Class08

Samuel Do (PID:A15803613)

2/9/2022

```
#Read UK_foods.csv file data
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url)
# [Q1[ Number of rows and columns
nrow(x)
```

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

```
ncol(x)
```

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

```
#View first 6 rows of data
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 data set
row.names(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
```

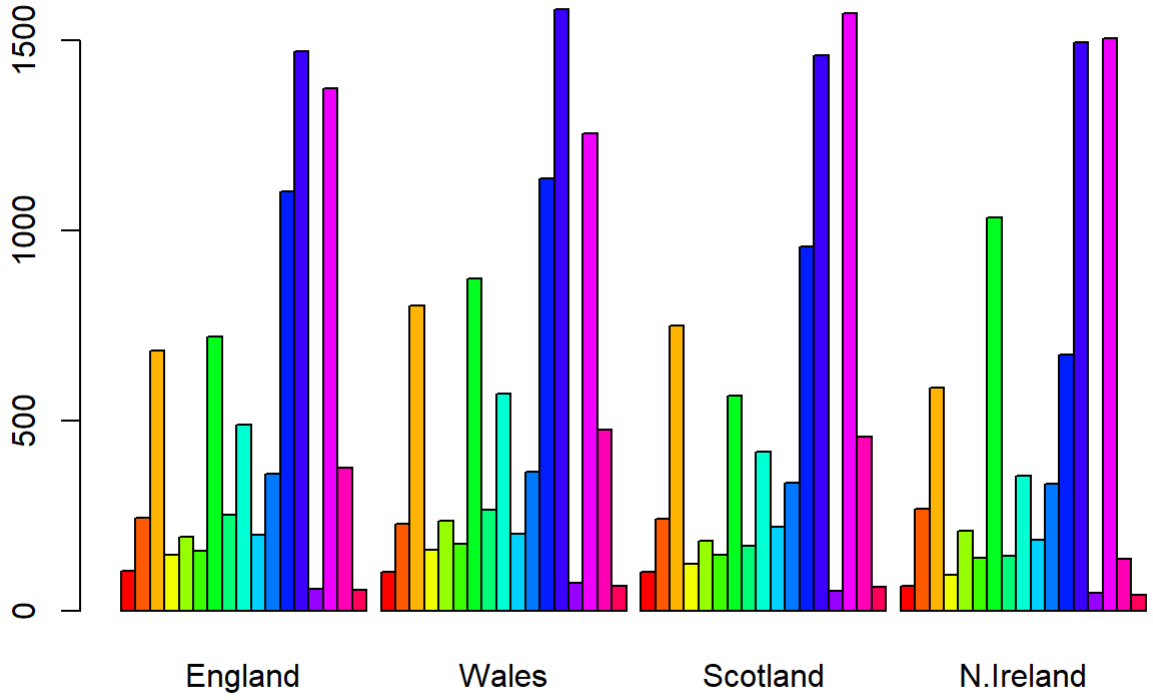
```
#Alternate method to fix data set
x <- read.csv(url, row.names=1)
head(x)
```

```
##                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
```

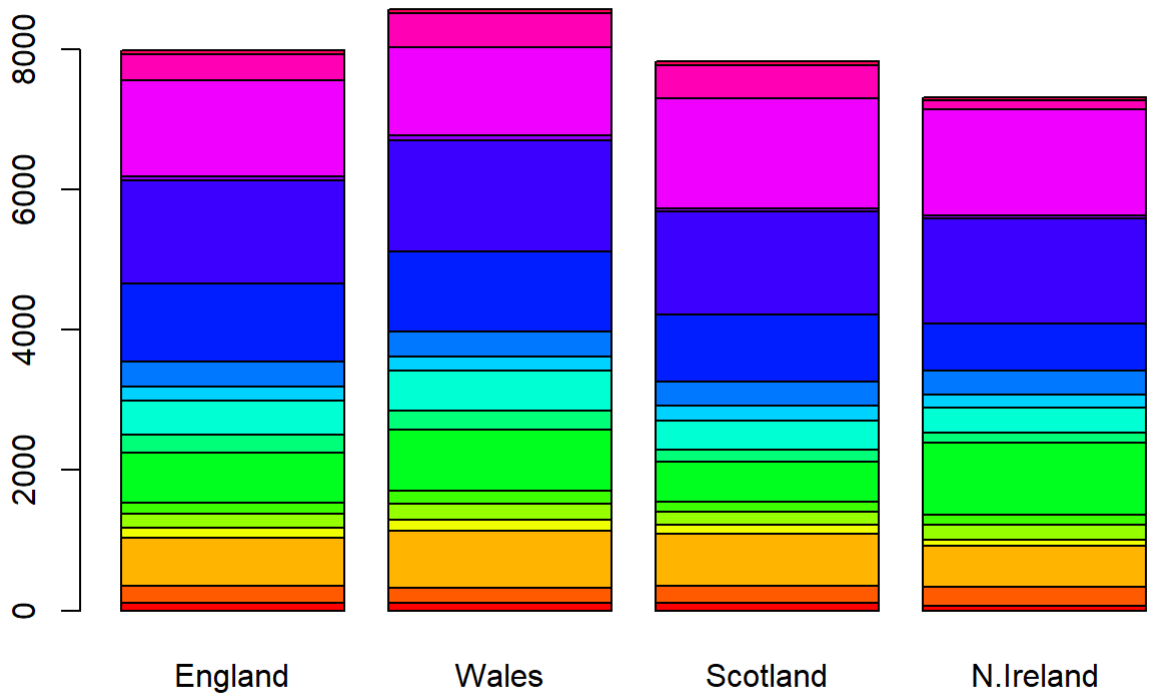
[Q2] Favorite method?

I prefer the second method as it is more prevalent to me that I made sure to designate the row names in the data. While the first method does work, performing this method incorrectly such as running it multiple times results in multiple columns designated as row names

