Class 07: Machine Learning 1

Amy Nguyen (PID: A18148284)

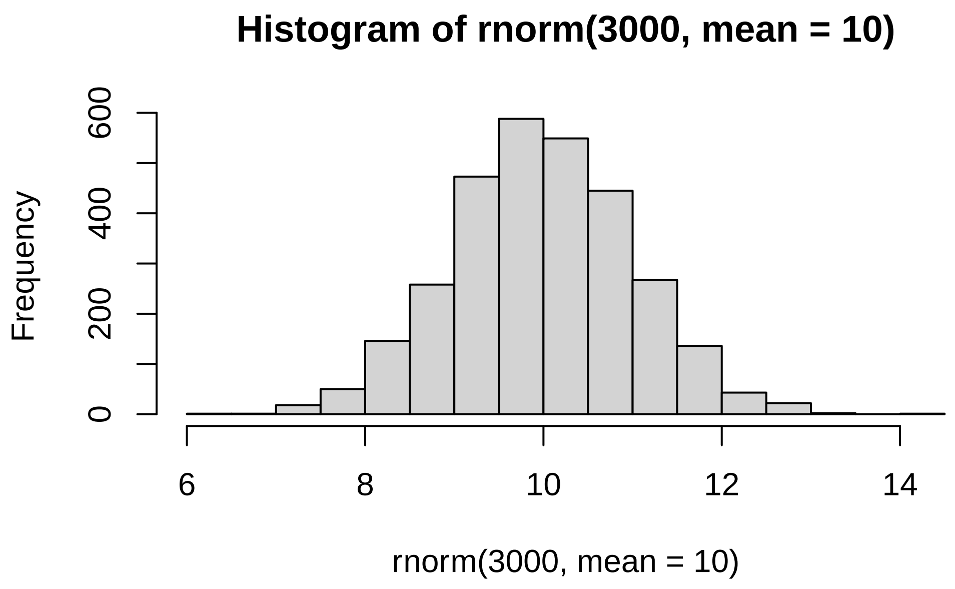
Table of contents

Background

Today we will be begin our re-exploration of important machine learning methods with a focus on **clustering** and **dimensionality reduction**

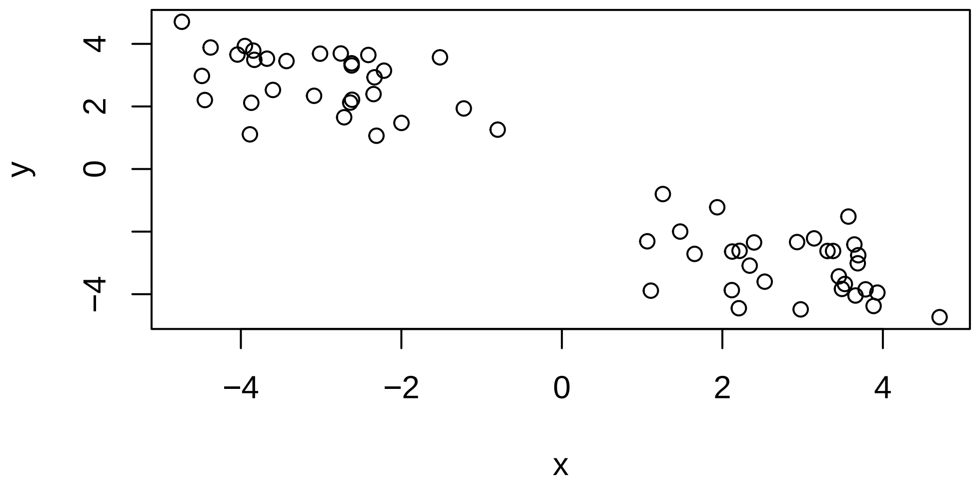
To start testing these methods let's make up some sample data to cluster where we know what the answer should be.

```r
hist(rnorm(3000, mean=10))
```



Q. Can you generate 30 numbers centered at +3 taken at random from a normal distribution?

```
tmp <- c(rnorm(30, mean=3),
         rnorm(30, mean=-3) )

x <- cbind(x=tmp, y=rev(tmp))
plot(x)
```



K-means clustering

The main function in "base R" for K-means clustering is called `kmeans()`, lets try it out:

```r
k <- kmeans(x, centers = 2)
k
```

```
K-means clustering with 2 clusters of sizes 30, 30

Cluster means:
          x         y
1 -3.046488  2.839540
2  2.839540 -3.046488

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] 56.28266 56.28266
 (between_SS / total_SS =  90.2 %)
```

```
Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

Q. What component of your kmeans result object has the cluster centers?

