Lab8_Machine_learning

Vivian Cai

2/11/2022

1. PCA of UK food data

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

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

```
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
```

```
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
```

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

17 rows and 4 columns, we could use many functions. nrow(), ncol(), dim(), or even str() would give us the information. In the chunck above, I used dim().

```
# a better/more robust way to assign row names
x <- read.csv(url, row.names=1)
head(x)
```

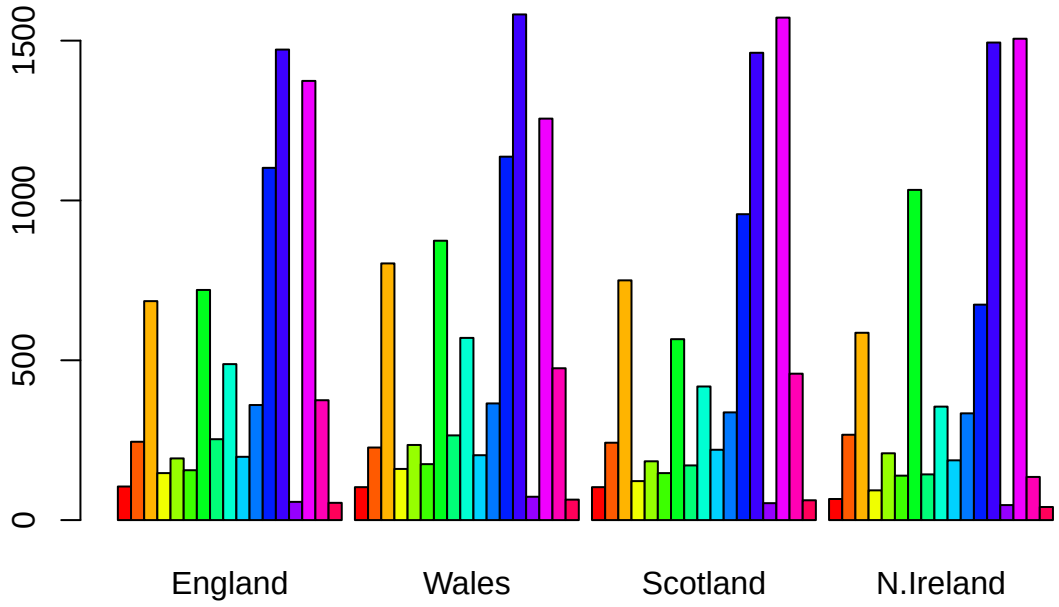
```
##                England Wales Scotland N.Ireland
## Cheese             105   103      103        66
## Carcass_meat       245   227      242       267
## Other_meat         685   803      750       586
## Fish               147   160      122        93
## Fats_and_oils      193   235      184       209
## Sugars             156   175      147       139
```

```
# when the x <- x[,-1] is called multiple times, we kept taking away columns, which might contain data
```

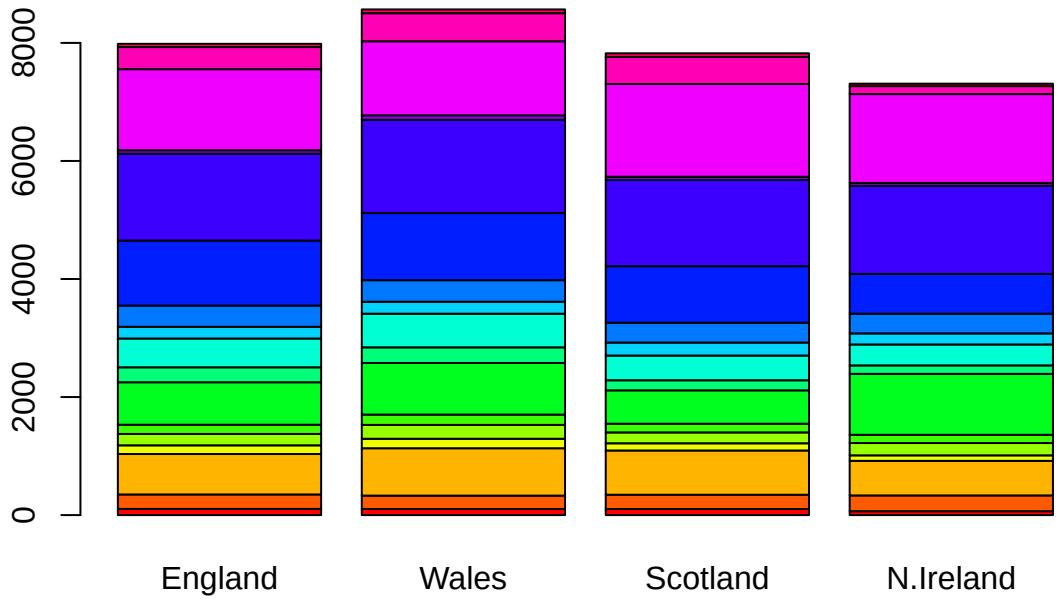
Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

The second approach is better because if the first cunck of code is ran multiple times, we would remove more data than just the row names.

```
# Generating regular bar-plots of the data
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```




```
# Generating stacked bar-plots of the data
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



