Machine Learning 01

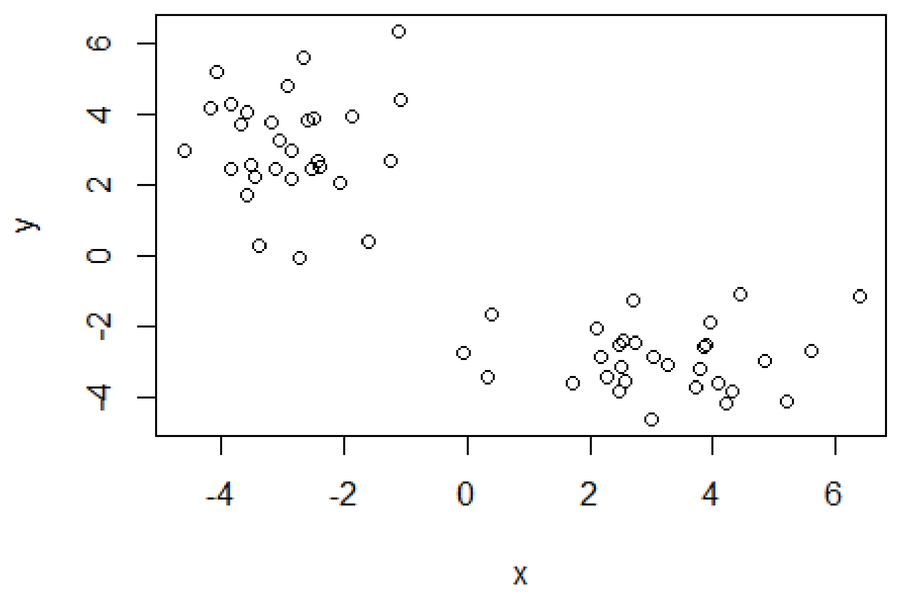
Meg Robinson

2/14/2022

Clustering Methods

First, create 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(), specifying # of clusters.

```
k = kmeans(data,centers=2,nstart=20)
k

## K-means clustering with 2 clusters of sizes 30, 30
##
## Cluster means:
##           x         y
## 1 -2.871373  3.151390
## 2  3.151390 -2.871373
```

```
##
## 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
##
## Within cluster sum of squares by cluster:
## [1] 87.8321 87.8321
##  (between_SS / total_SS =  86.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

QUESTION: How many points are in each cluster?

```
k$size

## [1] 30 30
```

ANSWER: 30 points in each cluster

QUESTION: What component of your result object details cluster size?

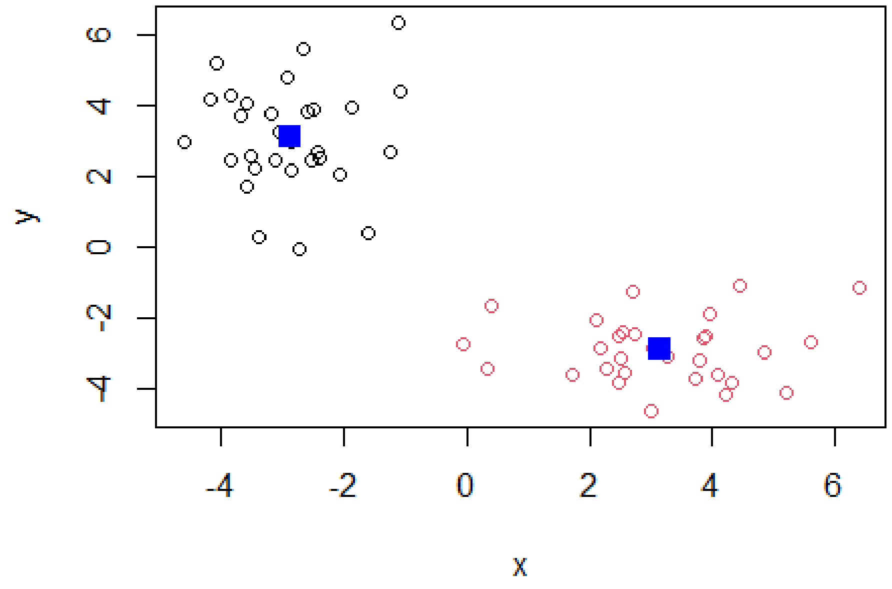
```
k$cluster

##  [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
```

QUESTION: What component of your result object details cluser center?

```
k$centers

##           x         y
## 1 -2.871373  3.151390
## 2  3.151390 -2.871373
```

QUESTION: Plot x colored by the kmeans cluster assignment and add cluster centers as blue points

