Lab07

Yushi Li (A15639705)

2/8/2022

1. PCA of UK food data

Data import

```
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
```

Checking the 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
```

```
# Remove row name for first row:
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
```

```
# Checking dimensions again:
dim(x)
```

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

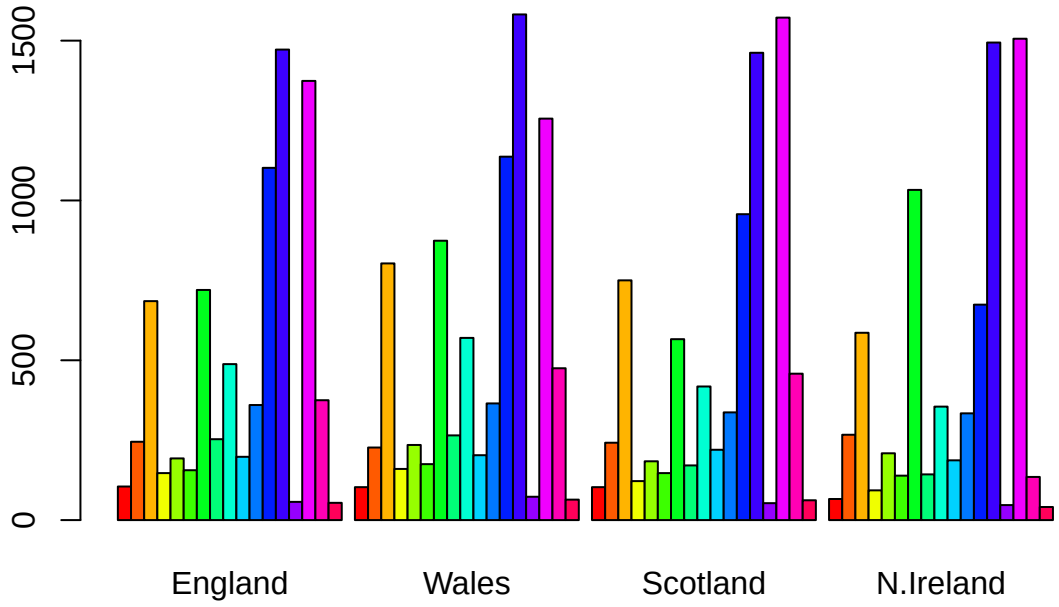
```
# An alternative approach to address incorrect row names:
x <- read.csv(url, row.names = 1)
head(x)
```

```
##                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
```

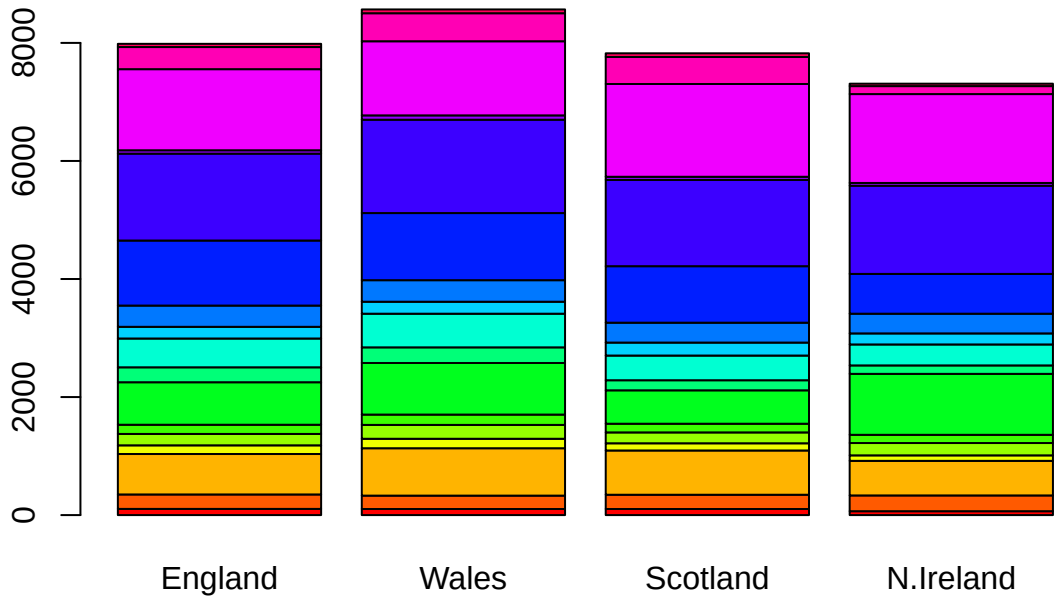
```
# Q2. Which approach to solving the 'row-names problem' mentioned above do you
# prefer and why? Is one approach more robust than another under certain
# circumstances?