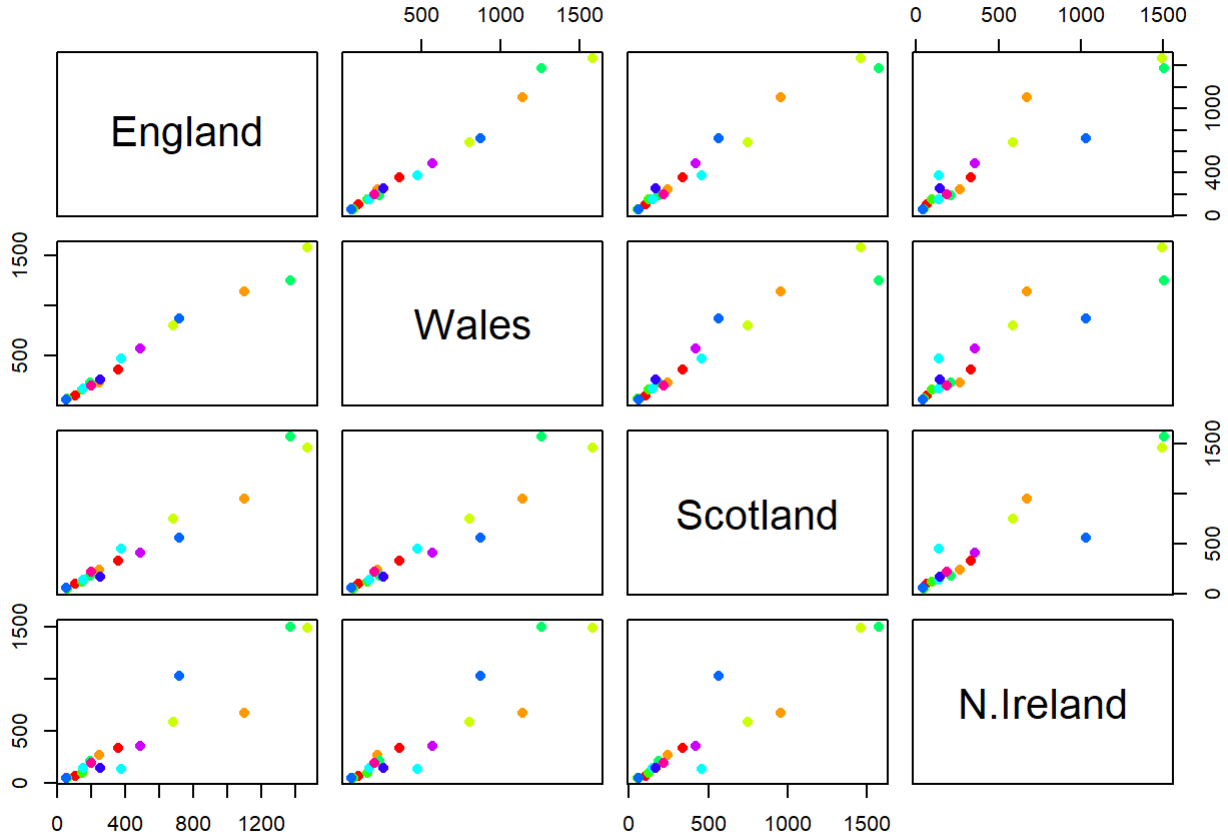
```
barplot(as.matrix(x), beside = T, col=rainbow(nrow(x)))
```




```
# [Q3] Which are argument changes barplot such that each category is stacked rather than side-by
-side
#Removing "beside = T" results in a different barplot
barplot(as.matrix(x),col=rainbow(nrow(x)))
```




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



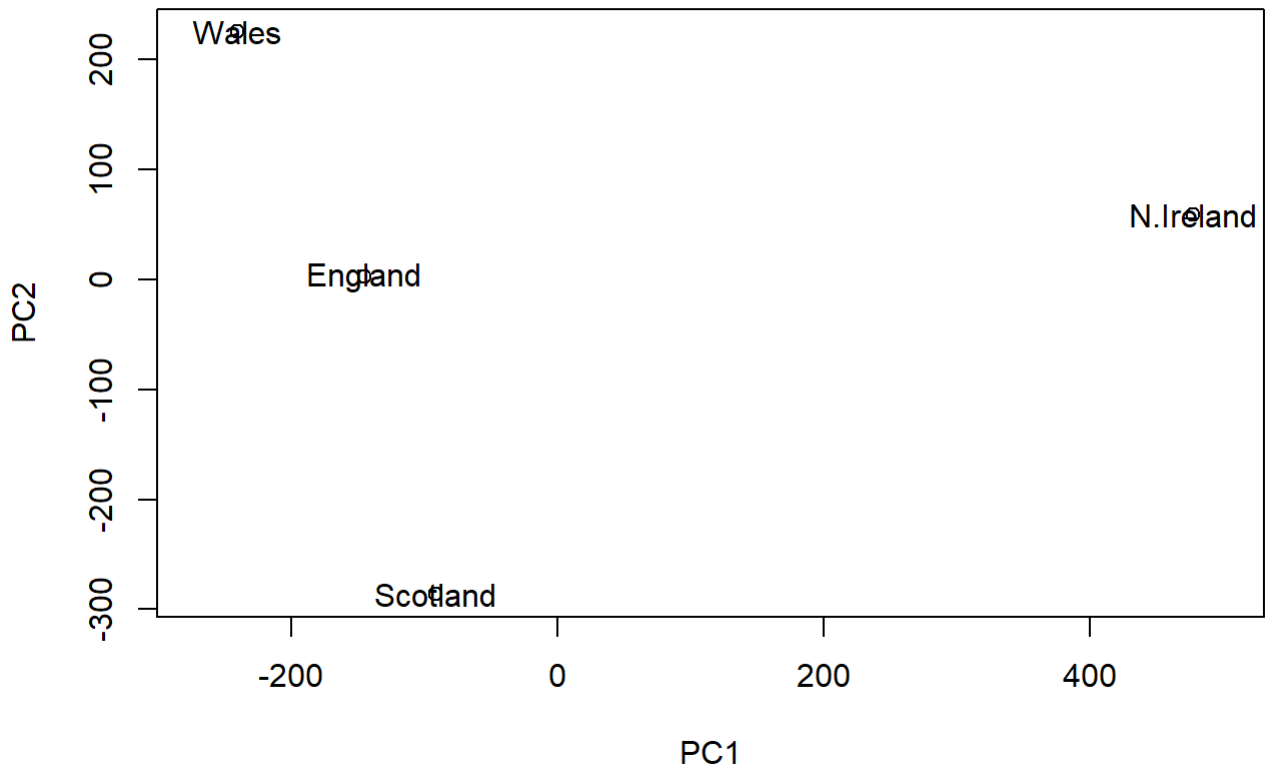

```
# [Q5]  Can you make sense of the following code and resulting figure? What does it mean if a gi
ven point lies on the diagonal for a given plot?
# If a given point lies on the diagonal of a given plot, this means that there is a good correla
tion between the specific food categorical data of two countries.

# [Q6] What is the main differences between N. Ireland and the other countries of the UK in term
s of this data-set?
# Based on the data-set, Fresh_potatoes and alcoholic_drinks are the main differences between N.
Ireland and the other UK countries
```