```
plot(data, col=k$cluster)
points(k$centers, col='blue', pch=15, cex=1.5)
```

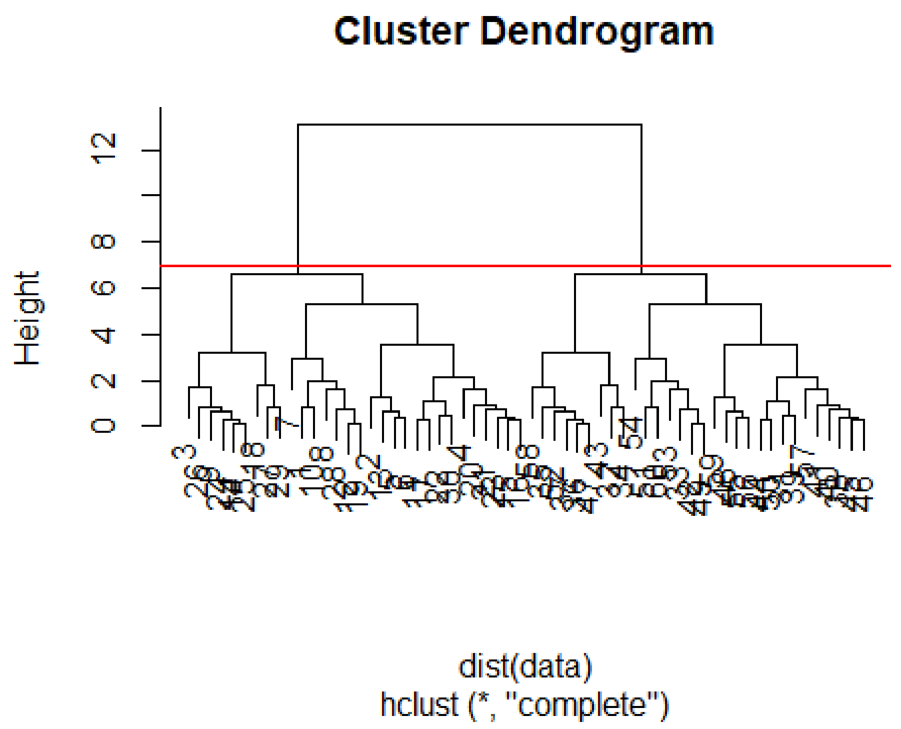


Hierarchical Clustering

```
h = hclust(dist(data))
h

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

Use the plot method:

```
plot(h)
abline(h=7,col='red')
```

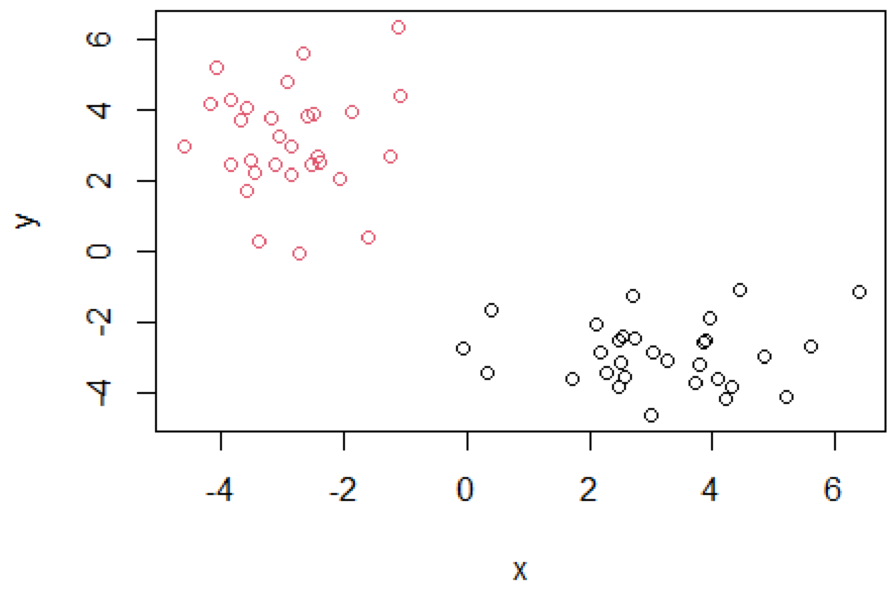


Find the membership vector using cutree()

```
cutree(h,h=7)

##  [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 1 1 1 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
```

And use this method to find number of k clusters

```
kclst = cutree(h, k=2)
plot(data,col=kclst)
```



We can see that kmeans() uses the data and number of centers, while hclust() uses the distance of the data.

PCA of UK food data

import data

```
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?

Complete the following code to find out how many rows and columns are in x?

___(x)

```
dim(x)

## [1] 17  5
```

Preview the first 6 rows

```
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
```

Row titles are incorrectly being stored as a column. Fix this.