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

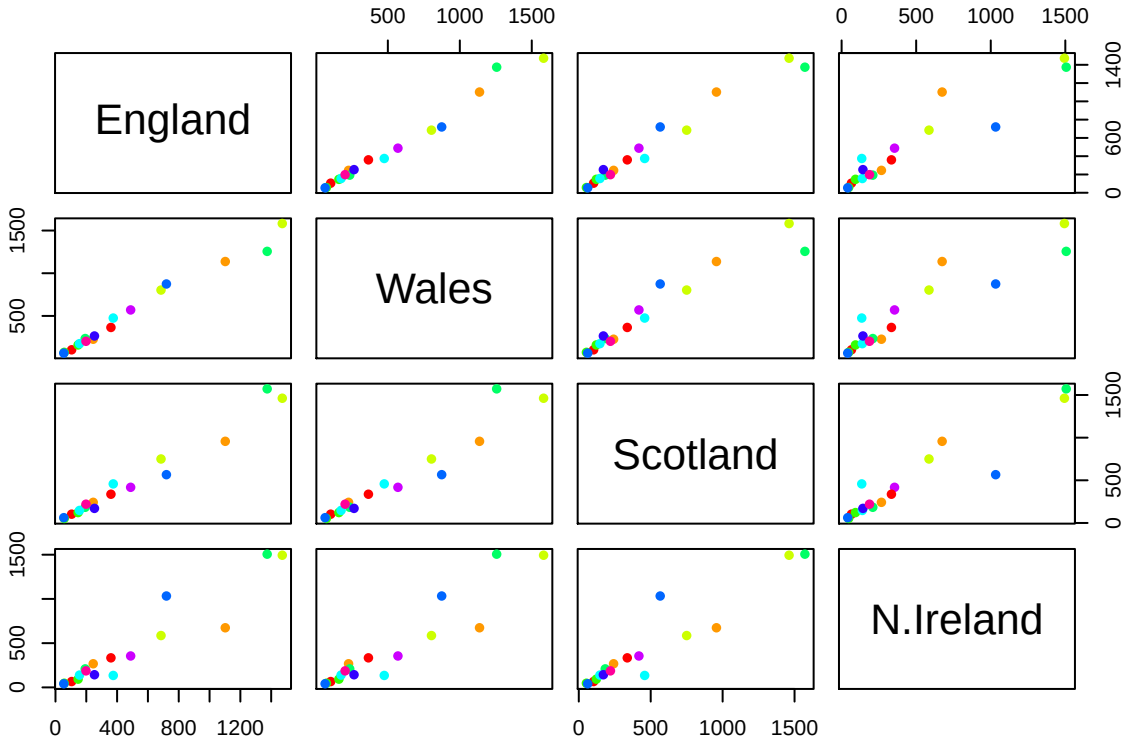
I changed the beside argument to F (FALSE).

Where is Q4???

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?

Landing on a diagonal means the two countries has very similar consumption of the corresponding type of food.

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




```
# The pairs function take data in x and plot the 17 variables between every two country. pch=16 specifi
```

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

Based on the scatter plots, N. Ireland seem to have the least number of points on the diagonal. This means N. Ireland has the most dissimilar food consumption pattern in among other countries of the UK.

Note to self: prcomp() expects the observations to be rows and the variables to be columns therefore we need to first transpose our data.frame matrix with the t() transpose function.

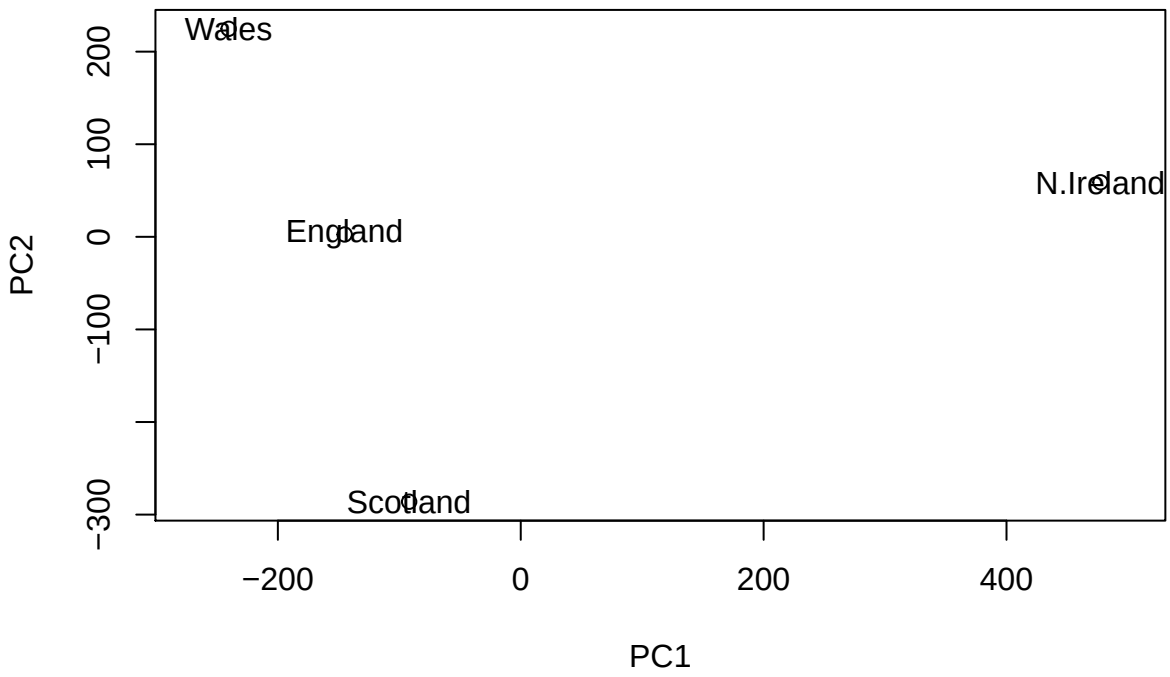
```
# Use the prcomp() PCA function
pca <- prcomp( t(x) )
summary(pca)
```

```
## 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
```

Note to self: PC1 is equivalent to the axis obtained through finding the (least-squares) line of best fit through the plotted data where it has the largest spread. The second best axis PC2, the third best PC3 etc.

