Lab 8: Introduction to machine learning for Bioinformatics

Caroline Mackey (A15522472)

10/24/2021

1. PCA of UK food data

Part A: Data Import

First, import data from provided link.

```
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?

```
dim(x)
```

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

> **A1.** There are 17 columns and 5 rows.The dim() R function was used to answer these questions, as shown above.

Part B: Checking My Data

Use head() function to check data.

```
# 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
```

Correct row names so that column 1 values are the row names.

```
# 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 dimensions

```
dim(x)
```

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

Alternatively, correct row-names when reading the data file.

```
# Using variable y to show that it is different from our previous attempt of using variable x.
url <- "https://tinyurl.com/UK-foods"
y <- read.csv(url, row.names=1)
head(y)
```

```
##                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
```

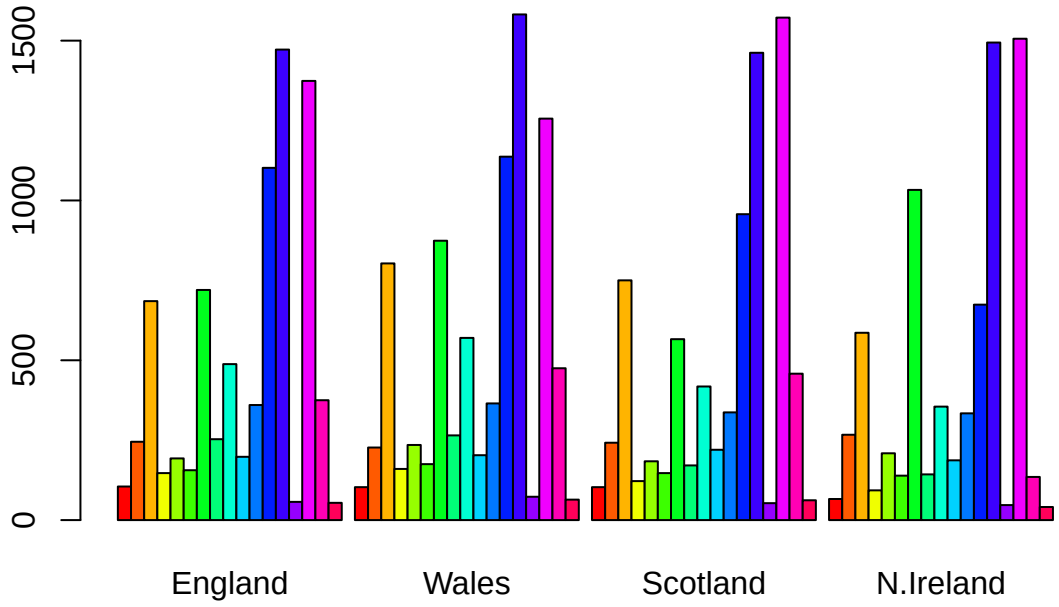
> **Q2.** Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

> **A2.** I prefer using the second approach of solving the 'row-names problem,' since it corrects the problem as we call the data set. Since we're correcting the problem as we call teh data set, it will be correct each time we use it. This requires fewer lines of code, and makes our code more robust.

Part C: Spotting major differences & trends

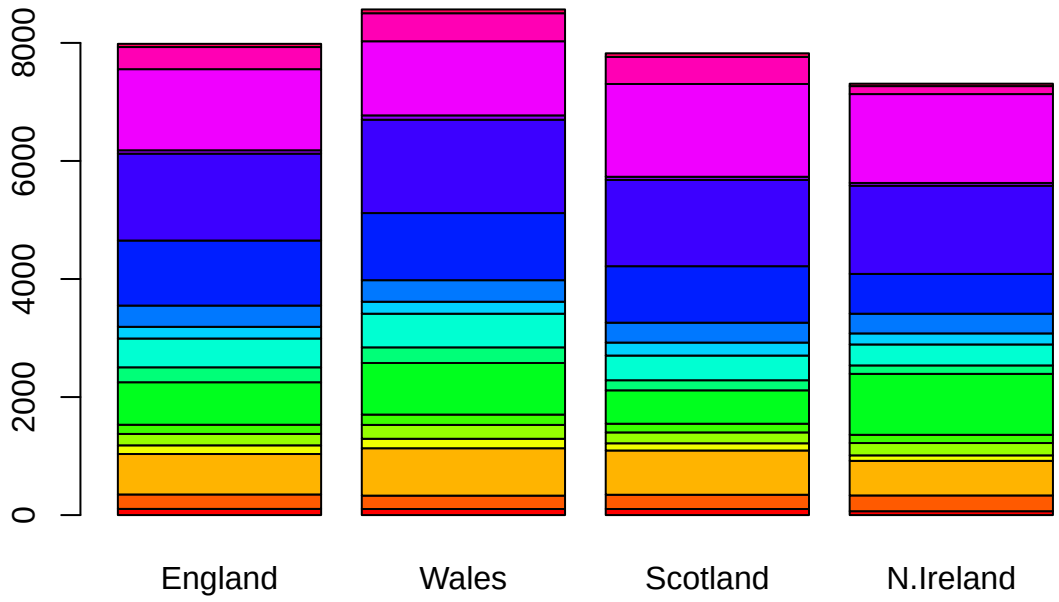
Generate a preliminary barplot to visualize the data.