```
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

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
```

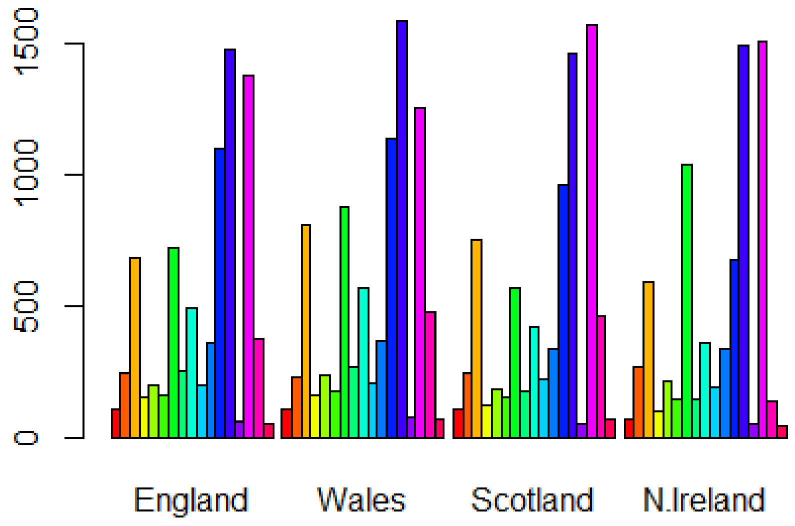
Now there are correctly 17 rows and 4 columns, yay!

> 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?

The second approach, read.csv() is way better than the first one in general because it loads the data without manipulation. The first approach, indexing x[,-1], removes data & will continue to remove the last column every time it is run.

Spotting major differences and trends

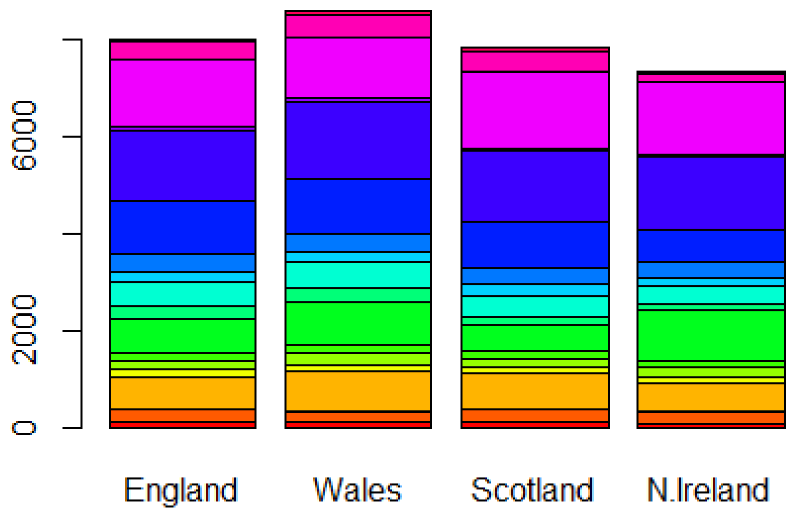
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



Q3: Changing what optional argument in the above barplot() function results in the following plot?

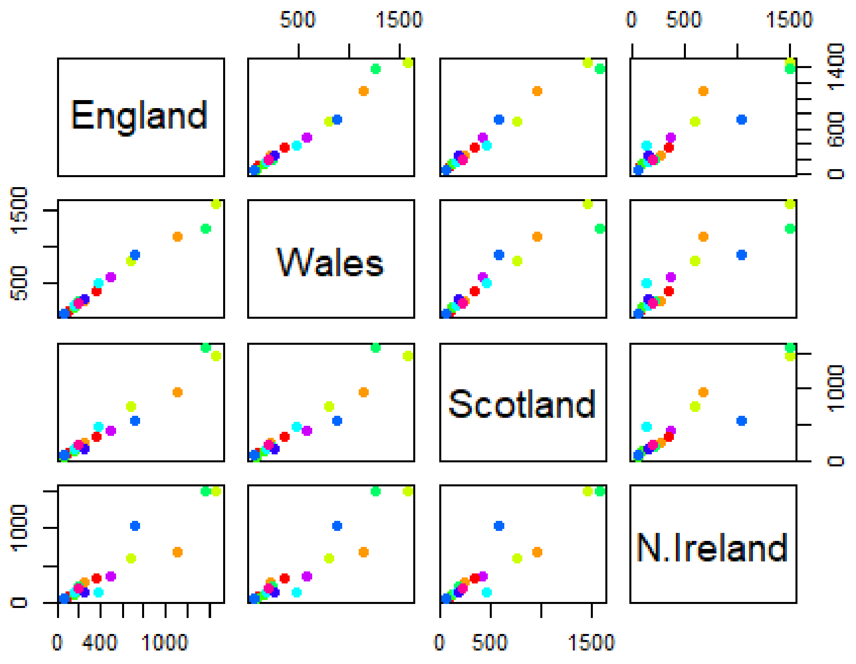
We can remove the beside=TRUE argument to obtain the following:

```
barplot(as.matrix(x), 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)
```



The y axis for each row of plots is the country name in that row, whereas the x axis for each column of the plots is the country name in that column. Therefore, the diagonal plots of this graph indicates that the country is the same on the row and column so there is no way to plot a pairwise interaction.

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

N. Ireland appears to be more unique as it contains the most values that deviate from the diagonal when compared to the other countries.

PCA to the rescue

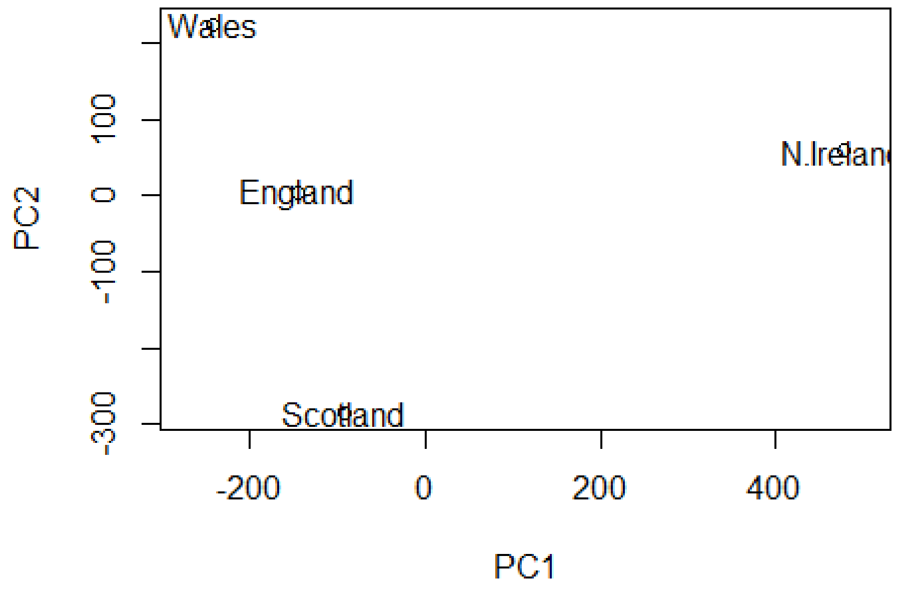
Use the prcomp() PCA function