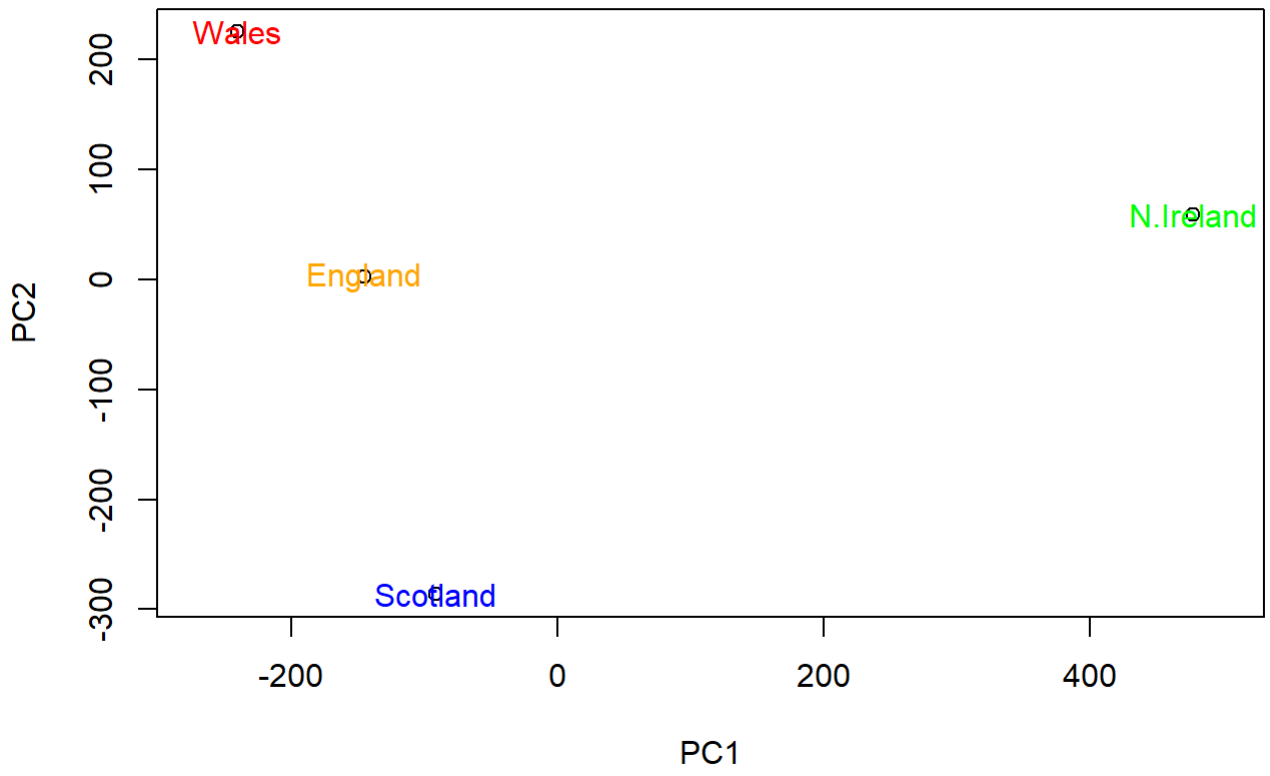
```
# Using prcomp() function
pca <- prcomp(t(x))
summary(pca)
```

```
## 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] 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] Add colors
country_colors <- c("orange", "red", "blue", "green")
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=country_colors)
```



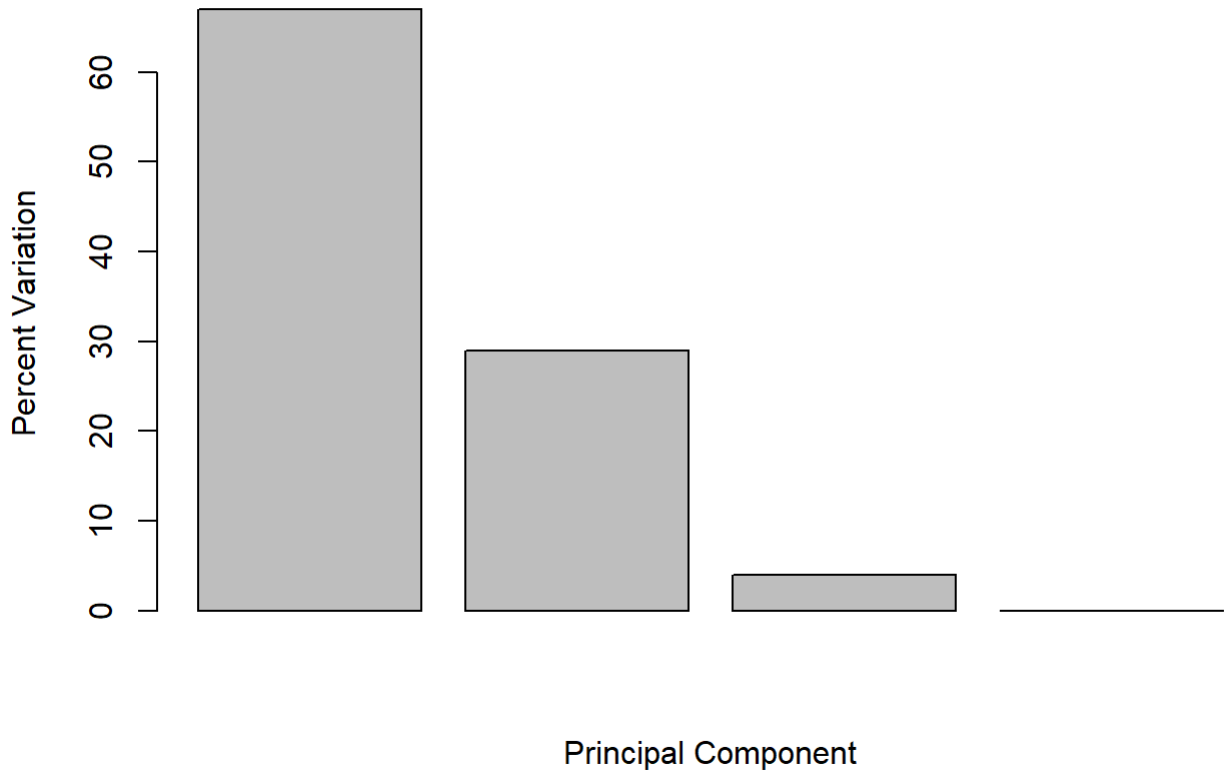

```
#Use pca$sdev to calculate variation
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

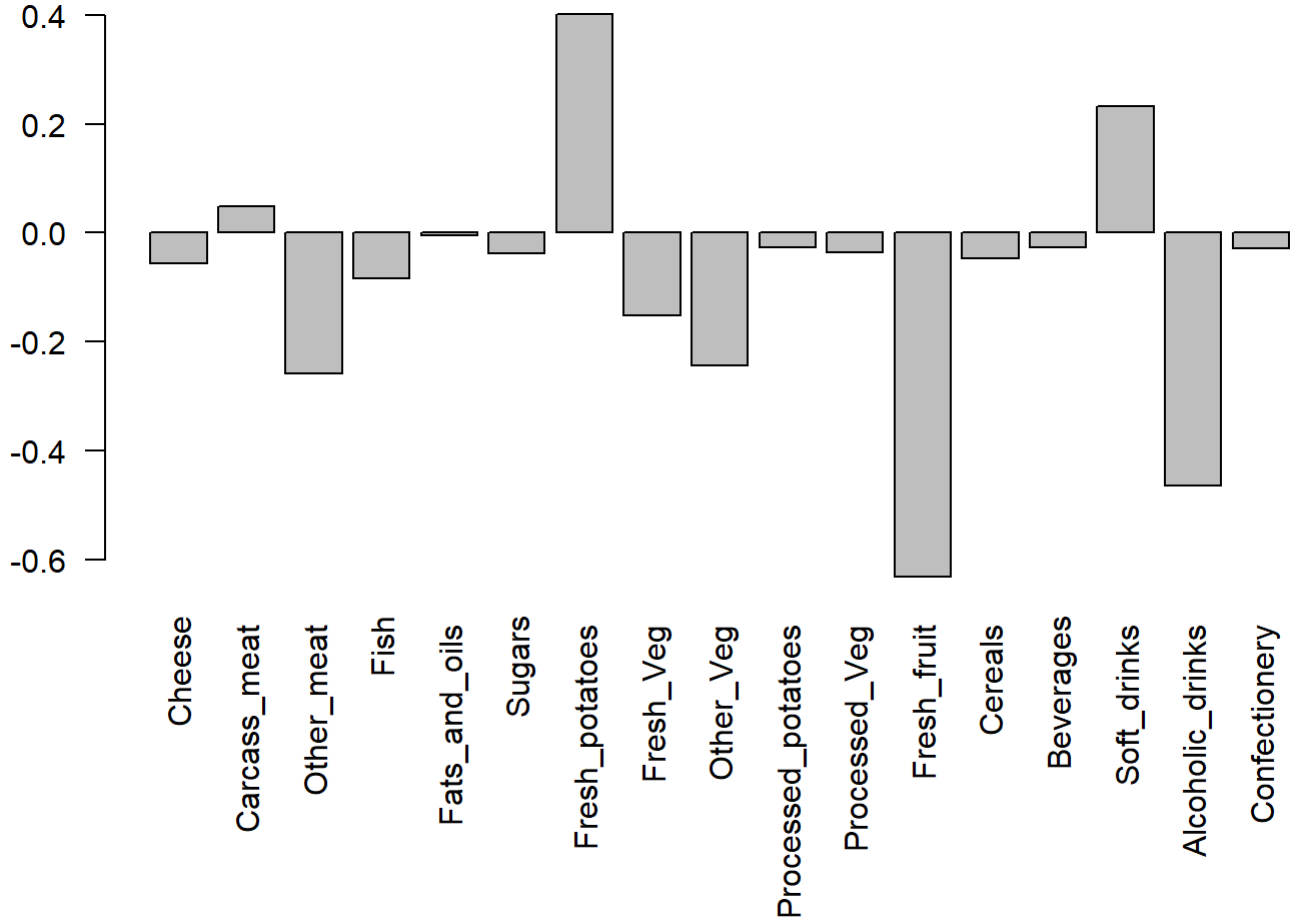
```
# Second method to calculate variation
z <- summary(pca)
z$importance
```