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



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

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

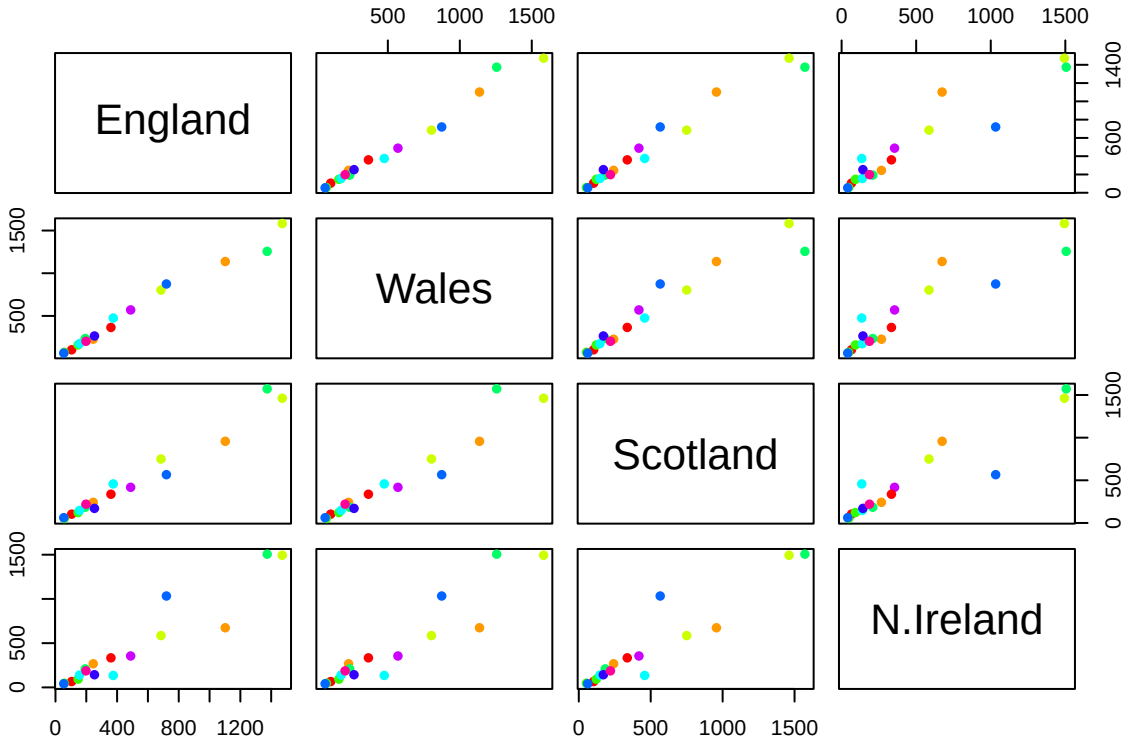


A3. As shown above, setting beside to F rather than T created the desired bar chart.

Q4. Question 4 was not included on our lab worksheet...

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,
      )
```



A5. The graphs above show all possible combinations of our 4 countries. The points on the plot represent our 17 categories of foods, with the x coordinate representing the consumption in the first country, and the y coordinate representing the consumption on the second country. If a point lies on the diagonal for a given plot, it suggests that people in those 2 countries eat a similar amount of that type of food.

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

A6. N. Ireland has a couple of outliers that the other three countries do not have. On the graph, the most significant outliers appear to be the blue and orange colored data points, as they don't fall as close to the diagonal line. Right now, we cannot easily tell which food categories these data points represent.

Part D: PCA to the Rescue

Perform PCA.

```
# Use the prcomp() PCA function.
# Transform x so that observations are rows and variables are be columns.
pca <- prcomp( t(x) )
summary (pca)
```

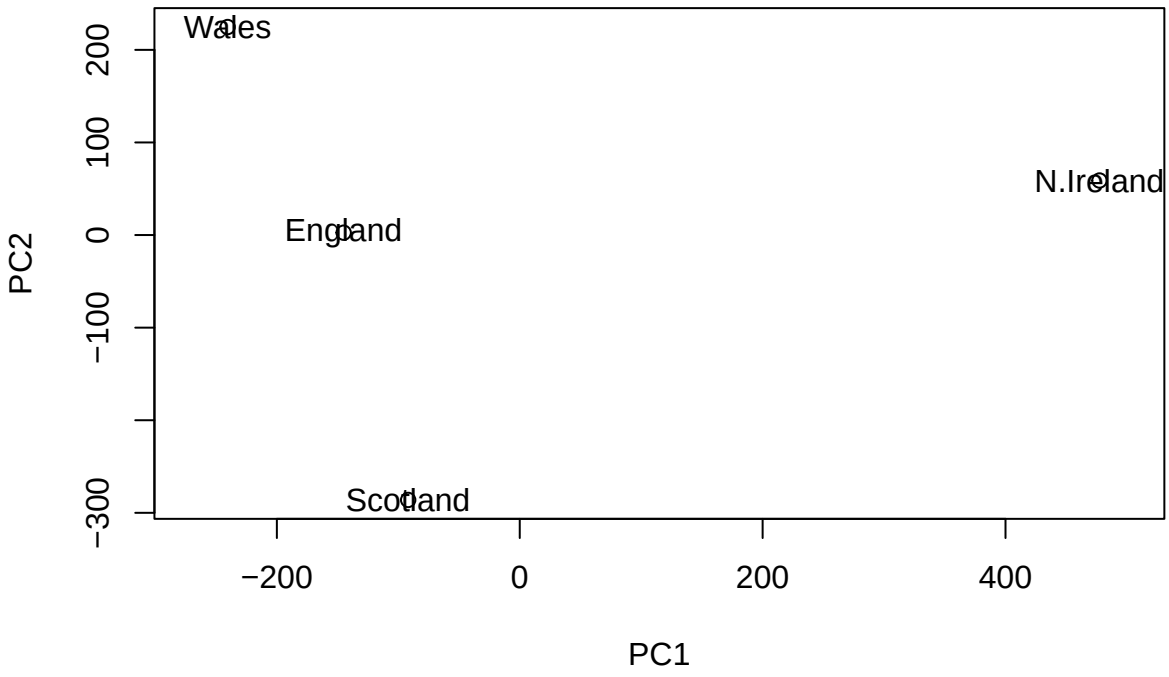
```
## Importance of components:
```

```
##                             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.

A7. See below.