```
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.

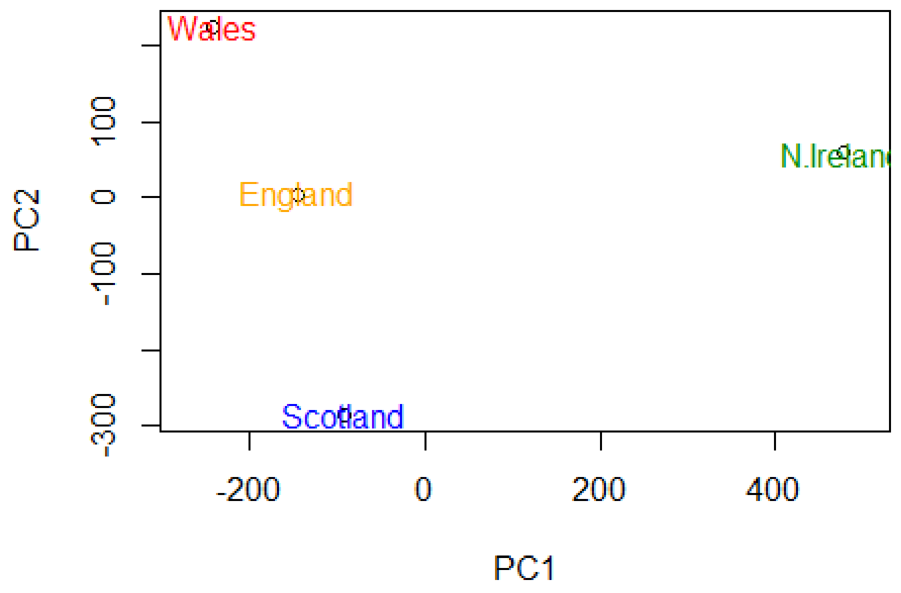
Plot PC1 vs PC2

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

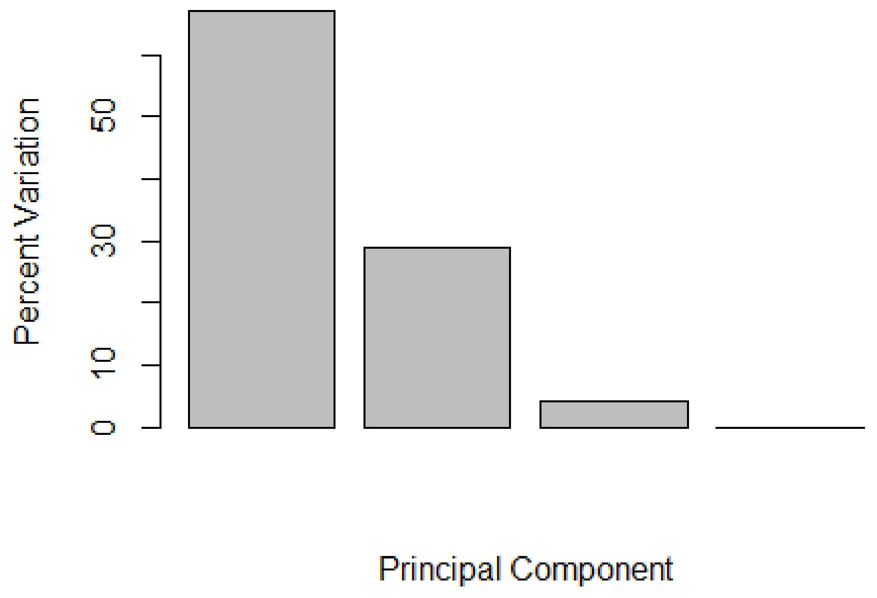


Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
mycols = c("orange", "red", "blue", "green4")
