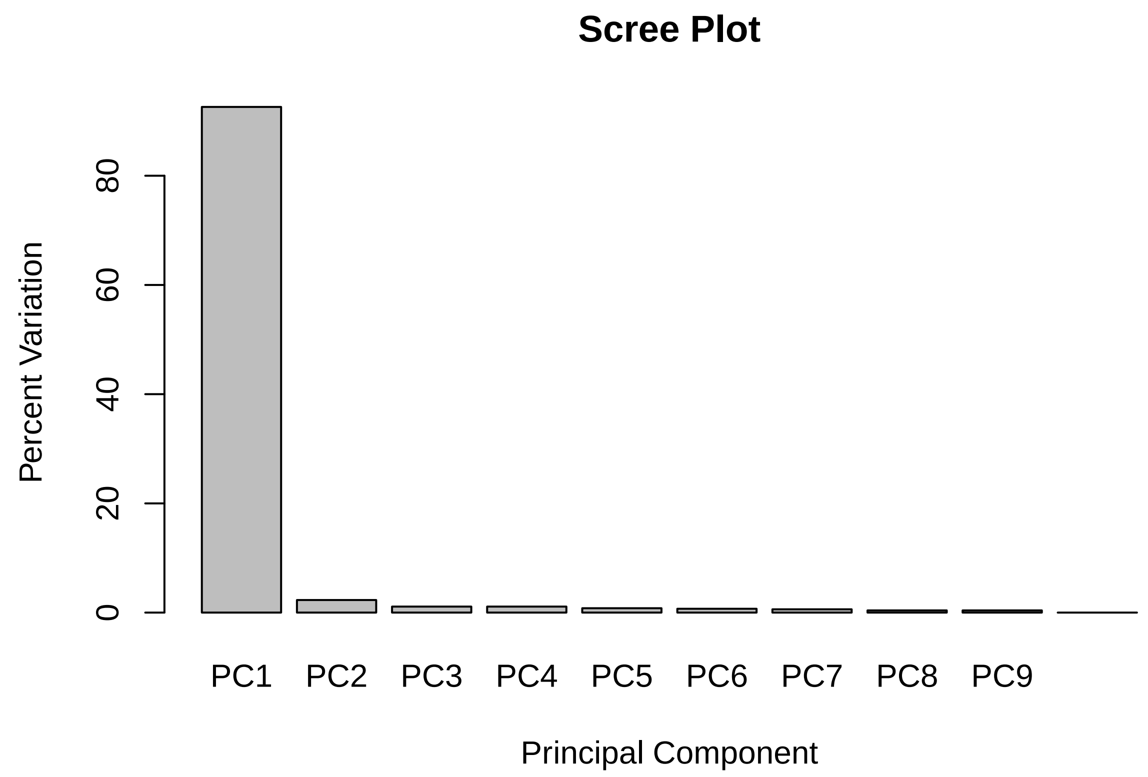


We can alternatively look at sdev (standard deviation) in pca to check how much variation each pc accounts for

```
## 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
```

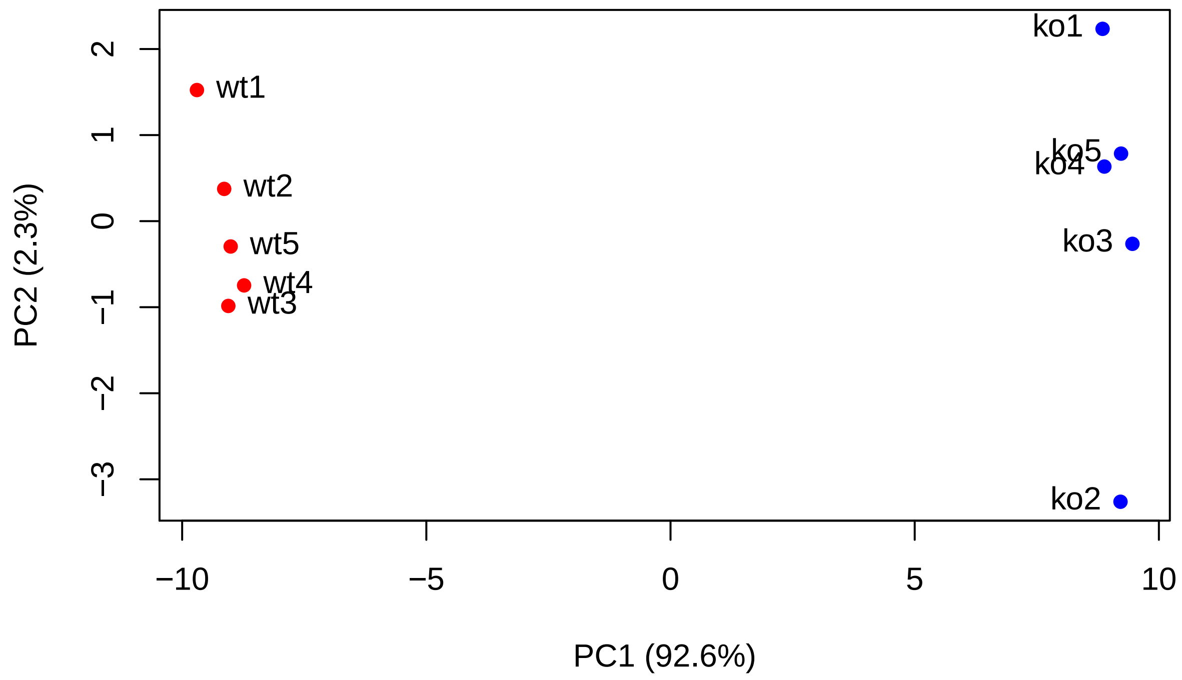
```
##  [1] 92.6  2.3  1.1  1.1  0.8  0.7  0.6  0.4  0.4  0.0
```