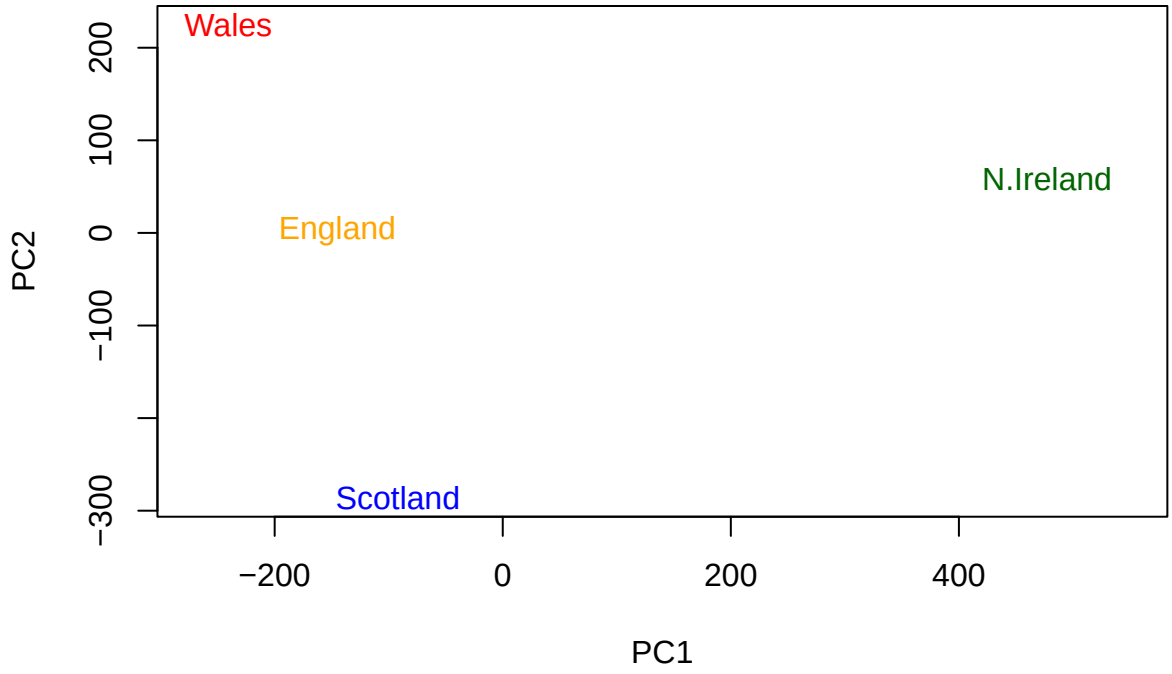
```
# Plot PC1 vs PC2
plot (pca$x[,1], pca$x[,2],
      xlab="PC1", ylab="PC2",
      xlim=c(-270,500))
text (pca$x[,1], pca$x[,2],
      colnames(x))
```



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.

A8. See below.

```
# Plot PC1 vs PC2 with customized colors.

# Make vector of desired colors.
country_cols <- c("orange", "red", "blue", "dark green")

plot (pca$x[,1], pca$x[,2],
      xlab="PC1", ylab="PC2",
      # Change xlim so that N. Ireland text isn't cut off:
      xlim=c(-270,550),
      # Remove data point dots to make text more readable:
      col=NA)
text (pca$x[,1], pca$x[,2],
      colnames(x),
      # Assign colors using vector from earlier.
      col=country_cols)
```



First, calculate the amount of variation in the original data each PC accounts for using square of pca$sdev (standard deviation).

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

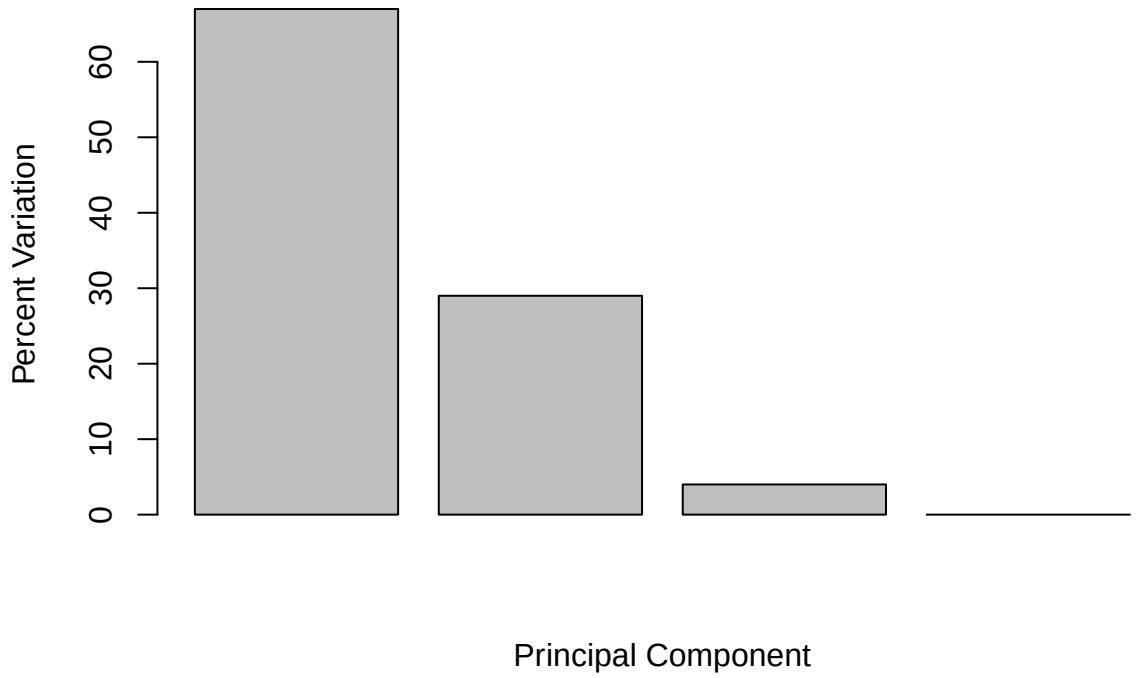
```
## [1] 67 29  4  0
```

```
# 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
```

Summarize in a plot of the variances (eigenvalues) with respect to the principal component number (eigenvector number).