```
##                             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
```

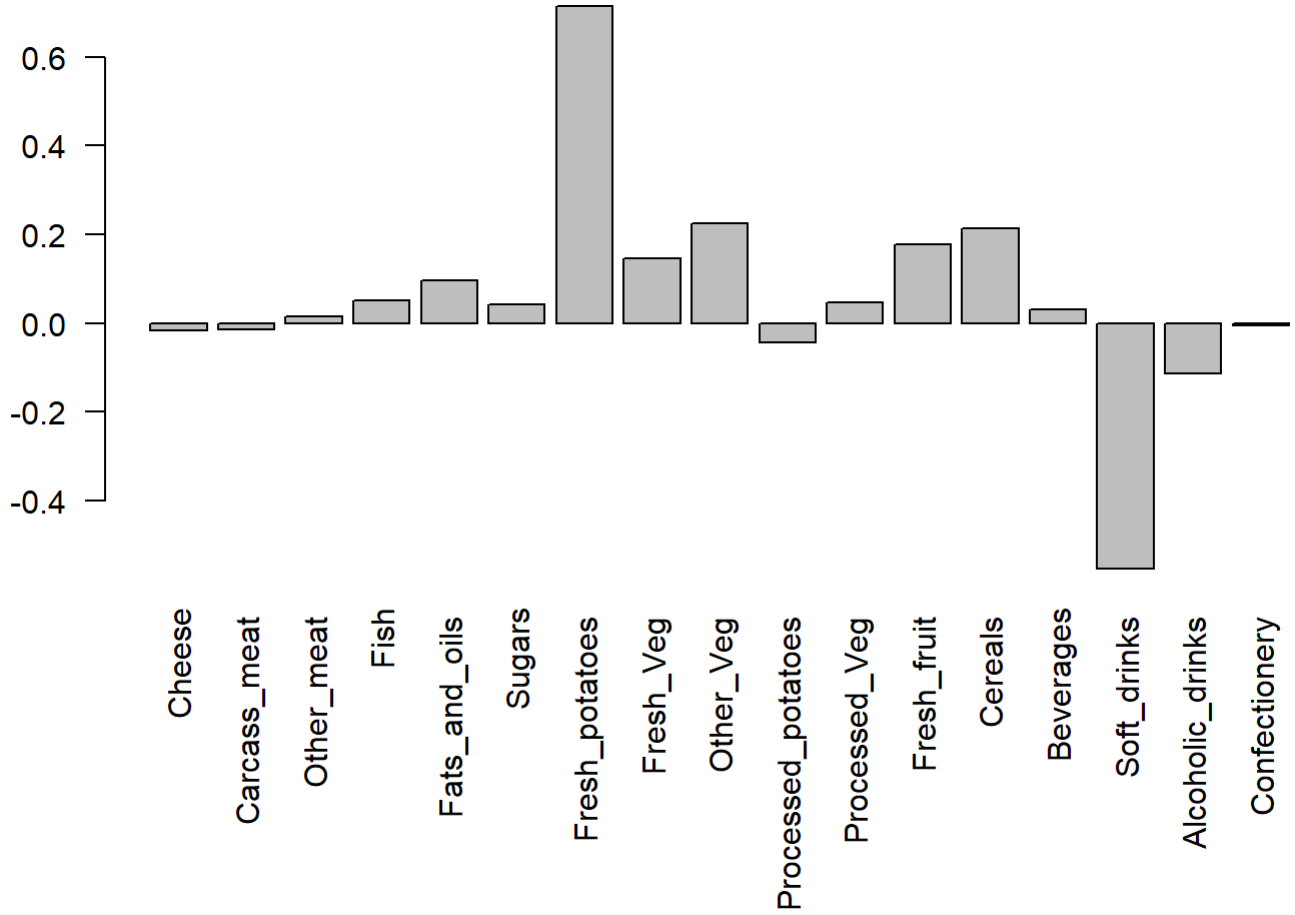
```
#Variation plot
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```




```
# PC1 Loading Plot using pca$rotation
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```

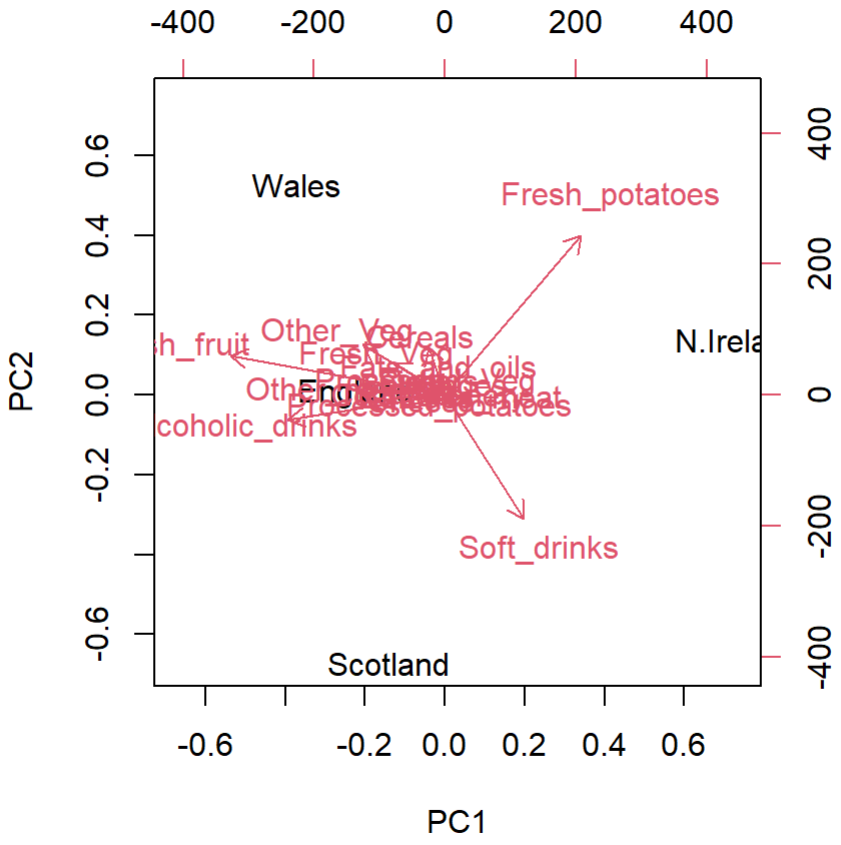