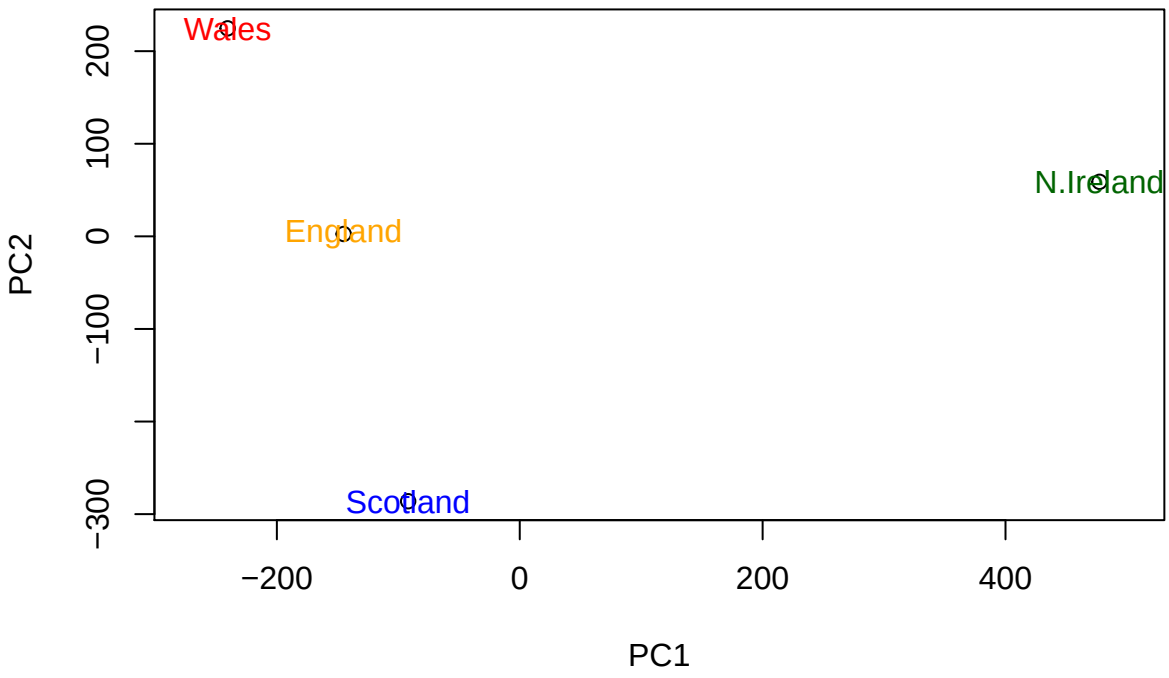
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
# 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.

```
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 = c("orange", "red", "blue", "dark green"))
```




```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
# square of pca$sdev , which stands for "standard deviation"

v
```

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

In practice, it is usually sufficient to include enough principal components so that somewhere in the region of 70% of the variation in the data is accounted for.

```
z <- summary(pca)
z
```

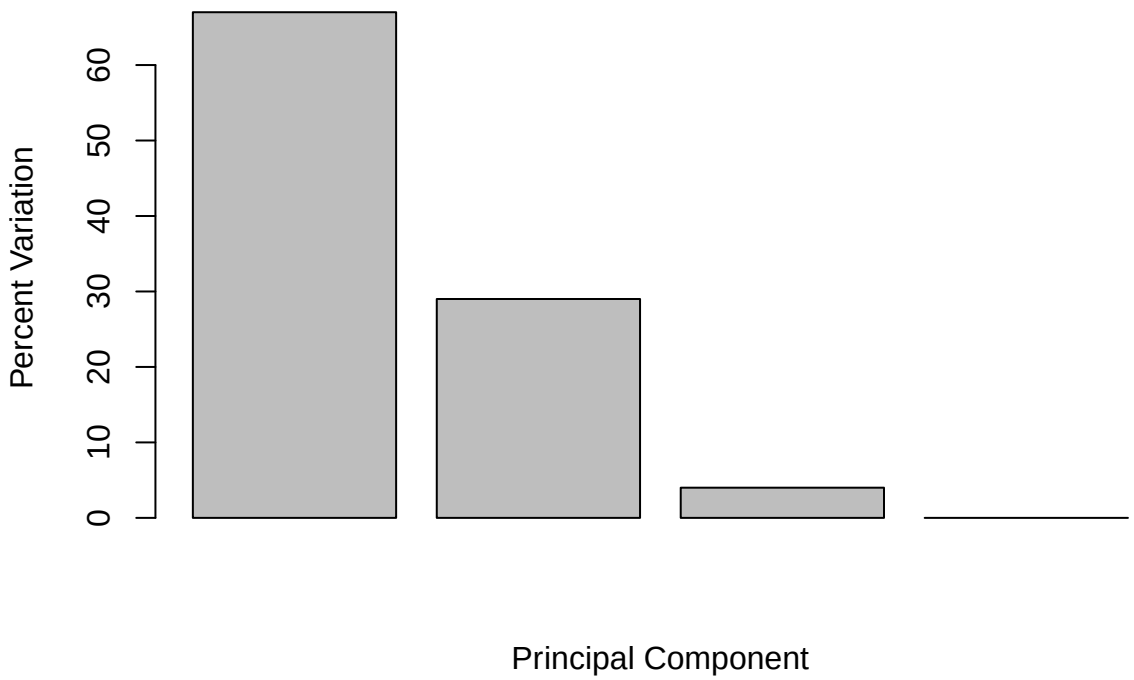
```
## Importance of components:
##                             PC1      PC2      PC3       PC4
## Standard deviation     324.1502 212.7478 73.87622 4.189e-14
## Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
## Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