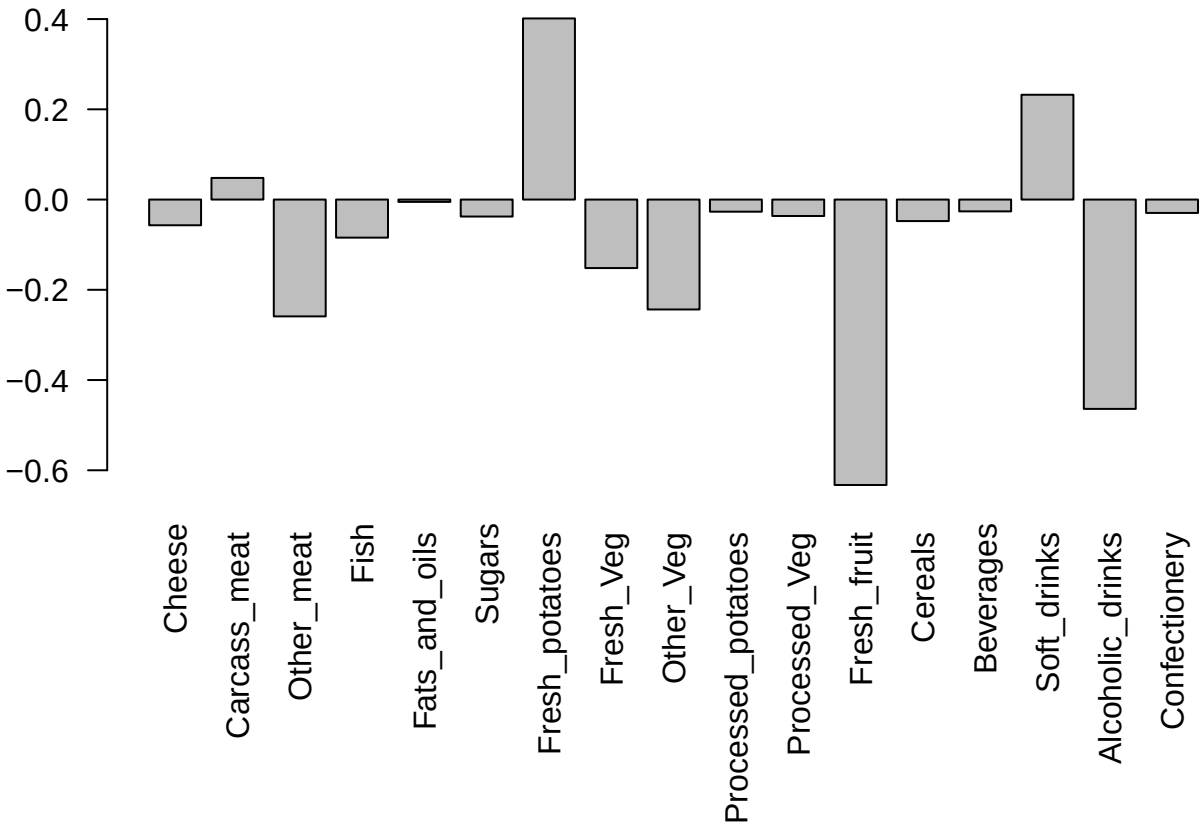
```
barplot (v,
         xlab="Principal Component",
         ylab="Percent Variation")
```



Part E: Digging deeper (variable loadings)

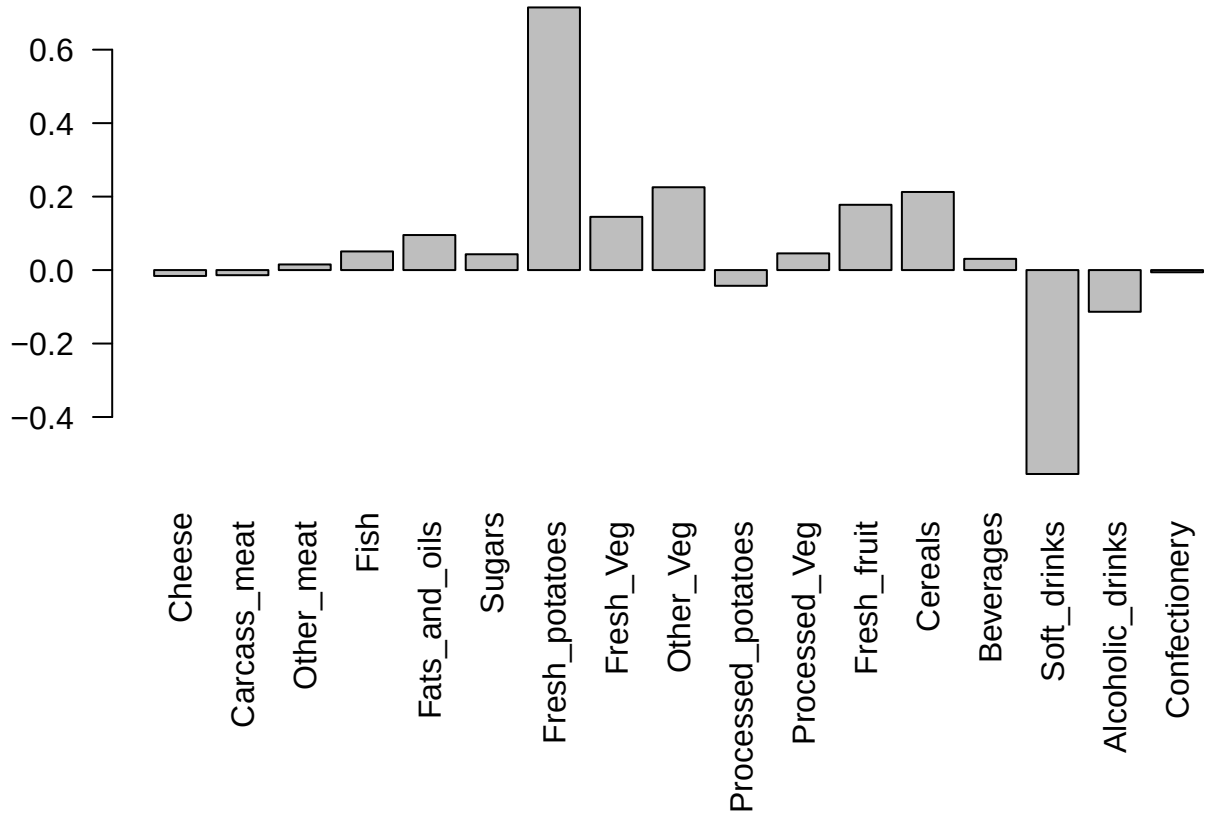
Look at loading scores.

```
# Let's focus on PC1 as it accounts for >90% of variation.
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?

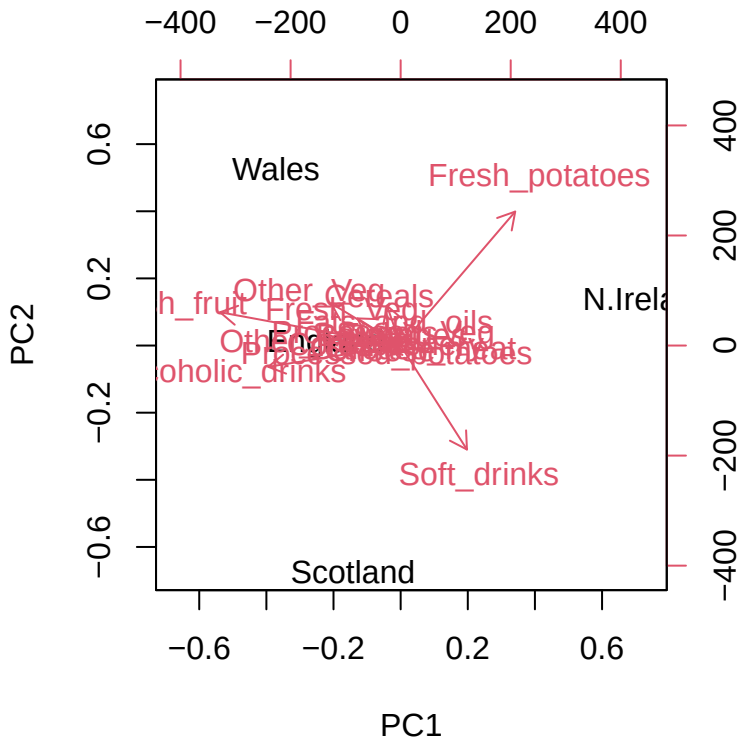
```
par (mar=c(10,3,0.35,0))
barplot (pca$rotation[,2], las=2)
```



A9. See the loading plot for PC2 above. Fresh potatoes and soft drinks feature prominently, with fresh potatoes being positive, or consumed more, and soft drinks being negative, or consumed less.

Part F: Biplots

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



2. PCA of RNA-seq data

Part A

Download data from provided link.