# A. I prefer the second method. It requires fewer lines of code and is also
# more robust due to the consistence of row.name=1 over x <- x[,1]. Calling
# x <- x[,1] multiple times would result in the loss of more than one row names.
```

Spotting major differences and trends

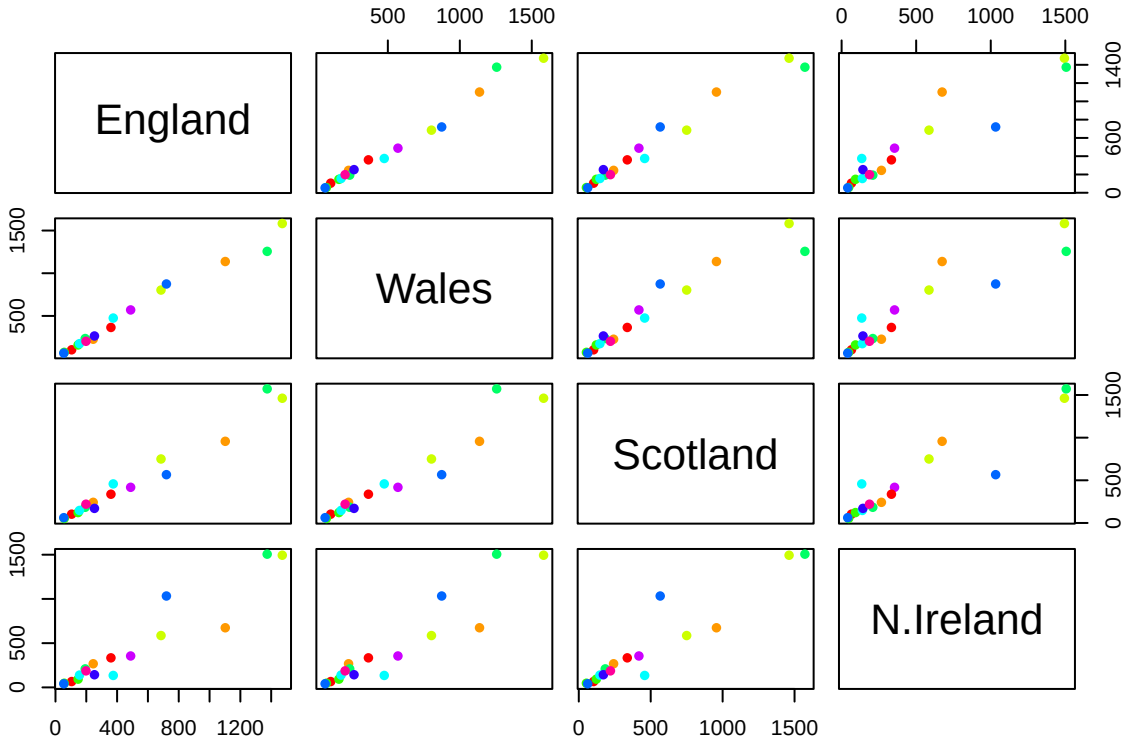
```
# Barplots don't really help much in terms of interpreting this set of data.
barplot(as.matrix(x), beside = T, col=rainbow(nrow(x)))
```




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

# A. Setting beside=F, or simply leave this argument out as it is F by default.
barplot(as.matrix(x), beside = F, col = rainbow(nrow(x)))
```




```
# Am I missing Q.4? I can't find it on the 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)
```




```
# A. I can somewhat make sense of the code and the resulting figure. The code
# makes one plot for each possible pair of regions (i.e. England vs. Scotland).
# The resulting figure show the relationships between the consumption of each
# food type between a pair of regions. If a point is on the diagonal, it means
# the consumption of that particular food is consistent with a linear
# relationship between two regions.


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

# A. Based on this particular set of data, an average person in N. Ireland
# consumes significantly more fresh potatoes and less fresh fruits and alcoholic
# drinks than people from other countries of the UK.
```