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



We will re-make the original score plot in a nicer way. First, we can make vectors of colors for the two types of samples (WT and KO)

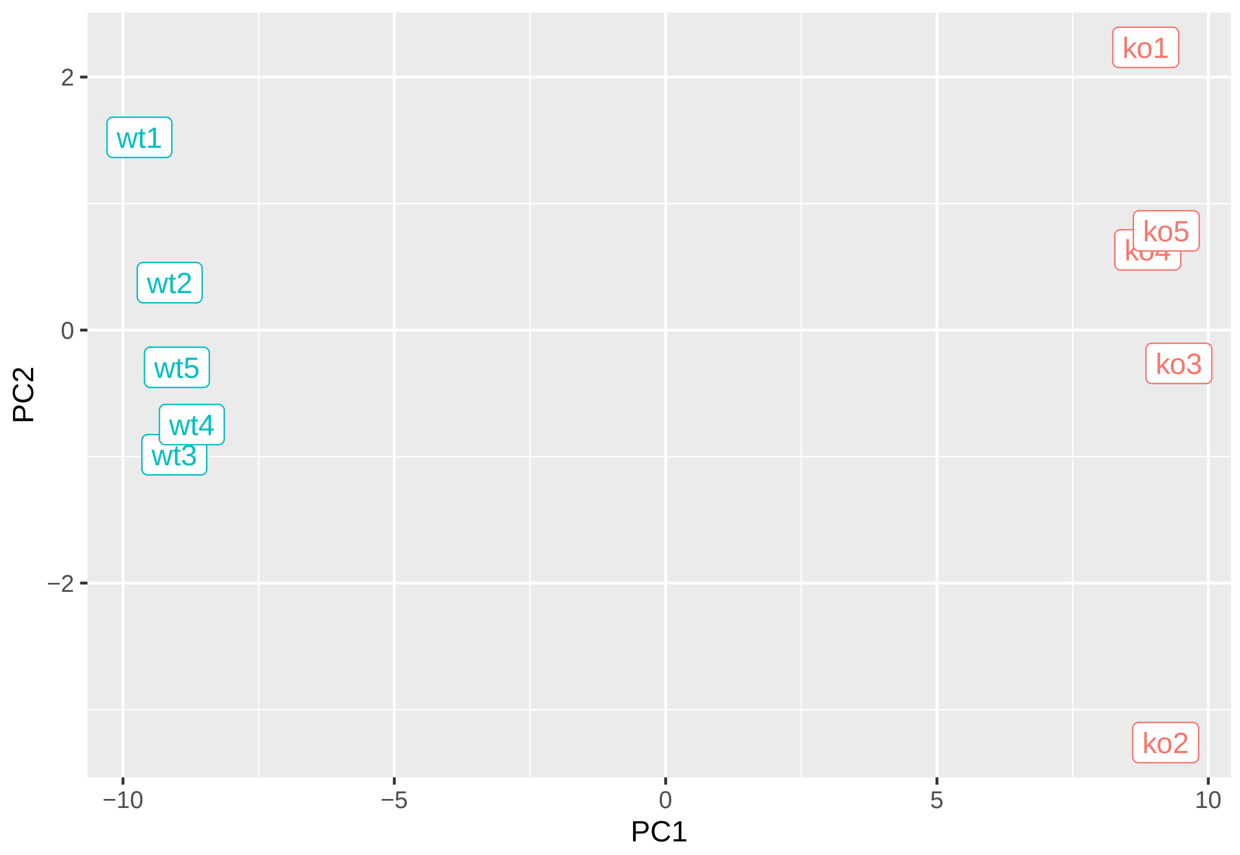
```
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

#then plot, add labels showing percent variance each pc accounts for and of each sample
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

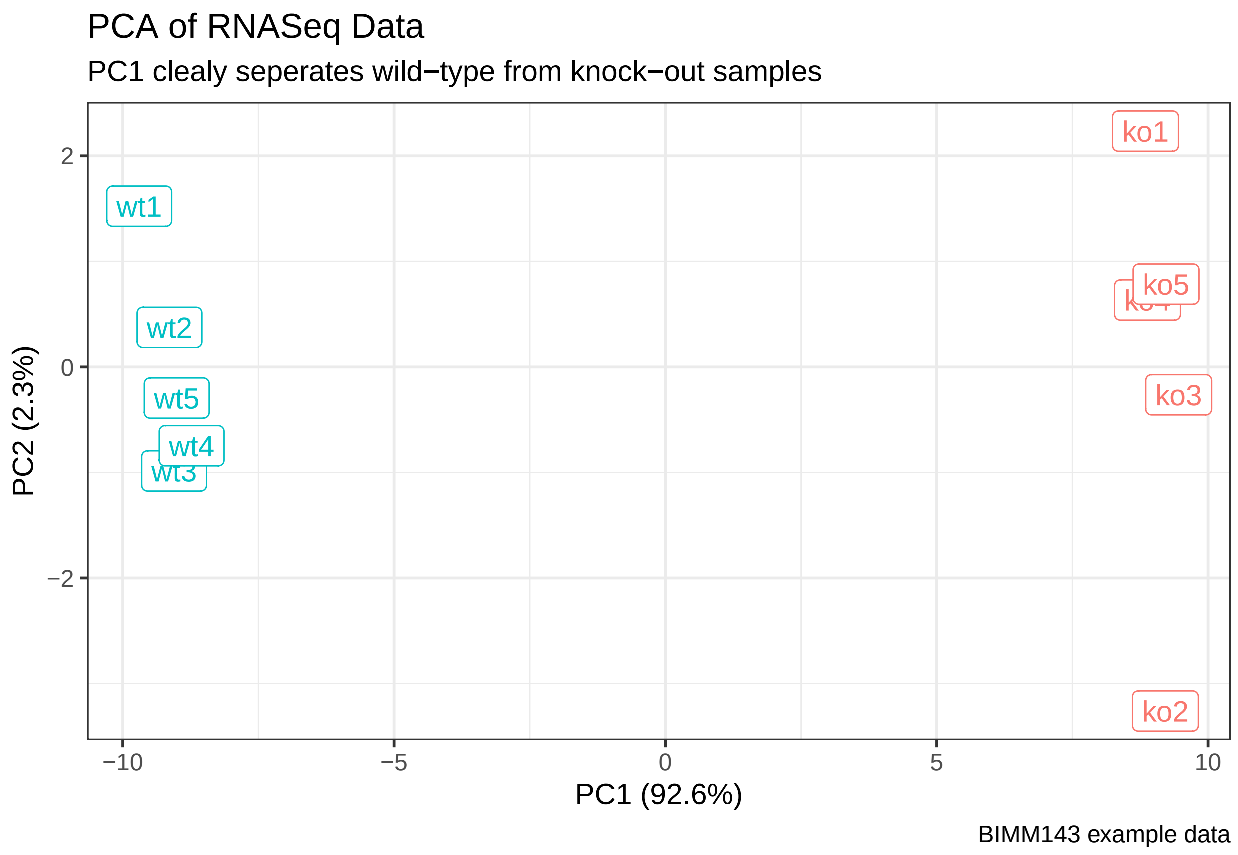
for ggplot, we need to turn our data into dataframe

```r
#need to first install and load ggplot (if not already installed)
library(ggplot2)

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



Add labels, make plot prettier

```
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()
```



We can find the top 10 genes that contribute to the most variation in PC1

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

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"
```