```
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?

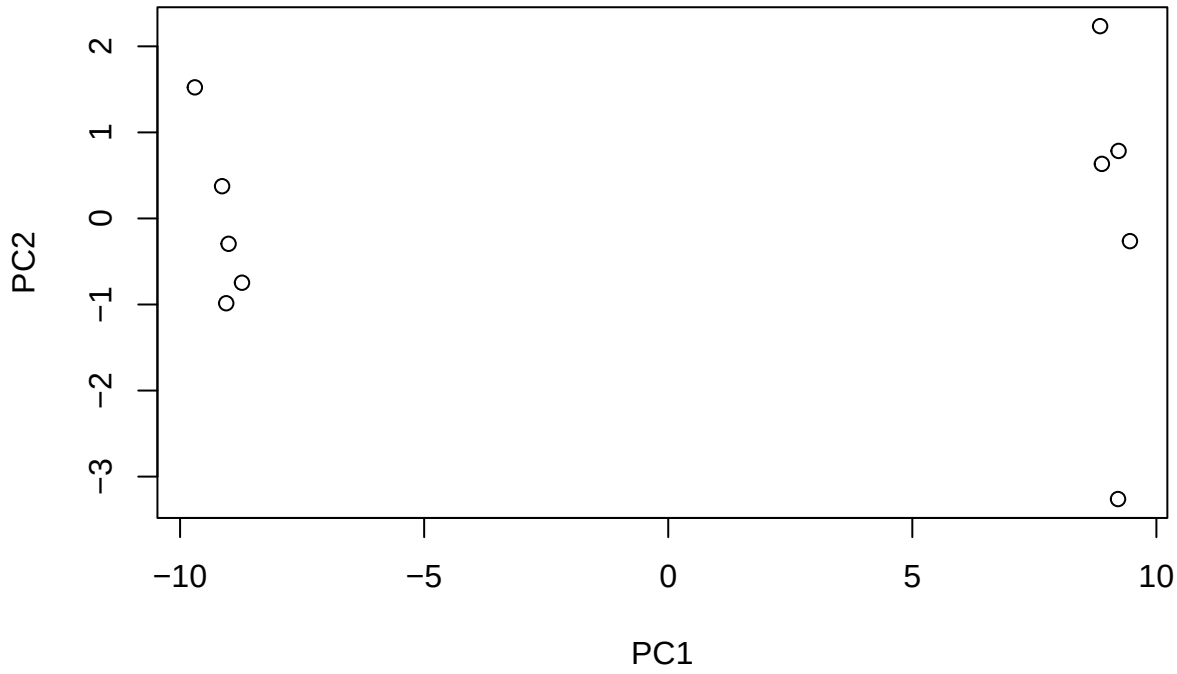
```
dim(rna.data)
```

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

A10. There are 100 genes and 10 samples in this data set.