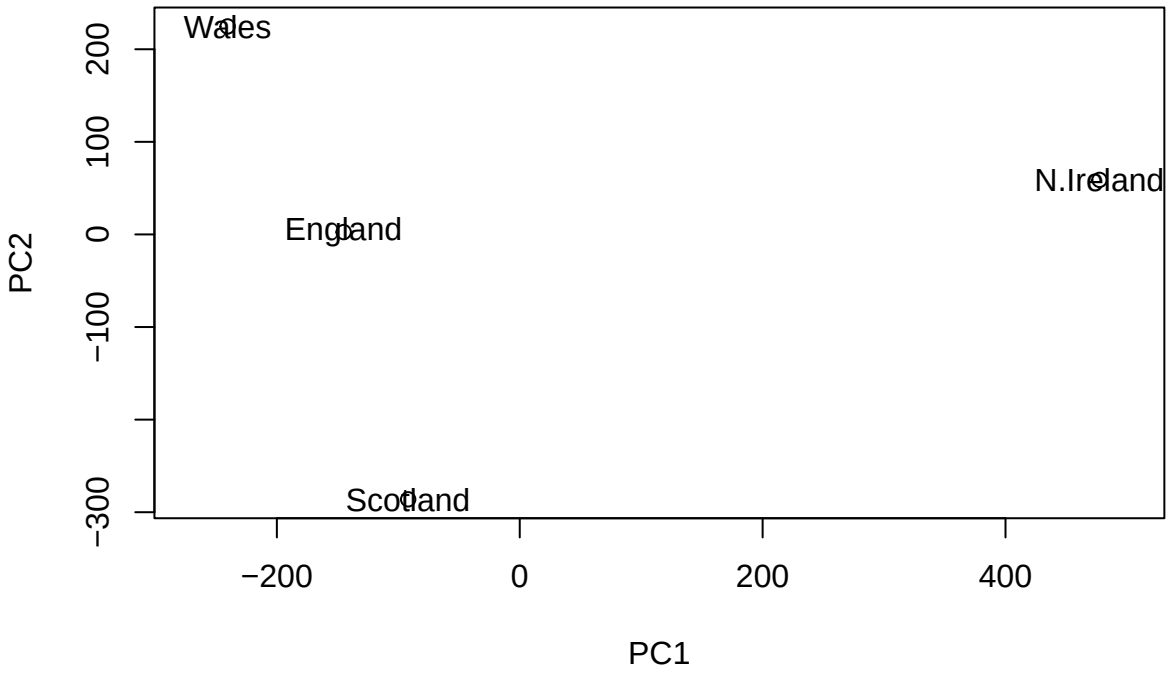
PCA to the rescue

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

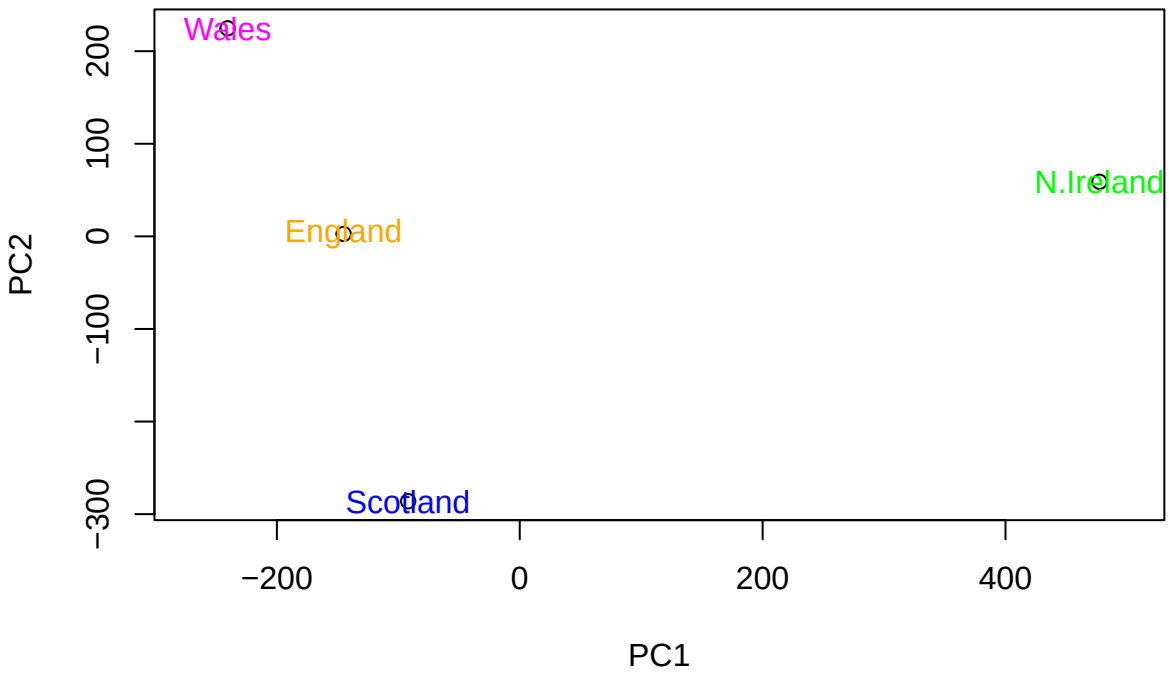
```
## Importance of components:
##                             PC1       PC2       PC3       PC4
```

```
## 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
```

```r
# Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over t
# 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))
```




```r
# Q8. Customize your plot so that the colors of the country names match the colors in our UK and Irelan
country_col <- c("orange", "magenta", "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_col)
```



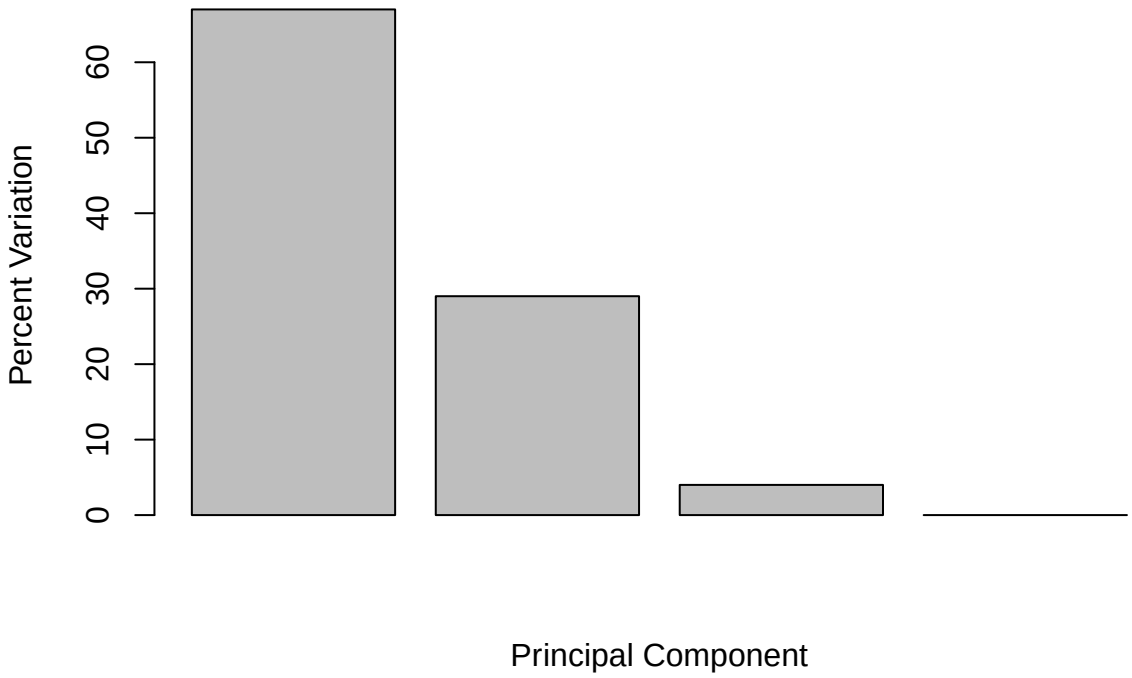

```
# To calculate how much variation in the original data each PC accounts for:
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