```
# [Q9] PC2 Loading Plot
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```




```
# The loading plot shows that for PC2, Fresh Potatoes and soft drinks feature predominantly. PC2 is a proposed axis, and the graph shows how foods such as fresh potatoes push Wales in the positive direction, while foods such as softs drinks push Scotland in the negative direction.
```

```
# Use bioplot() 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 dataset?
# Number of rows = number of genes
nrow(rna.data)
```

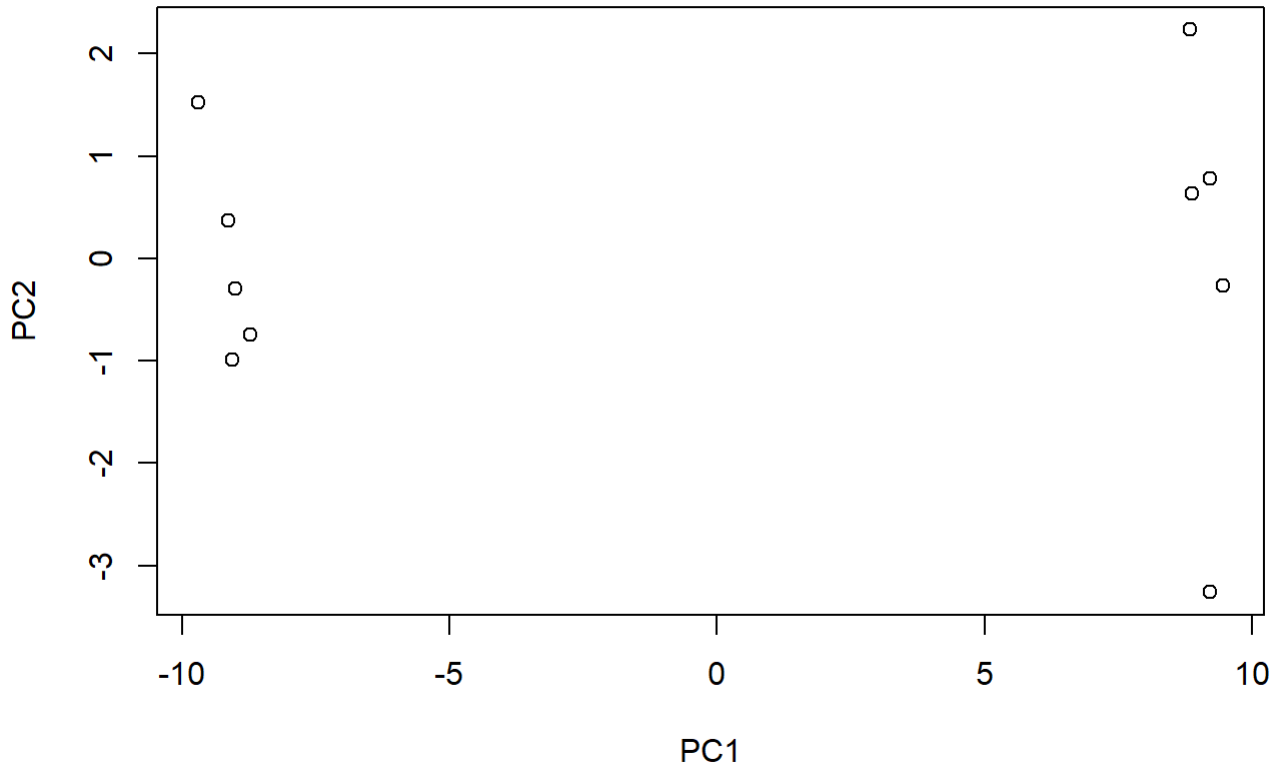
```
## [1] 100
```

```
# Number of columns = number of samples
ncol(rna.data)
```

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

```
# Therefore, there are 100 genes and 10 samples of each gene
```