Use PCA to plot the results.

```
# Take the transpose of our data.
pca <- prcomp( t(rna.data),
               scale=TRUE)

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



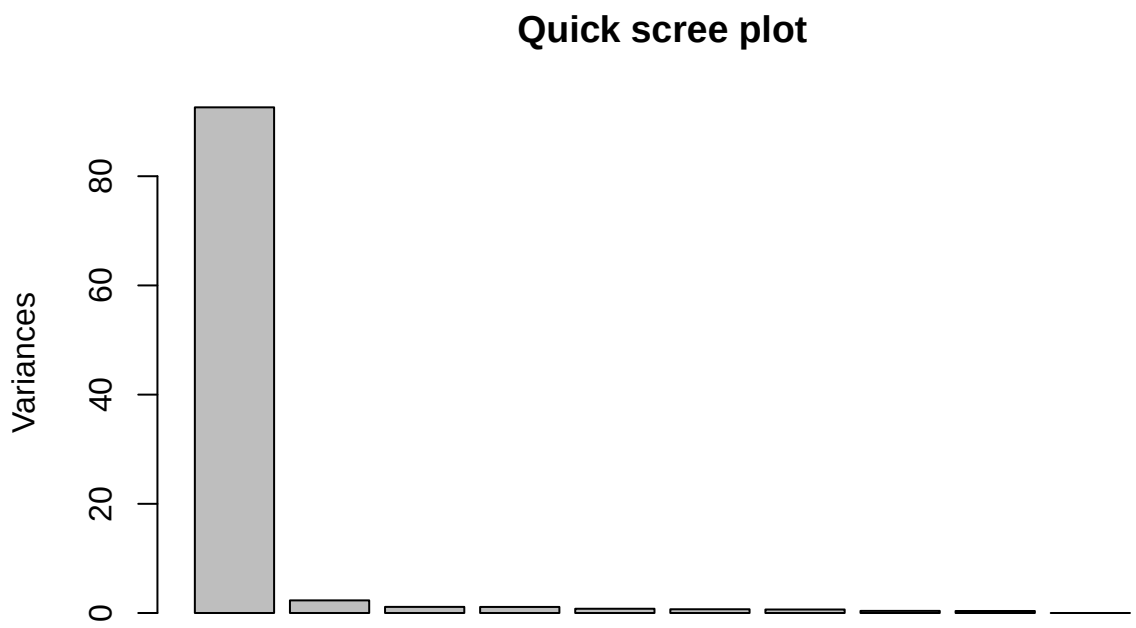
Examine summary of how much variation in the original data each PC accounts for.

```
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
```

Make a barplot summary of Proportion of Variance for Each PC using the plot() function.

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



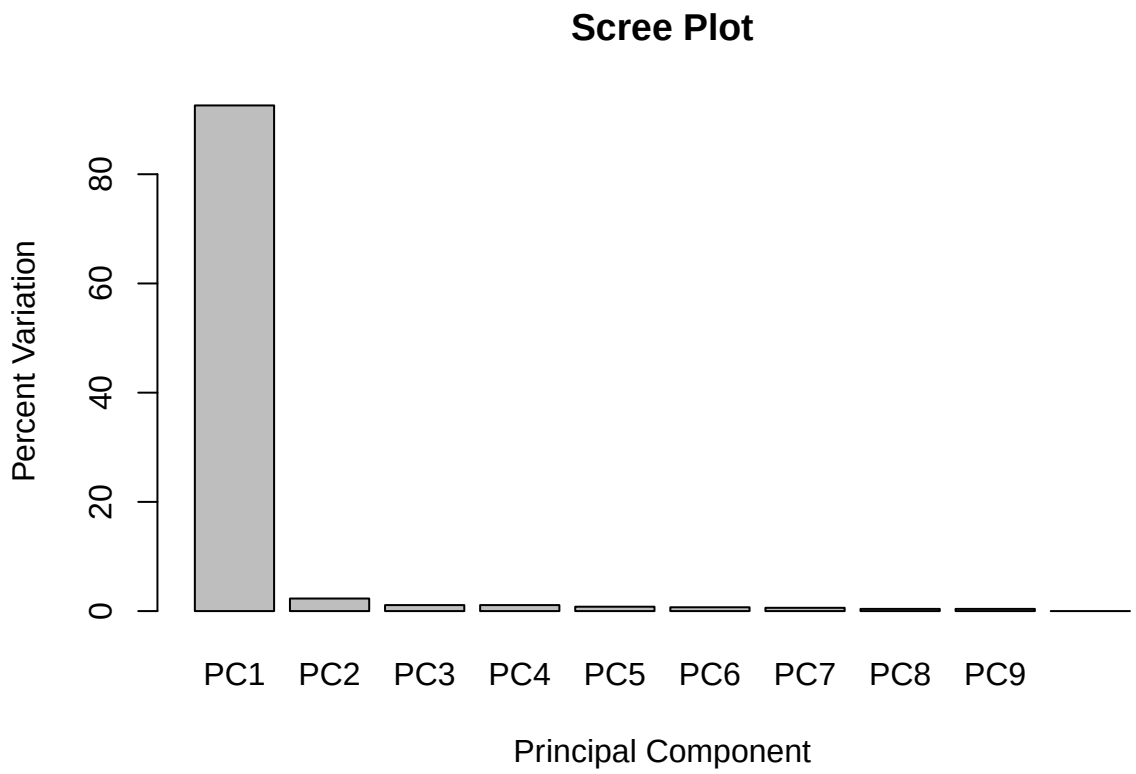
Make scree plot ourselves using square of pca$sdev (standard deviation)

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

# Percent variance is often more informative
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
```

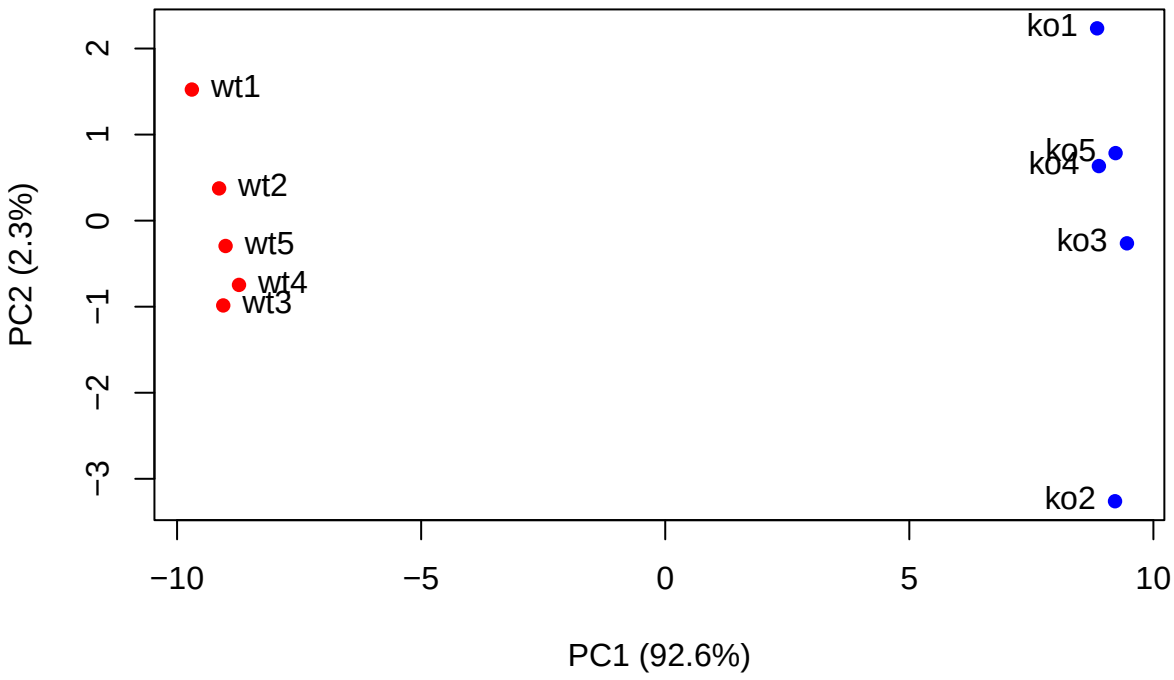
Make a bar plot with pca.var.per.