```
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
```

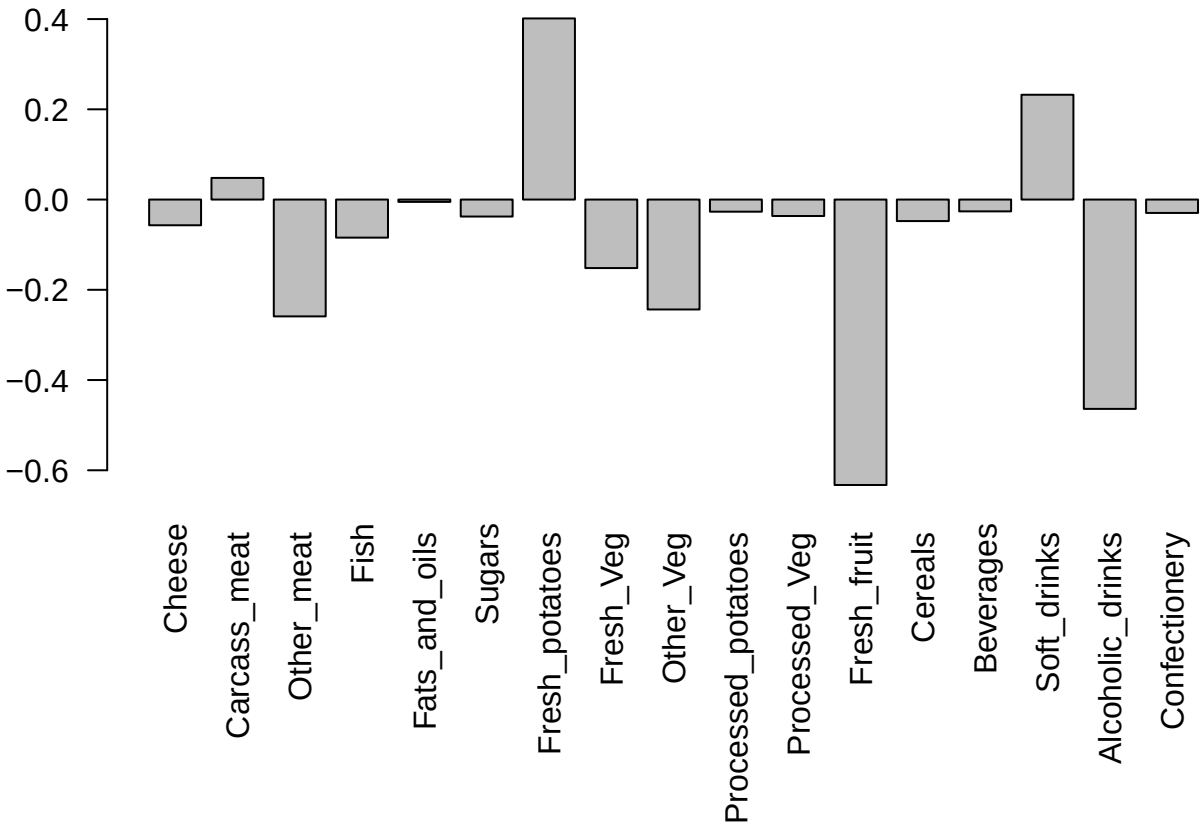
This information can be summarized in a plot of the variances (eigenvalues) with respect to the principal component number (eigenvector number), which is given below.

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



We can also consider the influence of each of the original variables upon the principal components (typically known as loading scores). This information can be obtained from the **prcomp() returned $rotation component**. It can also be summarized with a **call to biplot()**, see below:

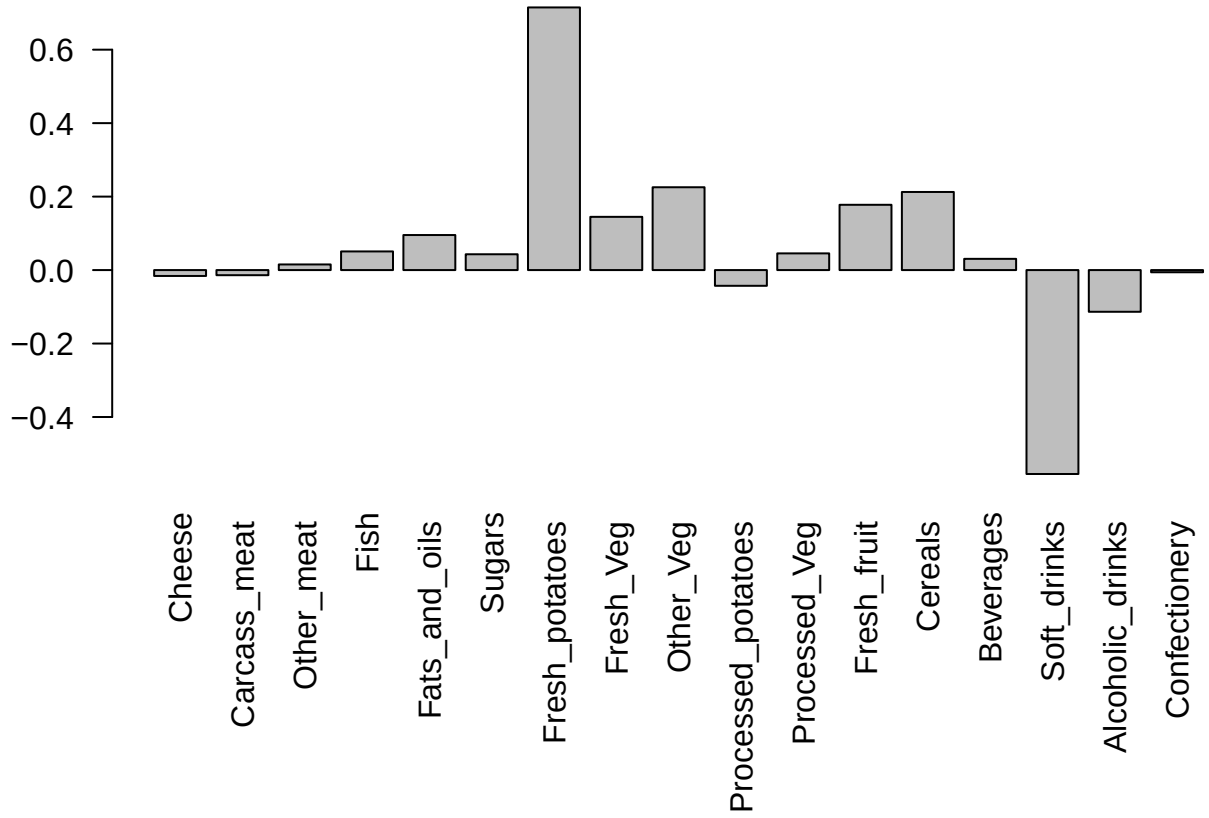
```
# Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