```
# Results agree with prcomp(), with PC1, PC2, and PC3 accounting for 67%, 29%,
# and 4% of total variation, respectively, as shown below using summary():
## the second row here...
z <- summary(pca)
z$importance
```

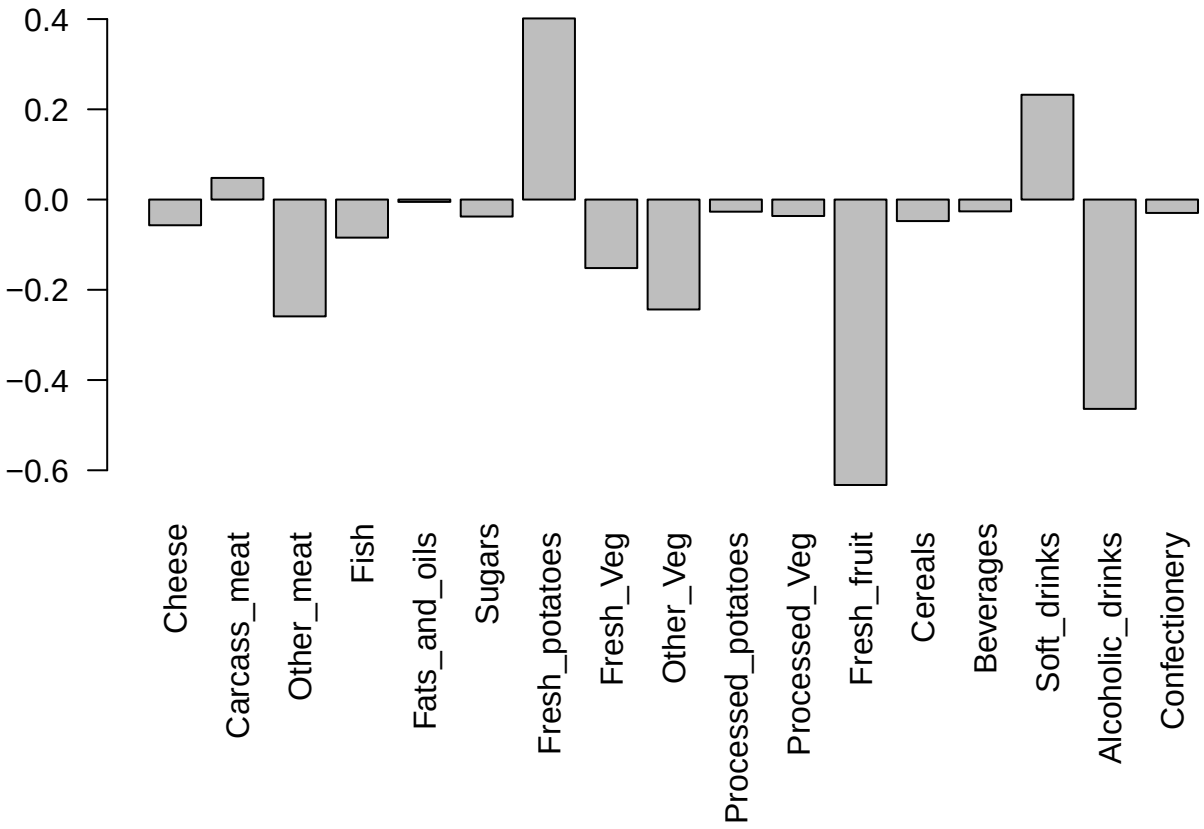
```
##                             PC1       PC2      PC3          PC4
## Standard deviation     324.15019 212.74780 73.87622 4.188568e-14
## Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
## Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

```
# Barplots of proportion of variance each PC accounts for:
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```

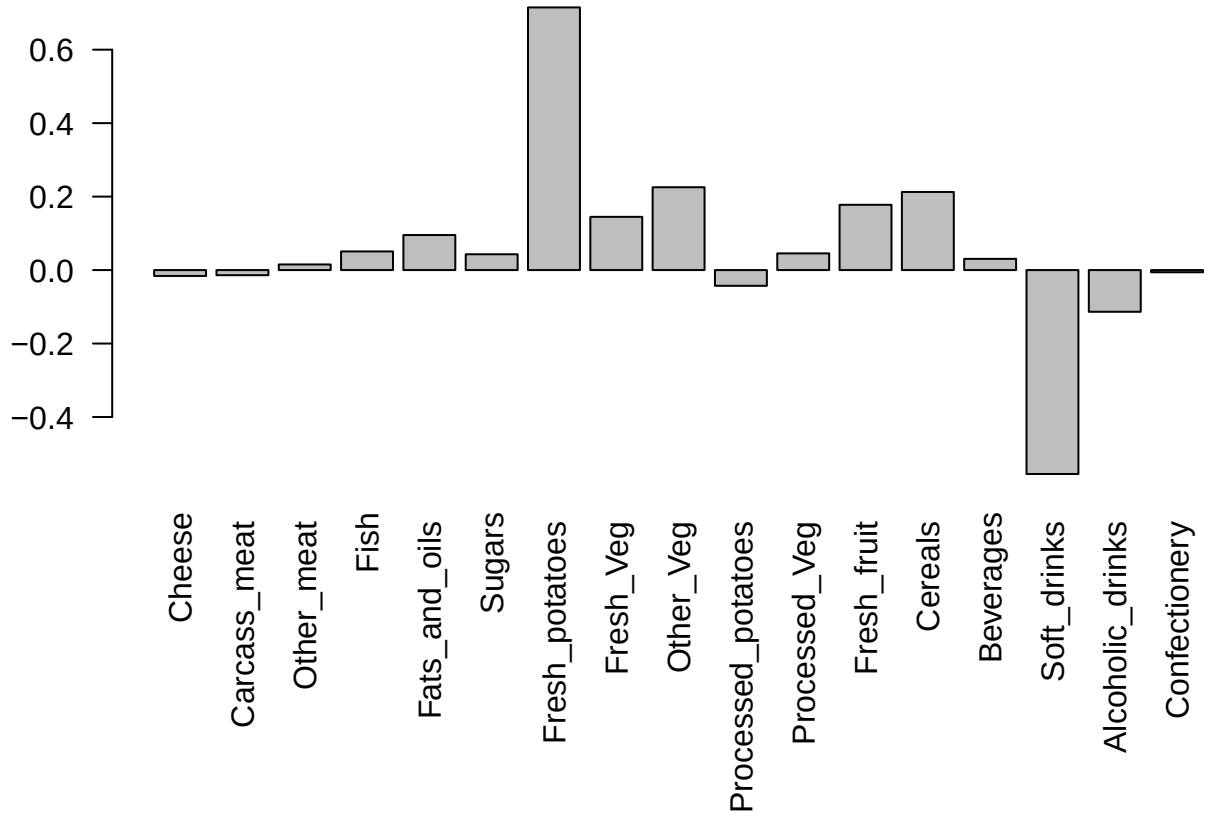


Digging deeper (variable loadings)

```
## 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 )
```



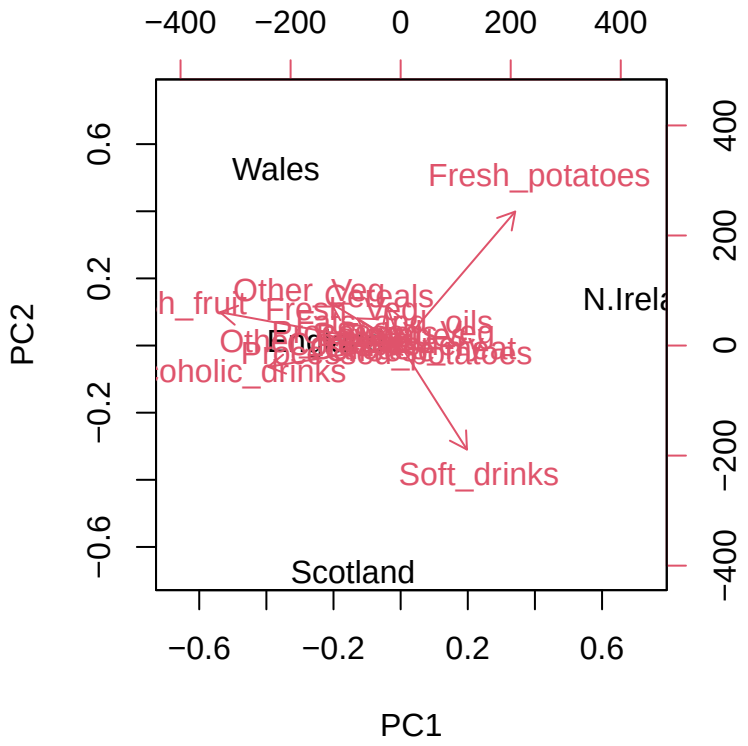