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x), col=mycols)
```

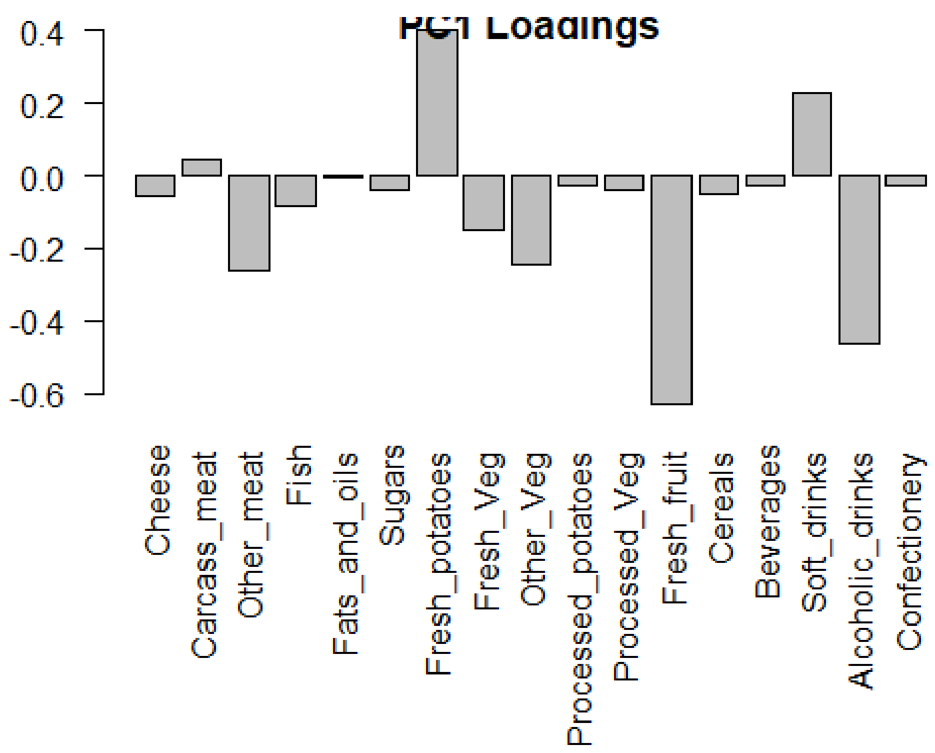


```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v

## [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")
```



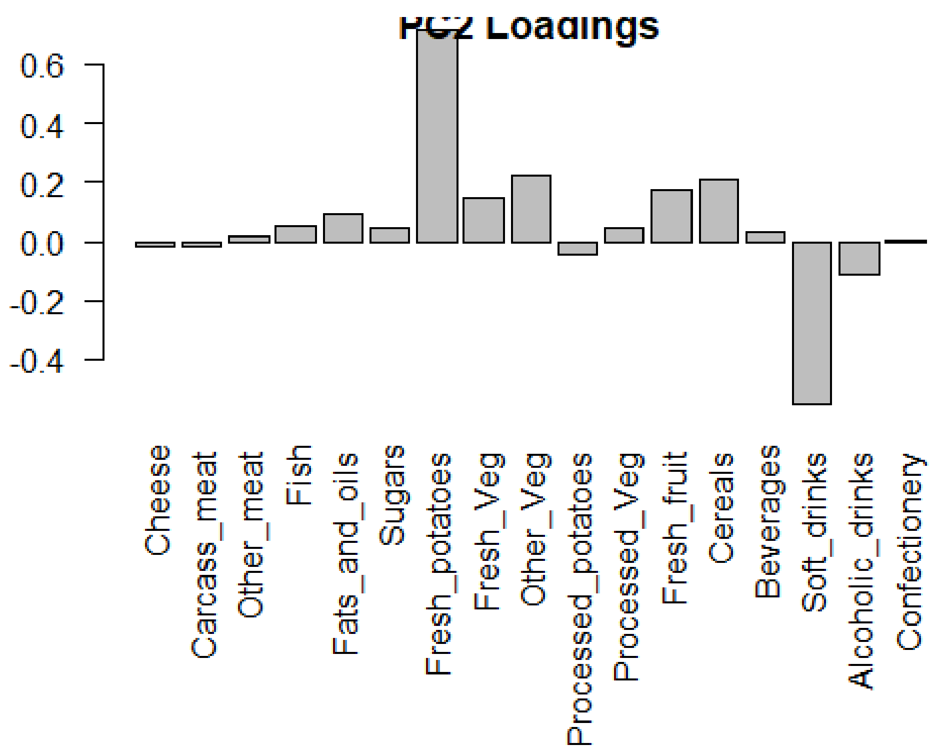
> Digging deeper (variable loadings):