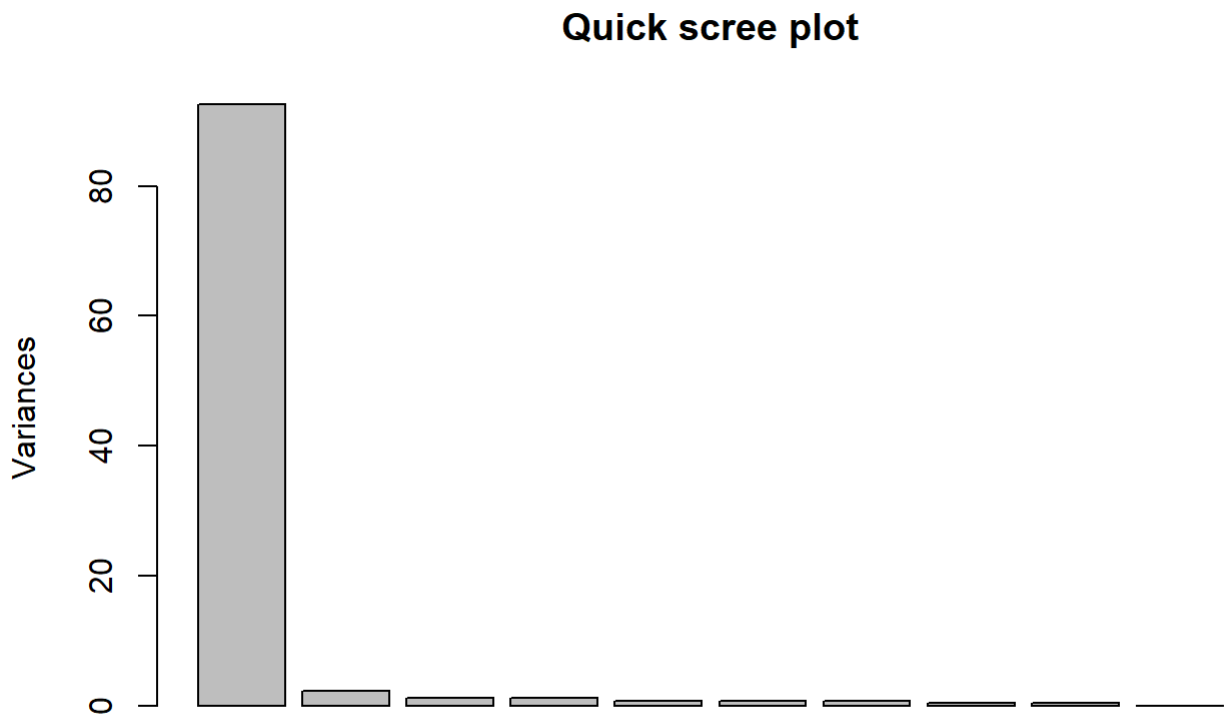
```
# Plotting PCA
# Transpose data
pca <- prcomp(t(rna.data), scale=TRUE)
# Simple plot of pc1 and pc2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```