```
# Q9. Generate a similar 'loadings plot' for PC2. What two food groups feature
# prominently and what does PC2 mainly tell us about?
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```




```
# A. Fresh_potatoes (again) and Soft_drinks feature most prominently in PC2,
# with Fresh_potatoes again pushing N. Ireland in the positive direction and
# with Soft_drinks pushing the rest of the countries in the negative direction
# on the y-axis.
```

Biplots

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



2. PCA of RNA-seq data

Import 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
```

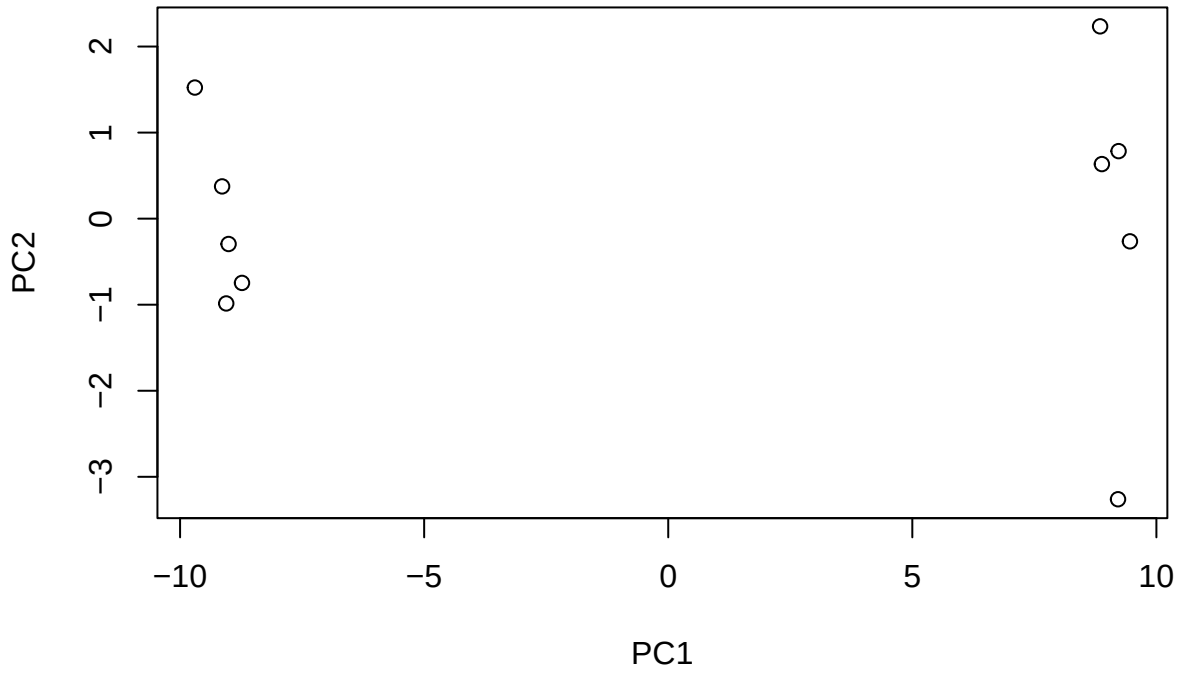
```
# Q10. How many genes and samples are in this data set?
dim(rna.data)
```

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

```
# A. There are 100 genes from 10 samples.
```

Make the unpolished plot of PC1 and PC2 after PCA.

```
pca2 <- prcomp(t(rna.data), scale=TRUE)
plot(pca2$x[,1], pca2$x[,2], xlab="PC1", ylab="PC2")
```

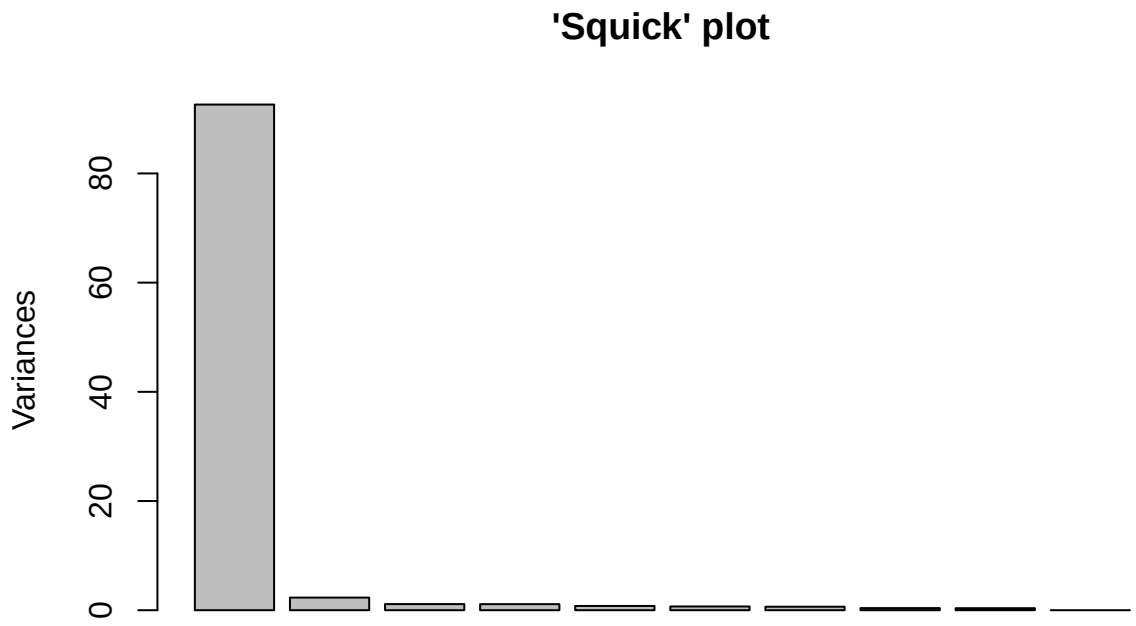


Assess how much variance each PC accounts for:

```
summary(pca2)
```

```
## 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
```

```r
#make a quick scree plot:
plot(pca2, main="'Squick' plot")
```

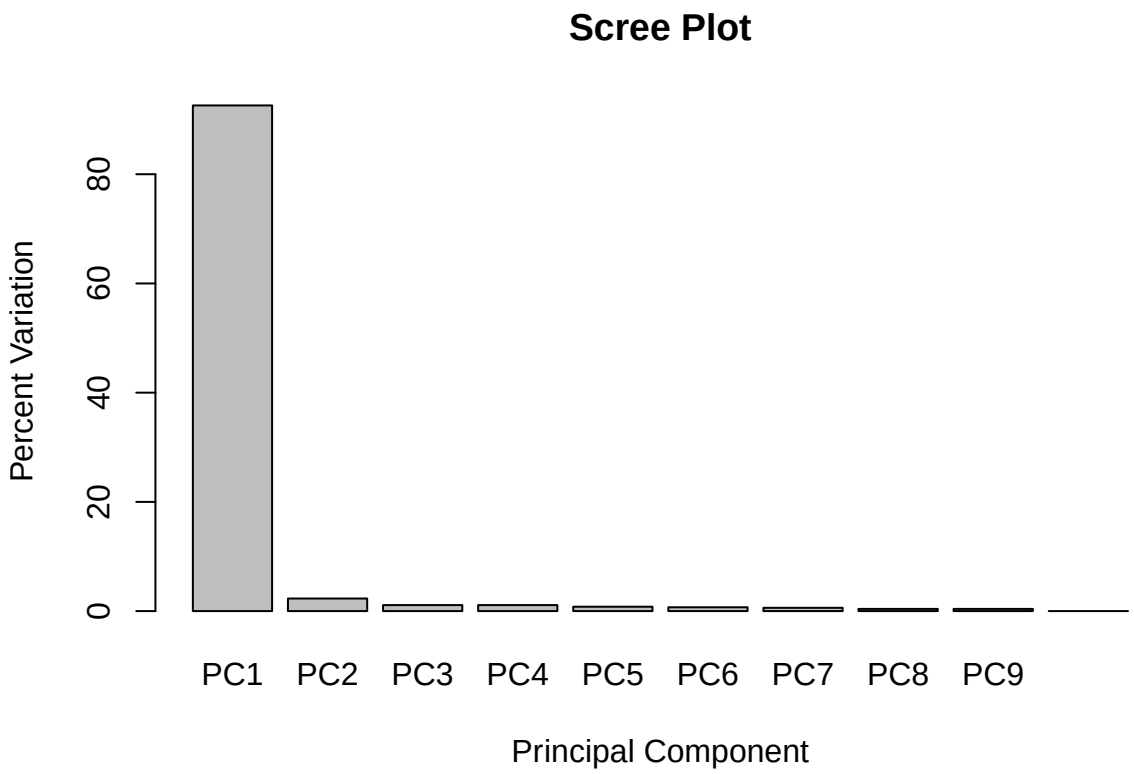


Try manually make the same scree plot:

```r
# Variance captured per PC
pca.var <- pca2$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
```

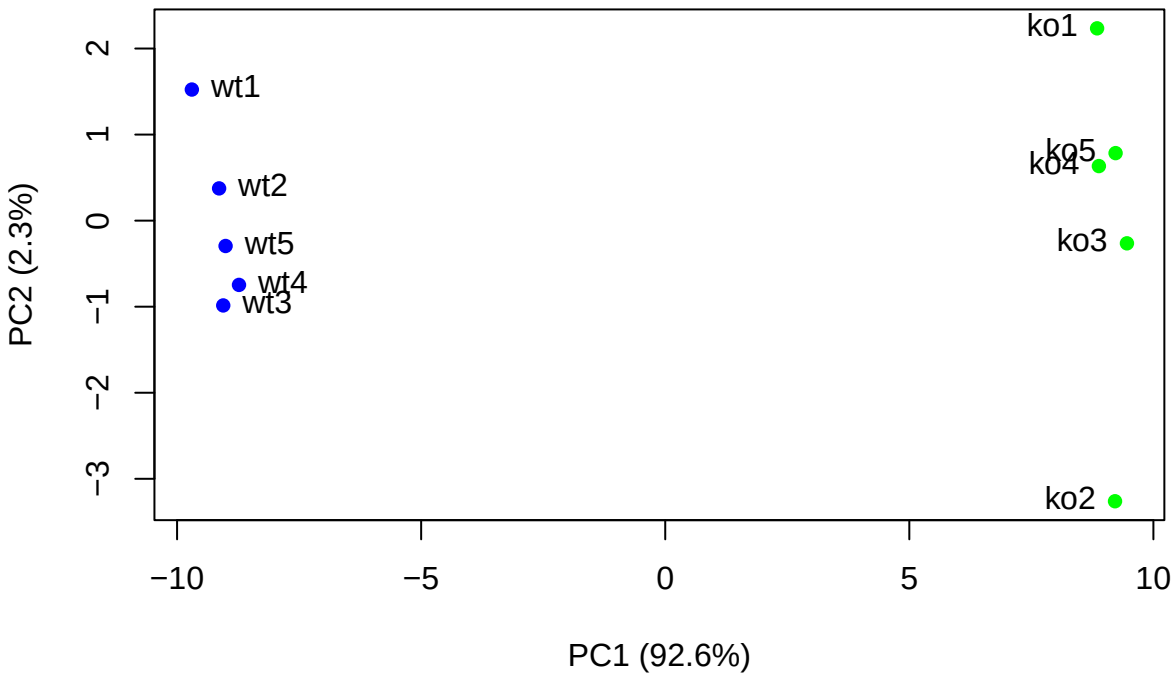
```
##  [1] 92.6  2.3  1.1  1.1  0.8  0.7  0.6  0.4  0.4  0.0
```

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



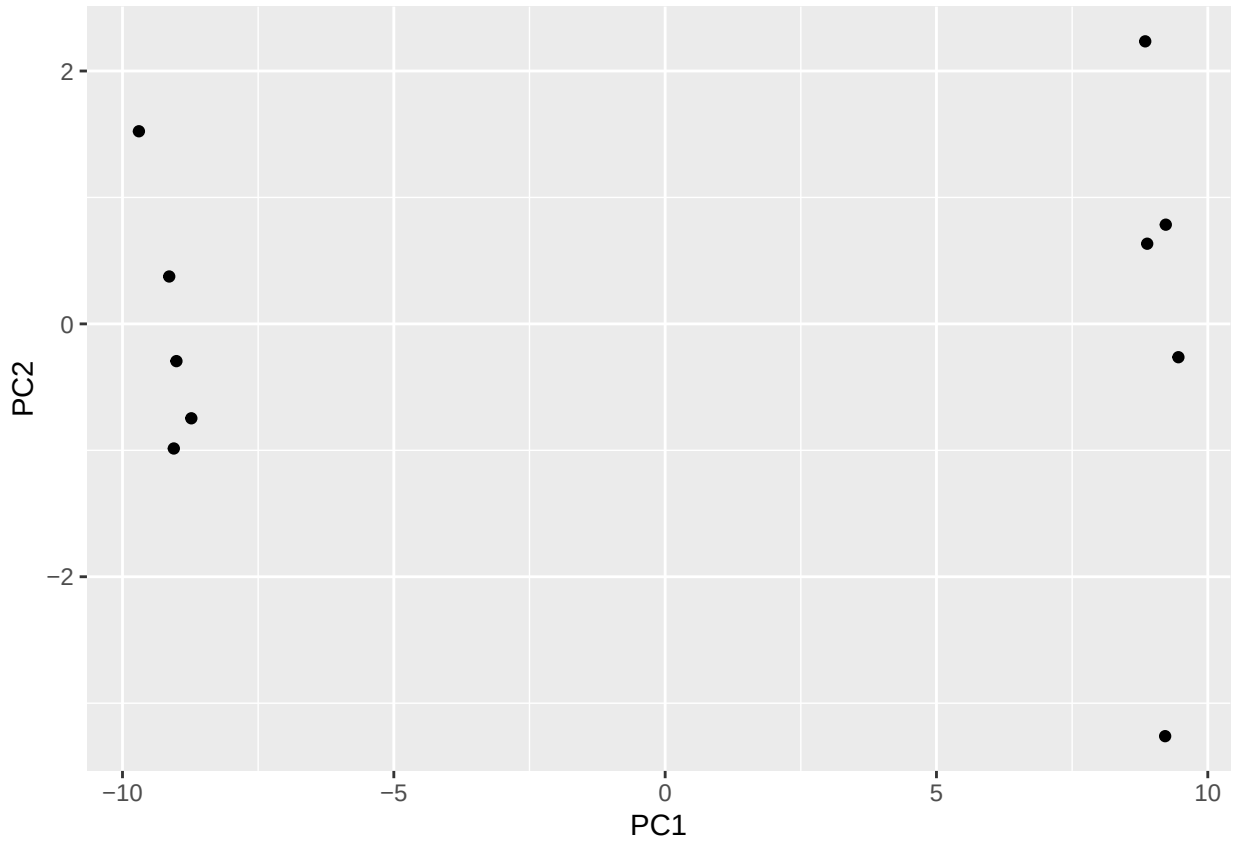
Make the PCA plot more presentable:

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

plot(pca2$x[,1], pca2$x[,2], col=colvec, pch=16,
     xlab=paste0("PC1 (", pca.var.per[1], "%)"),
     ylab=paste0("PC2 (", pca.var.per[2], "%"))

text(pca2$x[,1], pca2$x[,2], labels = colnames(rna.data), pos=c(rep(4,5), rep(2,5)))
```