```
## 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, main="PC1 Loadings" )
```



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

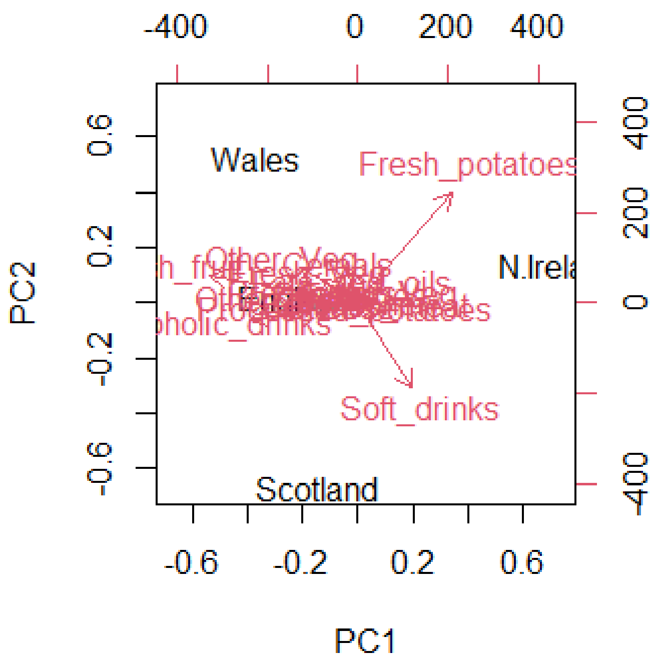
```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2, main="PC2 Loadings" )
```



From looking at this graph, it looks like 'fresh potatoes' and 'soft drinks' feature prominently. The plot also tells us which groups contribute the most to the remaining 10% variance, after PC1.

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?

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

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

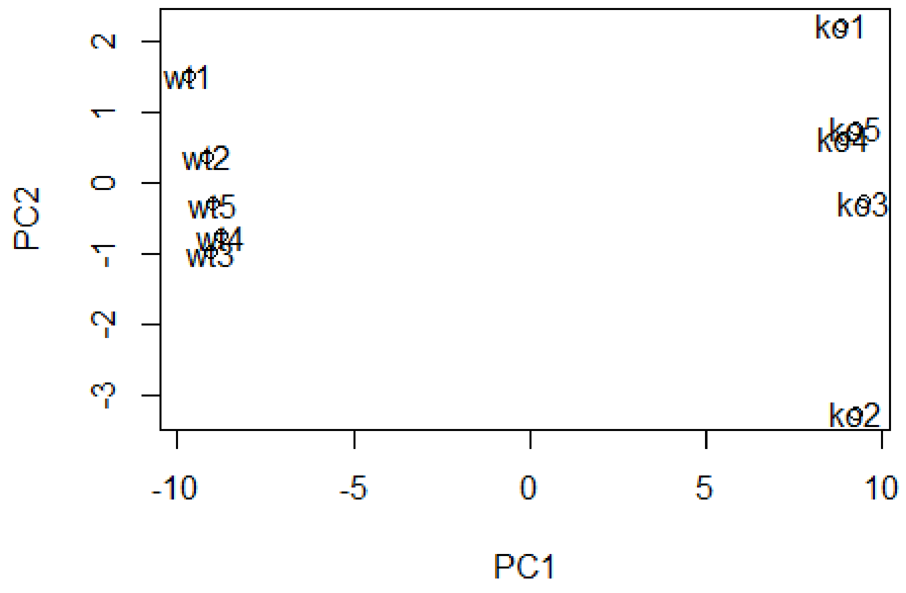
```
ncol(rna.data)
```

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

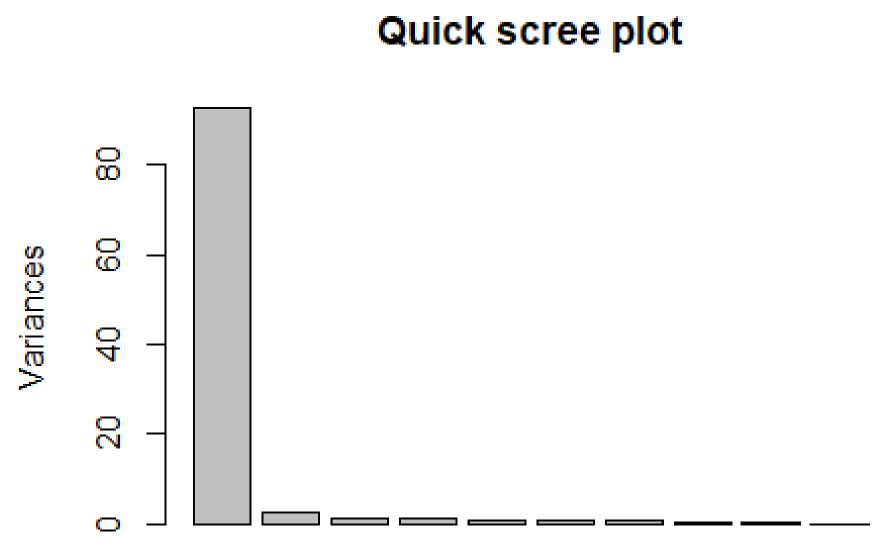
There are 100 genes and 10 samples.

```
## 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")

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

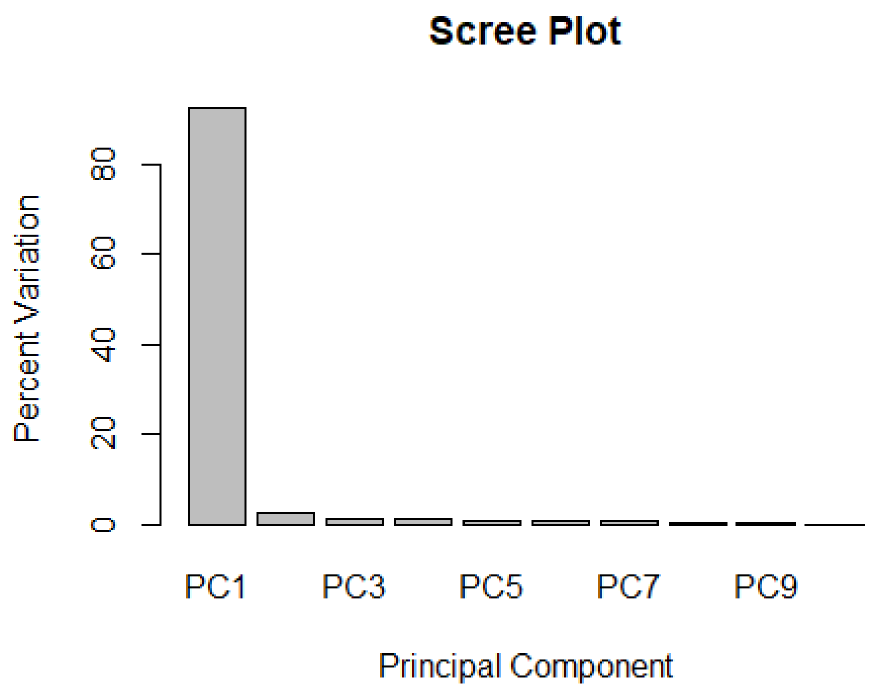


```
summary(pca)

## 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")
```