```
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
```

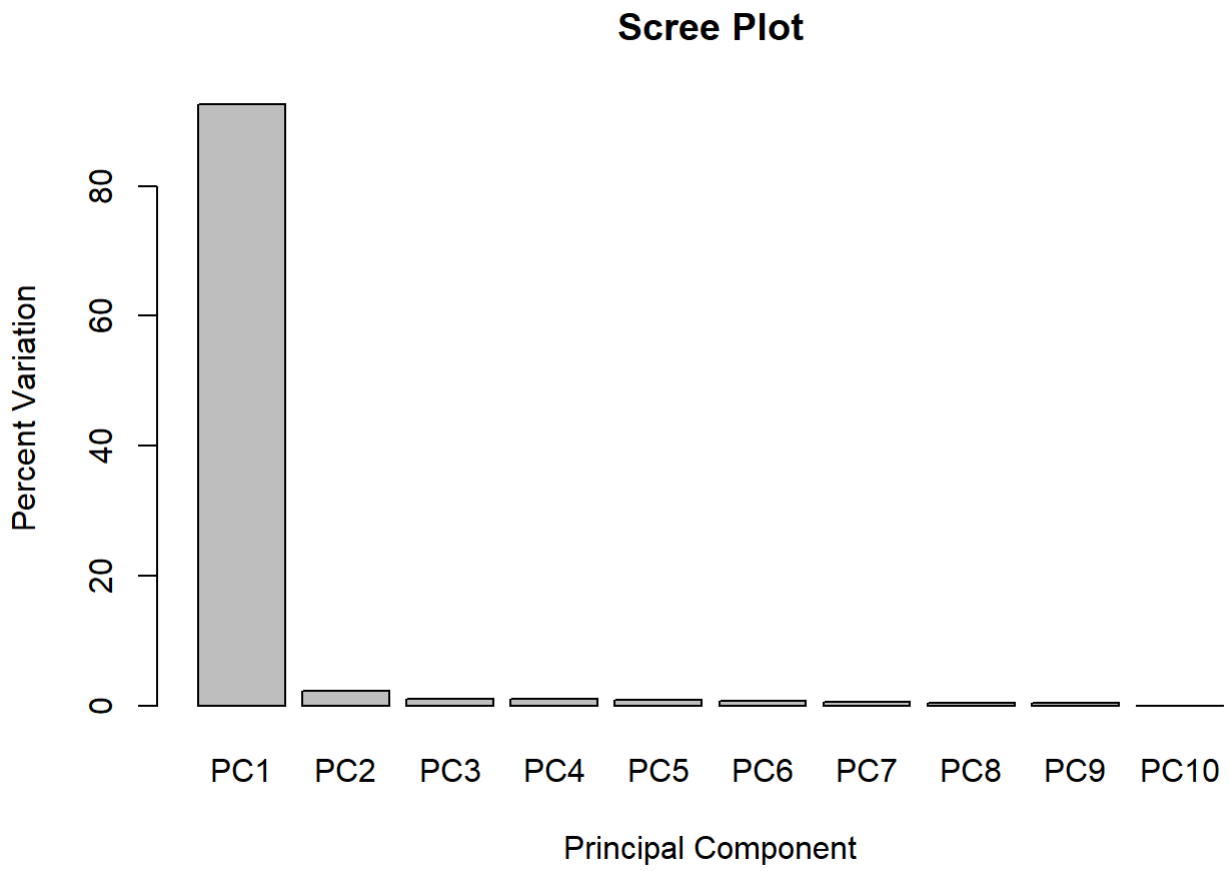
```
#Simple Scree Plot of proportion of variance
plot(pca, main="Quick scree plot")
```




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

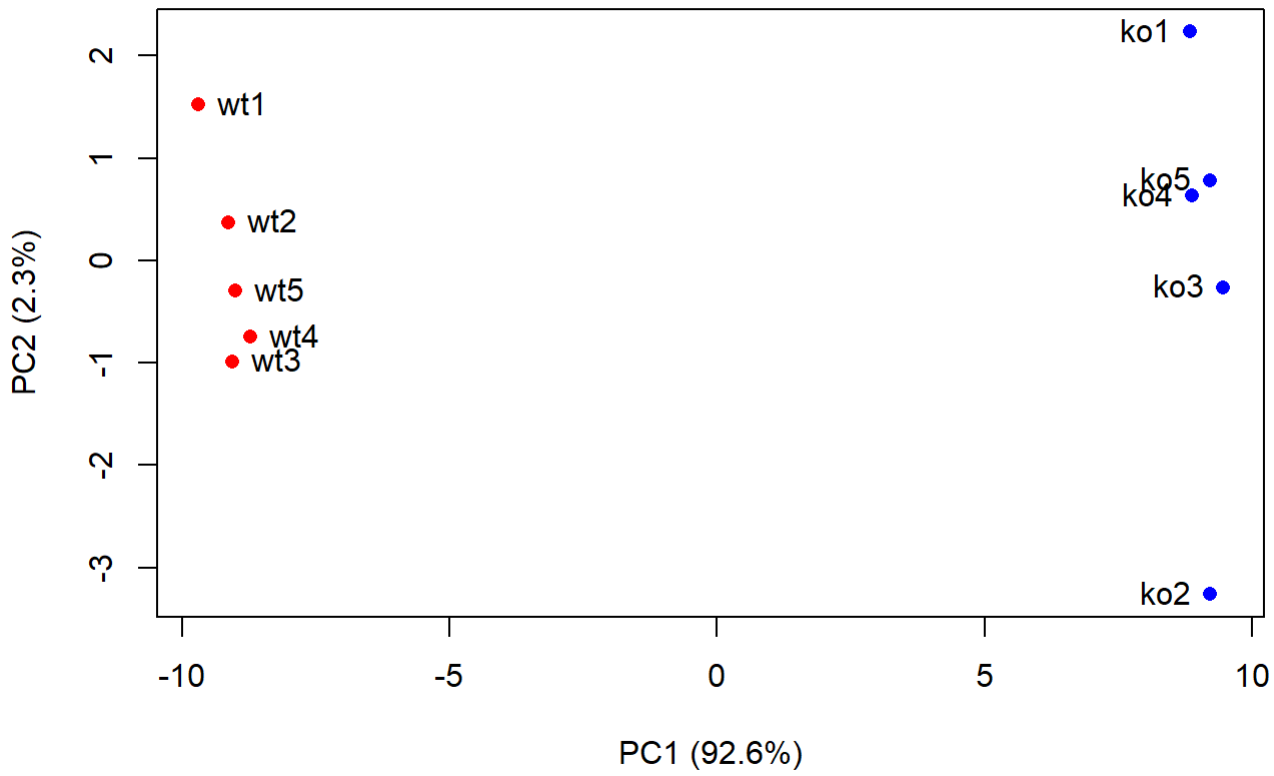
```
#Scree Plot of Percent Variation vs Principal Component
barplot(pca.var.per, main="Scree Plot",
        names.arg=paste0("PC", 1:10),
        xlab="Principal Component", ylab="Percent Variation")
```



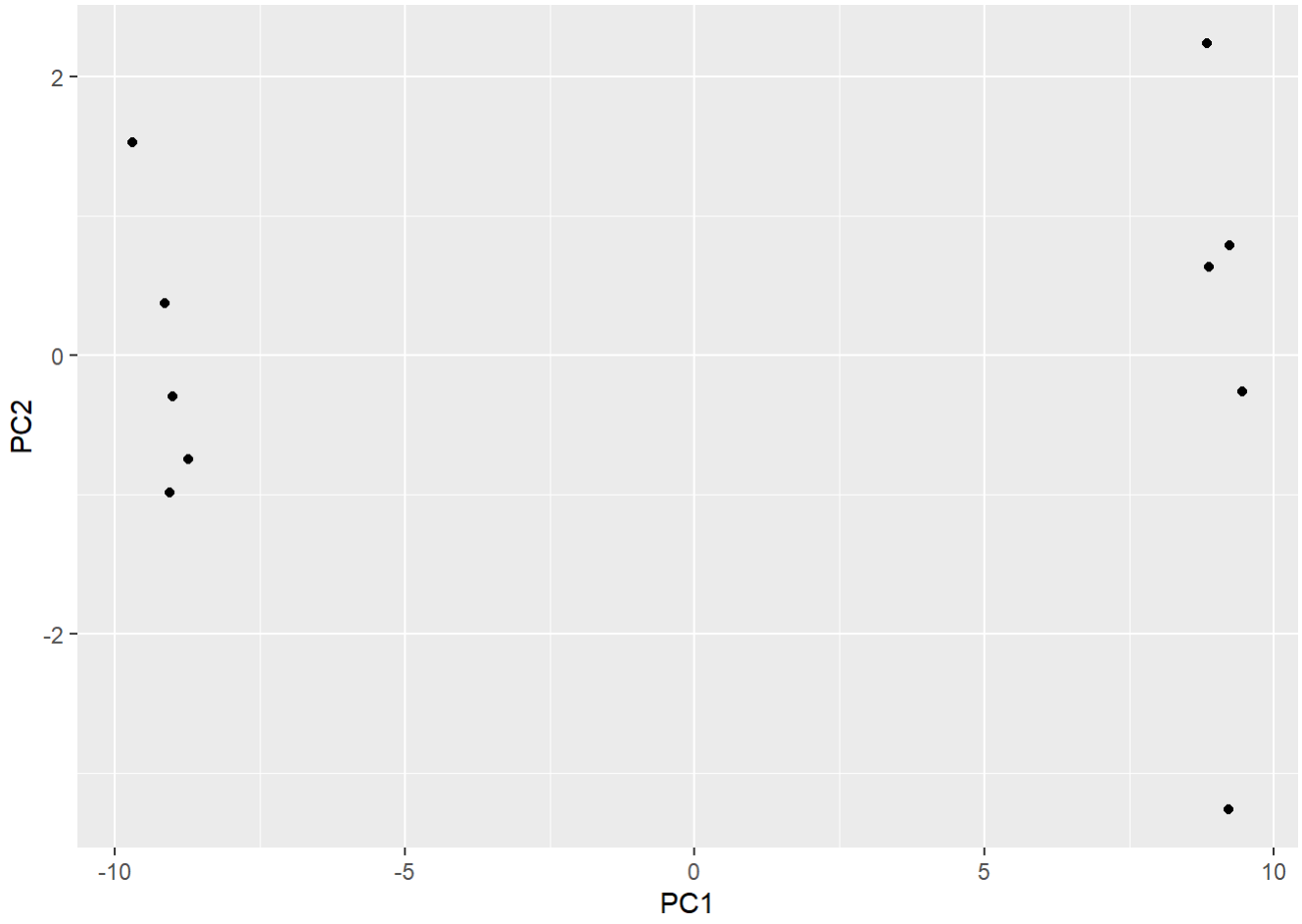

```
#A vector of colors for wt and ko samples
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