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



Make PCA plot look nicer / be more useful.

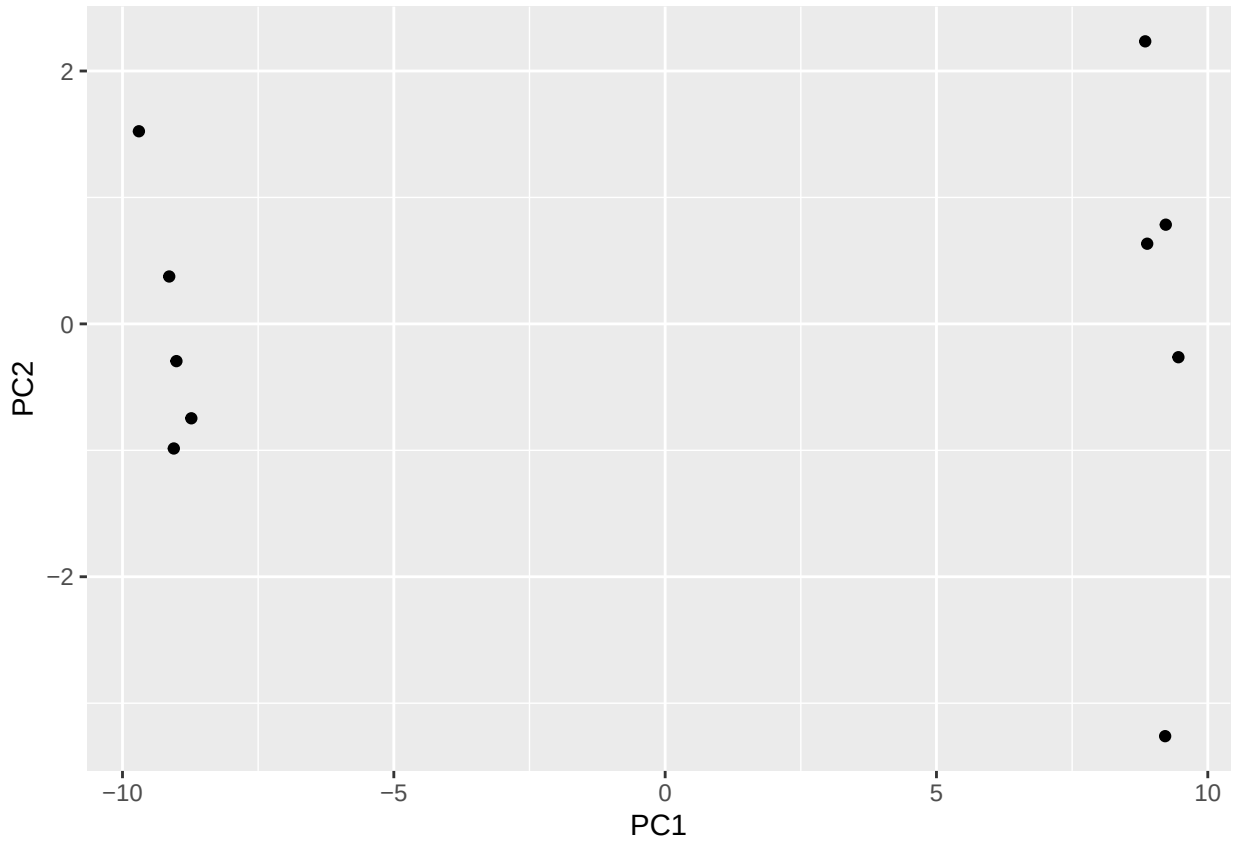
```
# A vector of colors for wt and ko samples
colvec <- colnames (rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

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))
)
```