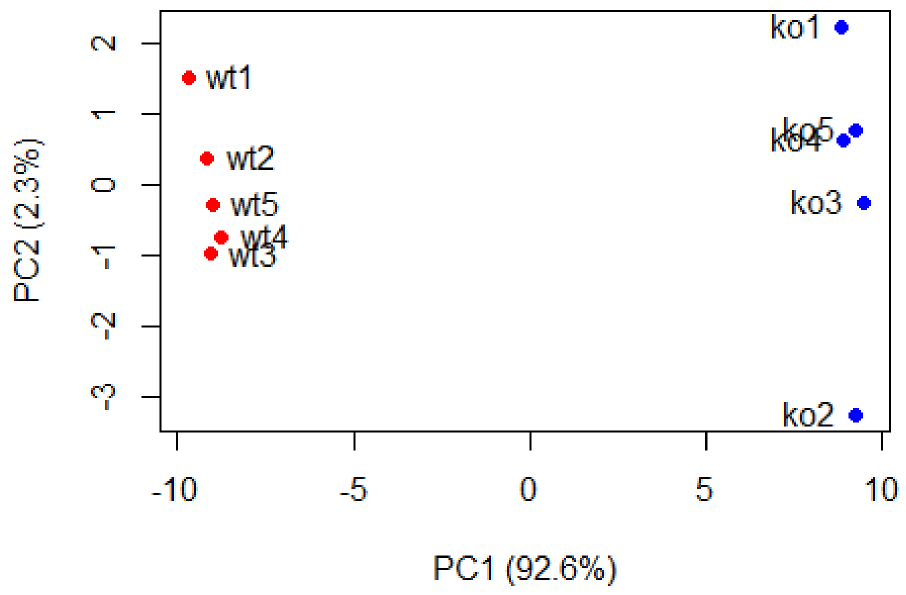
Lets make our own scree plots:

```
## 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")
```



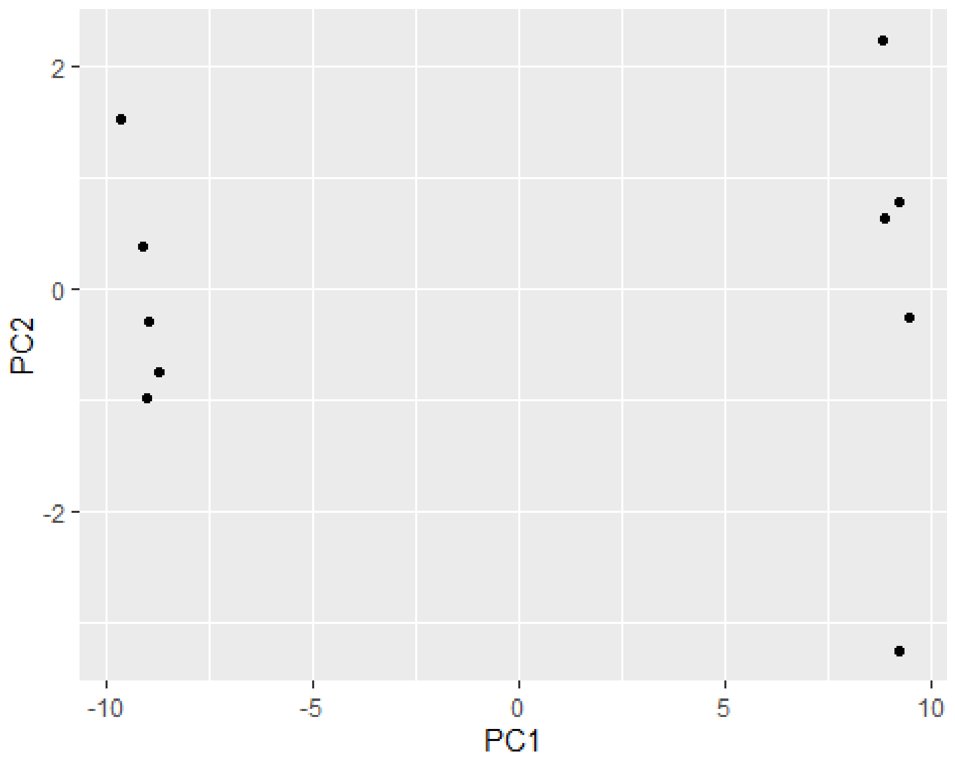
Make it more attractive and useful:

```
## 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)))
```