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

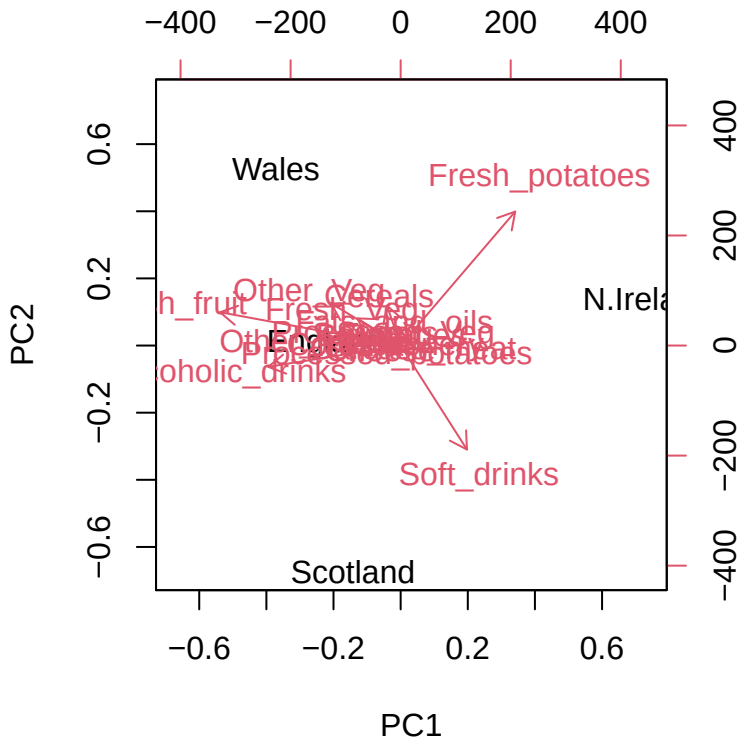
Fresh potatoes and soft drinks feature prominantely. It tells us that the differences between the 3 UK countries (Wales, England, and Scotland) are mainly caused by differences in fresh potato and soft drink consumptions.

```
par(mar=c(10, 3, 0.35, 0)) # is this defining the margins?
barplot( pca$rotation[,2], las=2 )
```



Let's also try biplots:

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



2. 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
```

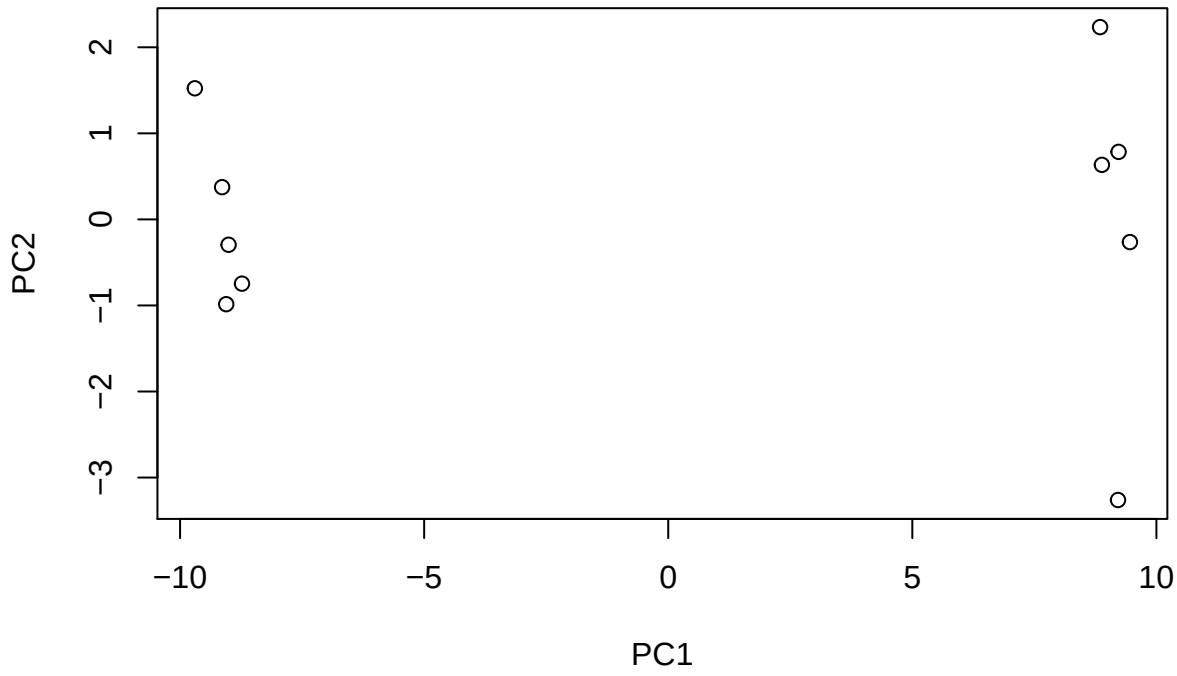
Q10: How many genes and samples are in this data set?

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

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

100 genes and 10 samples

```r
# take the transpose of our data
pca_rna <- prcomp(t(rna.data), scale=TRUE)