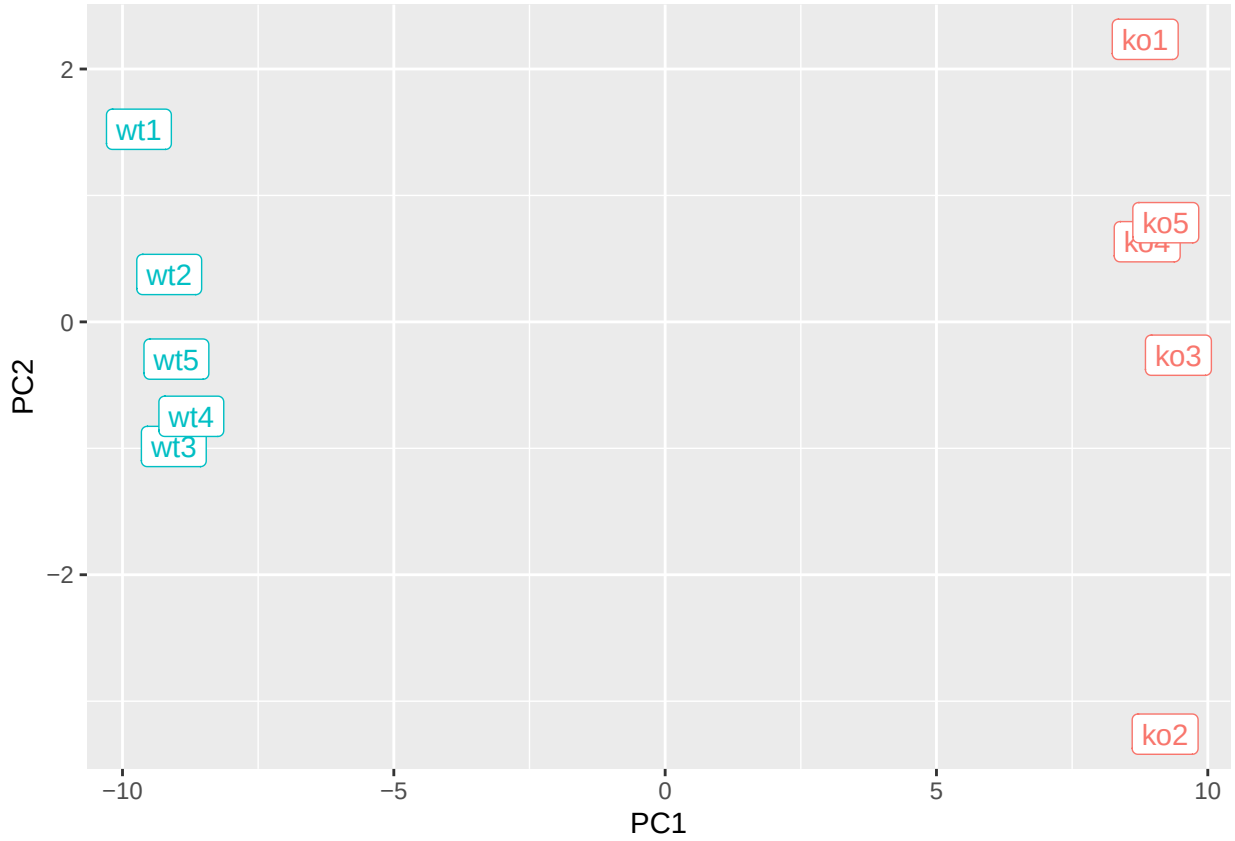


Use ggplot2

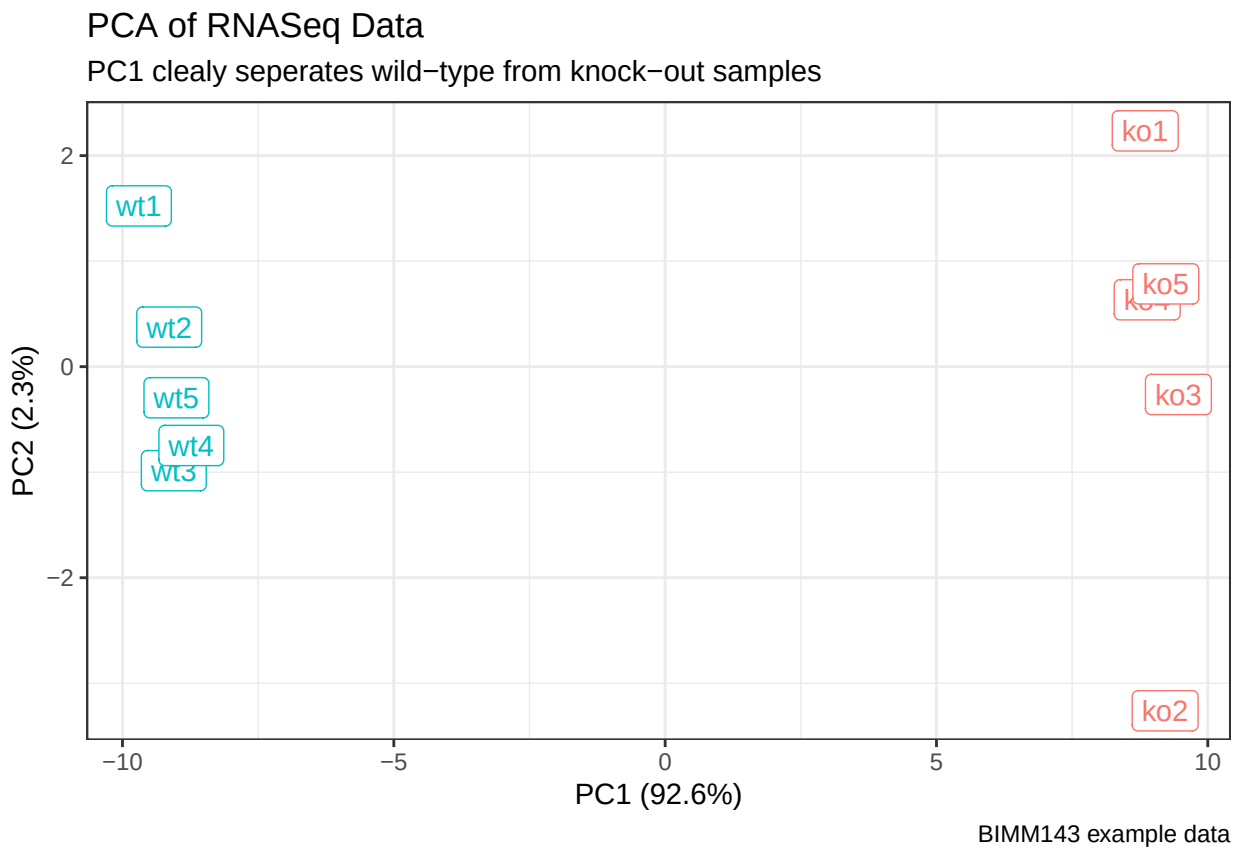
```
# Load ggplot2
library(ggplot2)

# Define a data frame containing the PCA data
df <- as.data.frame(pca2$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
```




```
# Polish the plot
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()
```



Optional: find the 10 genes that contribute the most to variance in PC1:

```
loading_scores <- pca2$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"
```

```
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19042)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] ggplot2_3.3.5
##
## loaded via a namespace (and not attached):
##  [1] pillar_1.7.0     compiler_4.1.2   highr_0.9        tools_4.1.2
##  [5] digest_0.6.29    evaluate_0.14    lifecycle_1.0.1  tibble_3.1.6
##  [9] gtable_0.3.0     pkgconfig_2.0.3  rlang_1.0.1      cli_3.1.1
## [13] yaml_2.2.2       xfun_0.29        fastmap_1.1.0    withr_2.4.3
## [17] stringr_1.4.0    dplyr_1.0.7      knitr_1.37       generics_0.1.2
## [21] vctrs_0.3.8      grid_4.1.2       tidyselect_1.1.1 glue_1.6.1
## [25] R6_2.5.1         fansi_1.0.2      rmarkdown_2.11   purrr_0.3.4
## [29] farver_2.1.0     magrittr_2.0.2   scales_1.1.1     ellipsis_0.3.2
## [33] htmltools_0.5.2  colorspace_2.0-2 labeling_0.4.2   utf8_1.2.2
## [37] stringi_1.7.6    munsell_0.5.0    crayon_1.4.2
```