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)))
```

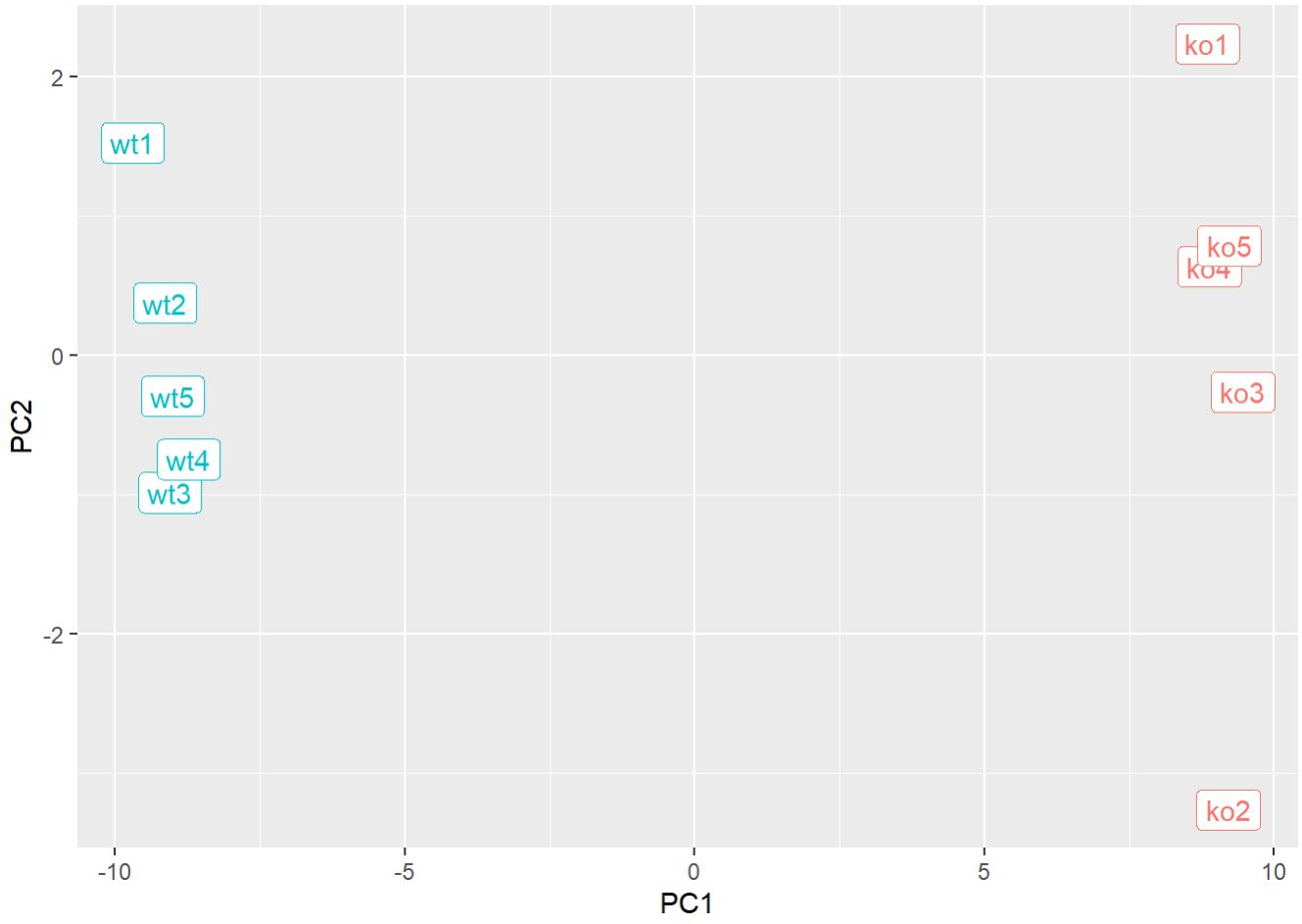



```
#Use ggplot2 for PCA
library(ggplot2)
df <- as.data.frame(pca$x)
# Basic ggplot
ggplot(df) +
  aes(PC1, PC2) +
  geom_point()
```

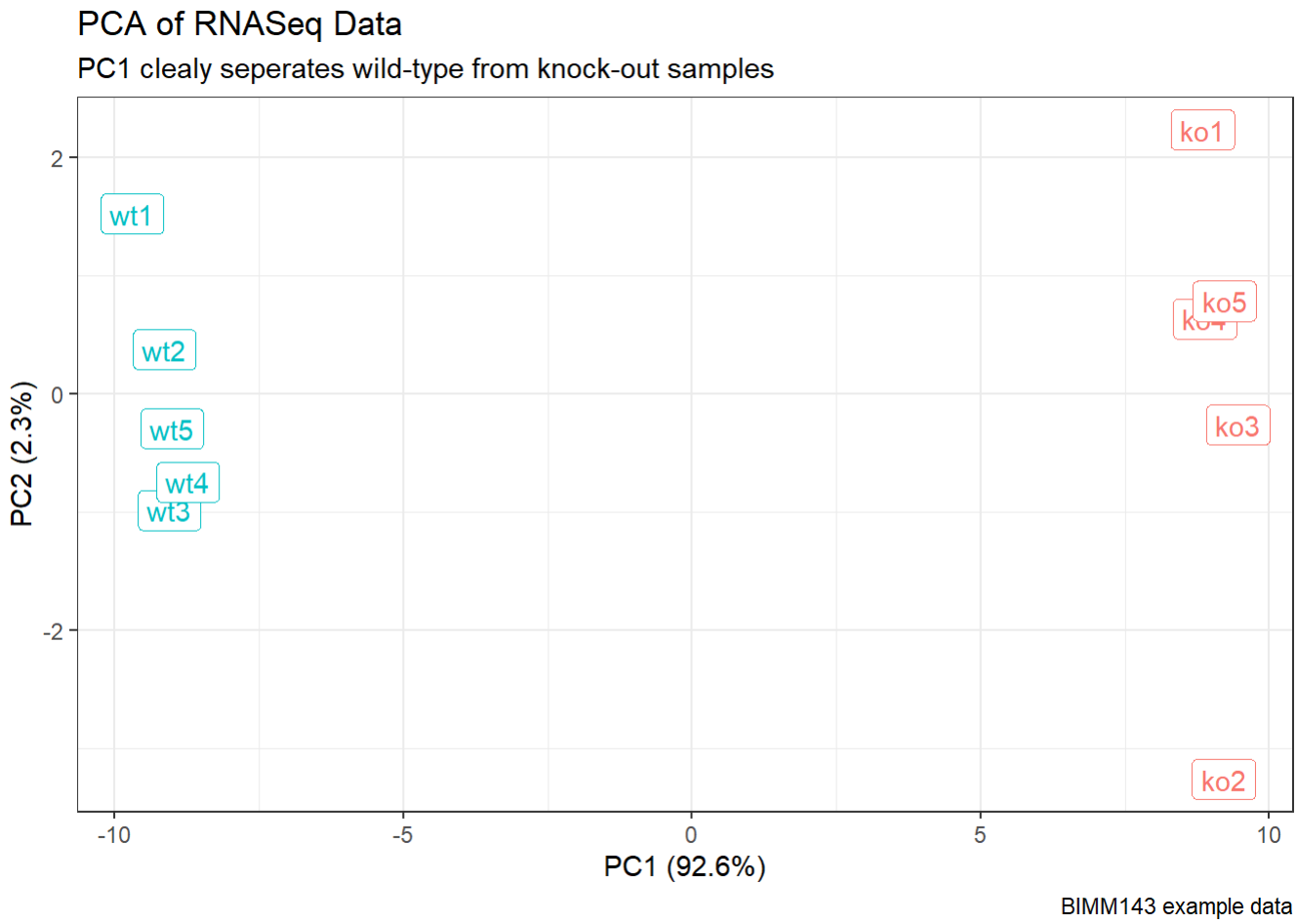



```
# Add a 'wt' and 'ko' "condition" column
df$samples <- colnames(rna.data)
df$condition <- substr(colnames(rna.data),1,2)

p <- ggplot(df) +
        aes(PC1, PC2, label=samples, col=condition) +
        geom_label(show.legend = FALSE)
p
```




```
#Add title, labels, and caption
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()
```




```
#Finding measurements (top 10) that contribute to PC1 in either direction
loading_scores <- pca$rotation[,1]
gene_scores <- abs(loading_scores)
gene_score_ranked <- sort(gene_scores, decreasing=TRUE)

#Names of 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"
```