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

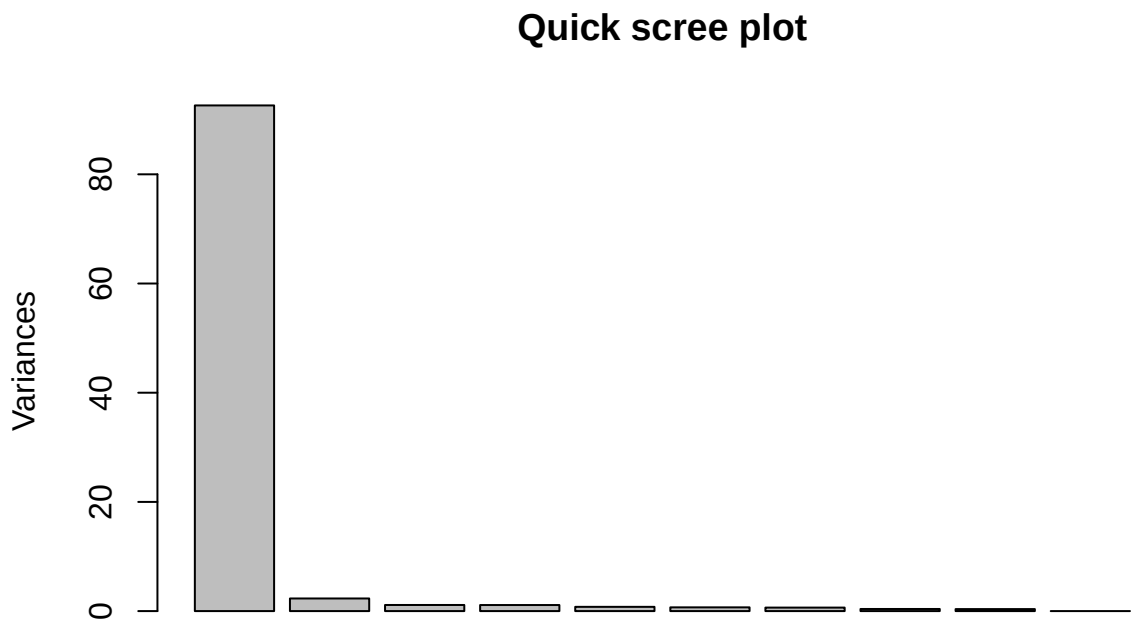



```r
summary(pca_rna)
```

```
## 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 3.348e-15
## Proportion of Variance 0.00385 0.00364 0.000e+00
## Cumulative Proportion  0.99636 1.00000 1.000e+00
```

PC1 accounts for 92.62% of the variance!

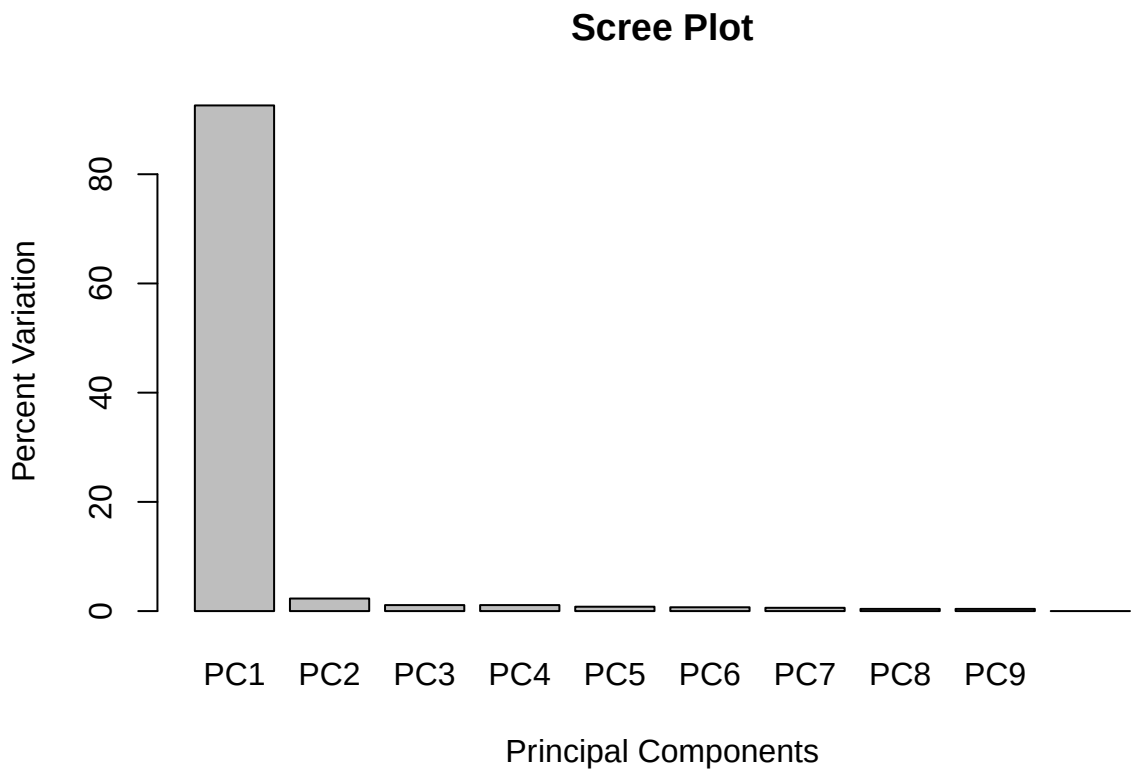
```r
plot(pca_rna, main="Quick scree plot")
```



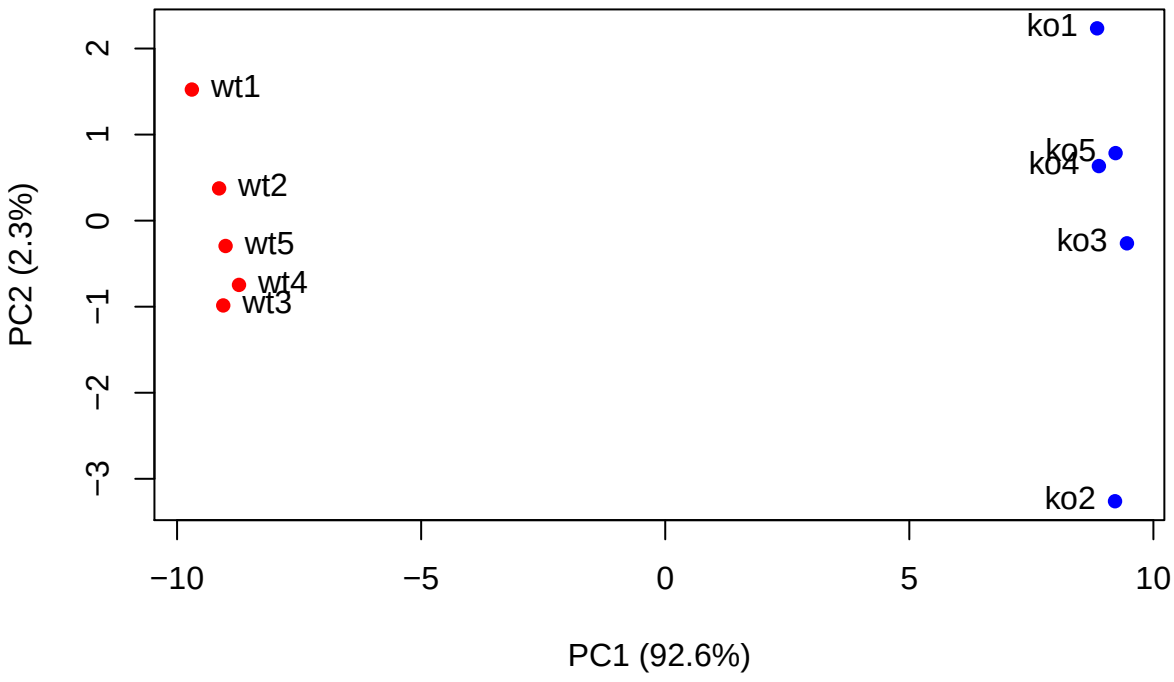

```
# Variance captured per PC
pca.var <- pca_rna$sdev^2

# 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
```