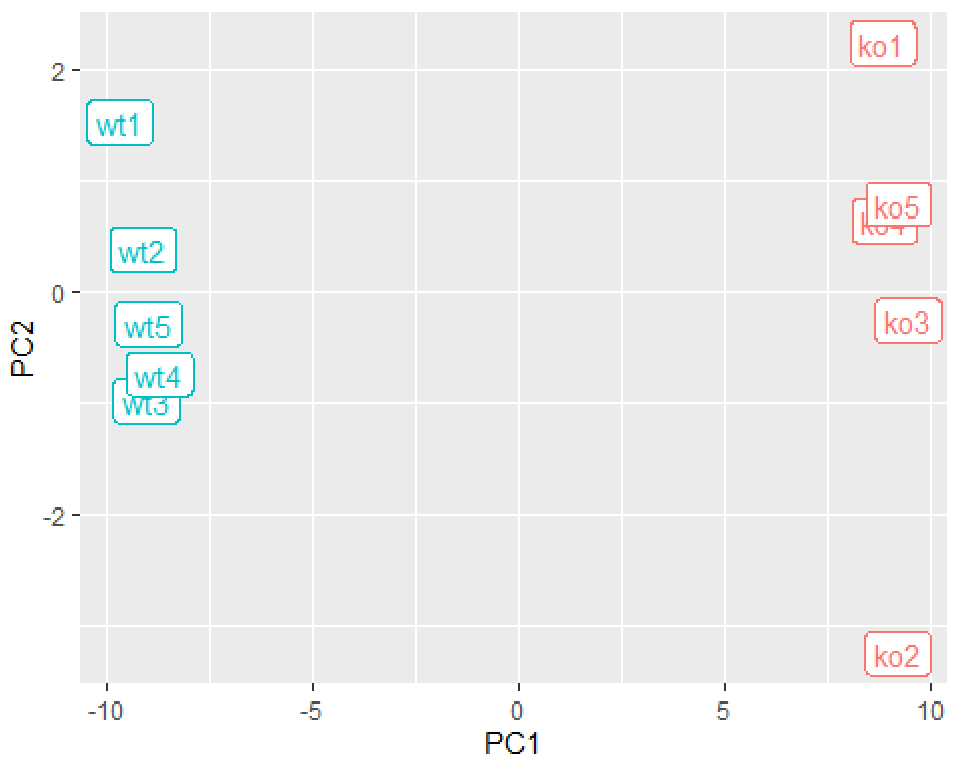


Using ggplot

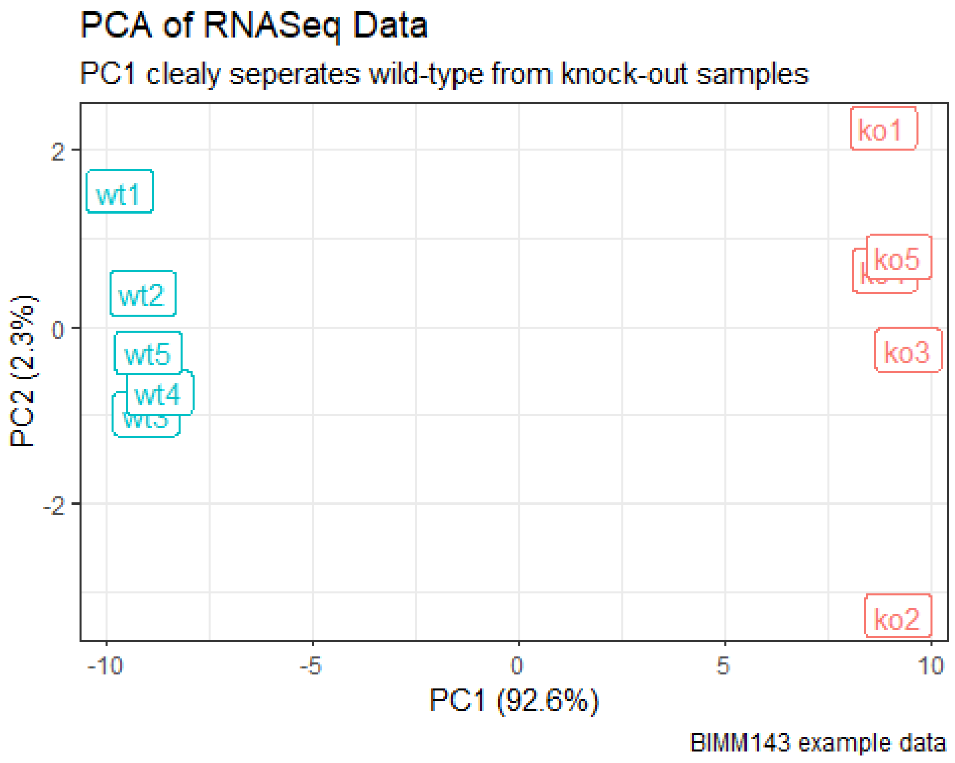
```
library(ggplot2)

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

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



```
# 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
```



```
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()
```


PCA of RNASeq Data
PC1 clealy seperates wild-type from knock-out samples
ko1
wt1
ko5
ko4
wt2
ko3
wt5
wt4
wt3
ko2
2
0
-2
-10
-5
0
5
10
PC2 (2.3%)
PC1 (92.6%)
BIMM143 example data