Part B: Using ggplot

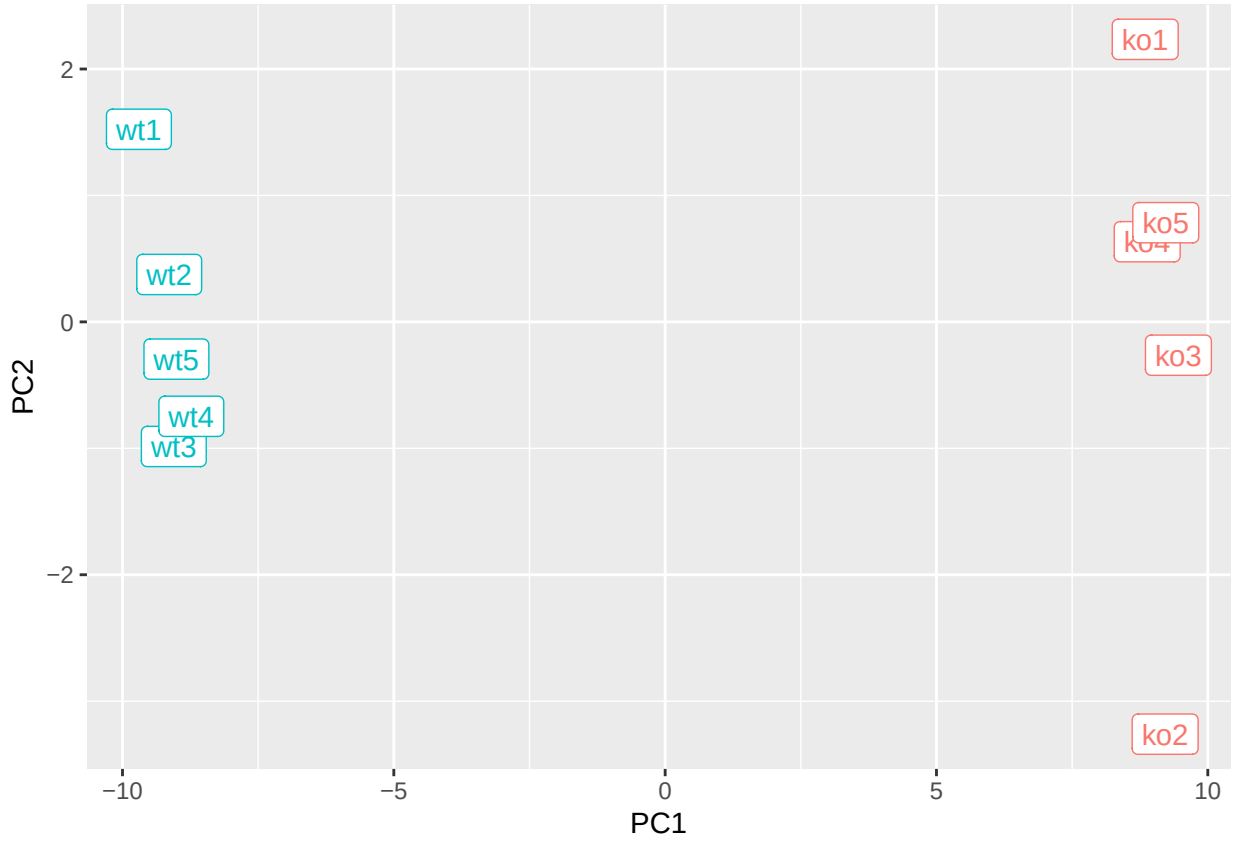
Use ggplot & make a data.frame for the input from PCA results.

```
library(ggplot2)

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

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



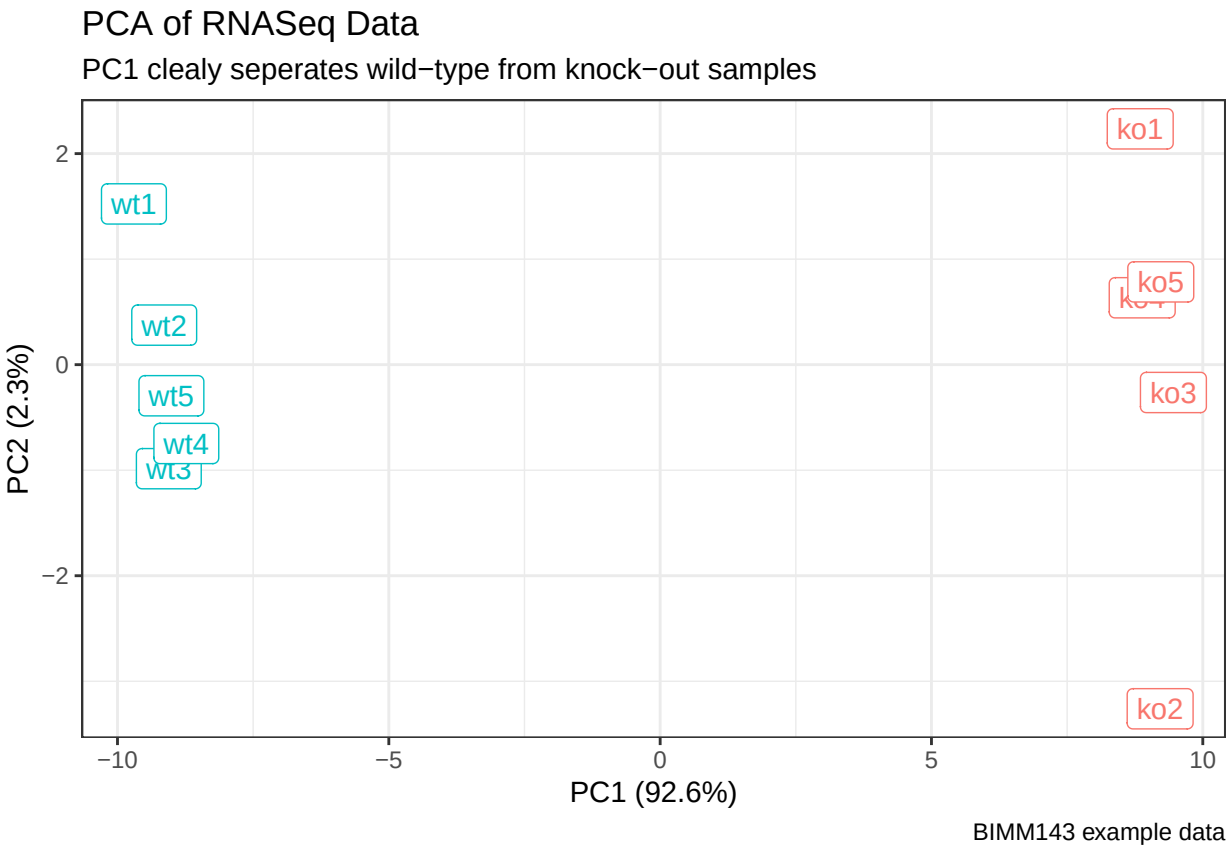
Adding aesthetics.

```
# 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
```



More aesthetics.

```
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()
```



Part C: Optional - Gene Loadings

List top 10 measurements (genes) that contribute most to PC1 (either positive or negative)

```
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"
```

3. Producing a PDF Report

See YAML header / current format.

4. Sync to GitHub

See my Github page: https://github.com/carolinemackey/bimm143