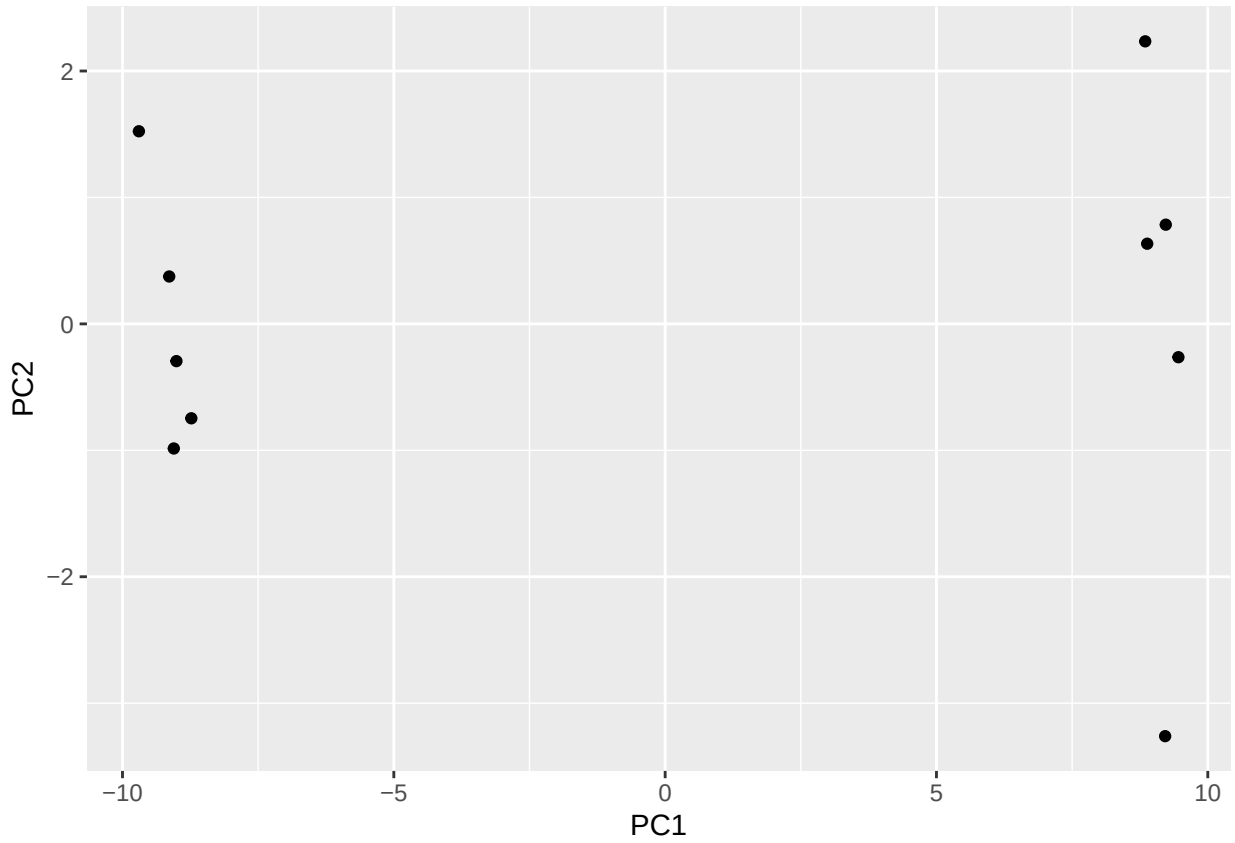
```
barplot(pca.var.per, main="Scree Plot",
        names.arg = paste0("PC", 1:10),
        xlab="Principal Components", ylab="Percent Variation")
```



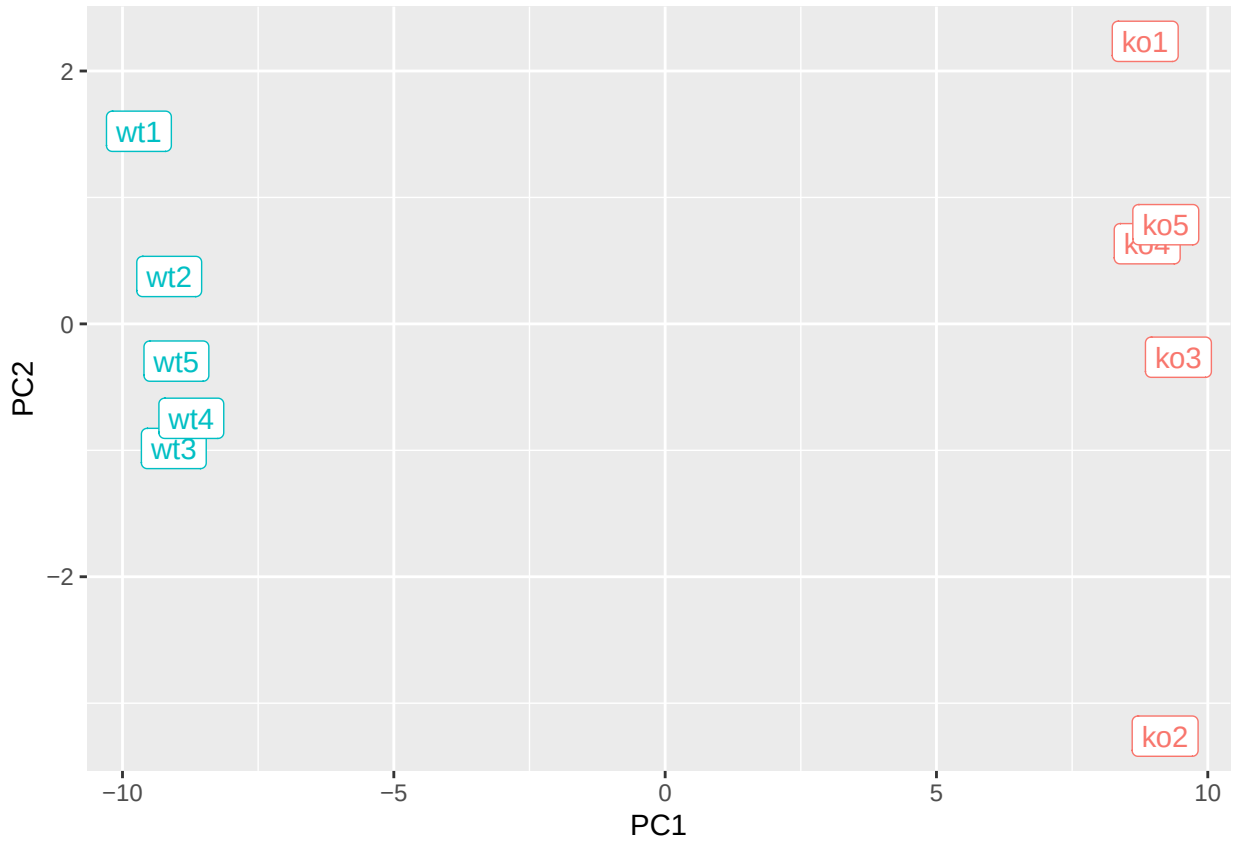

```
# A vector of colors for wt and ko samples
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

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

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



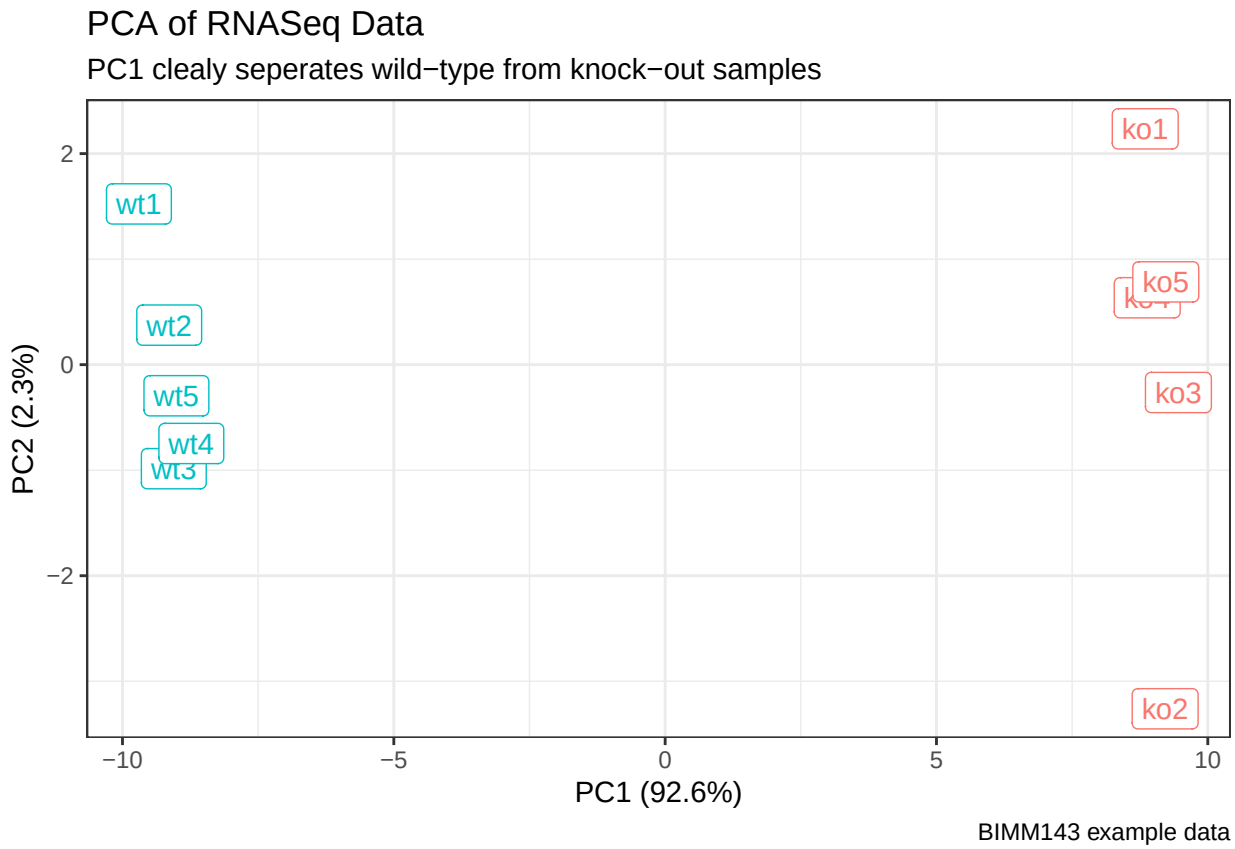

```
library(ggplot2)

df <- as.data.frame(pca_rna$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()
```




```
loading_scores <- pca_rna$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"
```