```r
k$centers
```

```
          x         y
1 -3.046488  2.839540
2  2.839540 -3.046488
```

Q. What component of your kmeans result object has the cluster size (i.e. how many points are in each cluster)?

```r
k$size
```

```
[1] 30 30
```

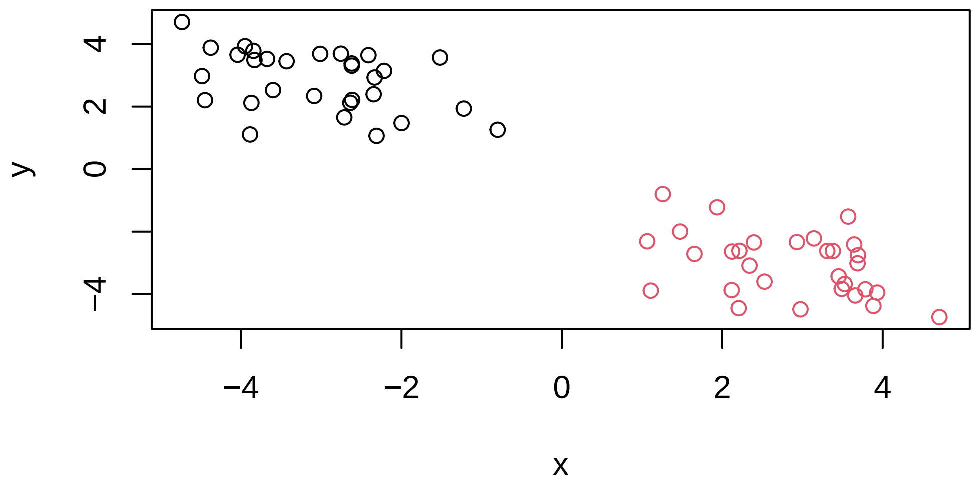
Q. What component of your kmeans result object has the cluster membership vector (i.e. the main clustering result: which points are in which cluster)?

```r
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
```

Q. Plot the results of clustering (i.e. our data colored by the clustering result) along with the cluster centers.

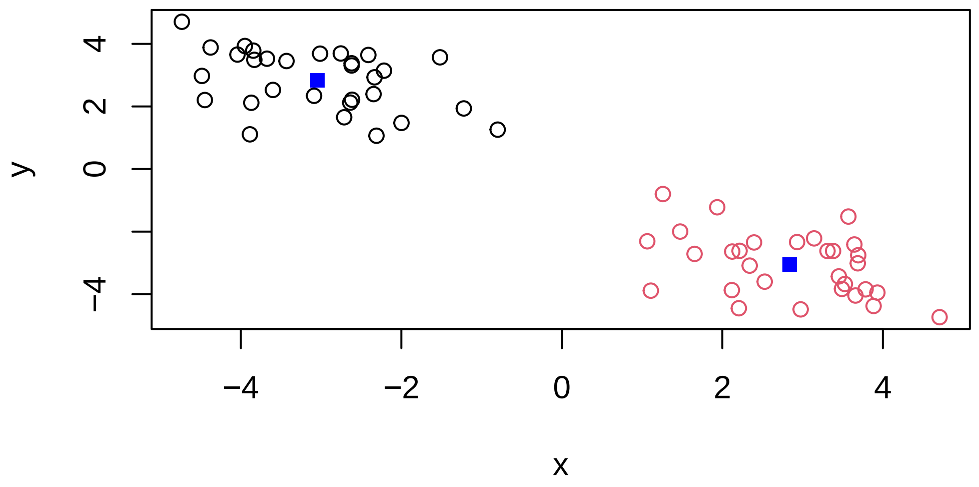
```r
plot(x, col=k$cluster)
```



```
#points(k$centers, col="blue", pch=15, cex=2)
```

Q. Can you run kmeans again and cluster into 4 clusters and plot the results just like we did above with coloring by cluster and the cluster centers shown in blue?

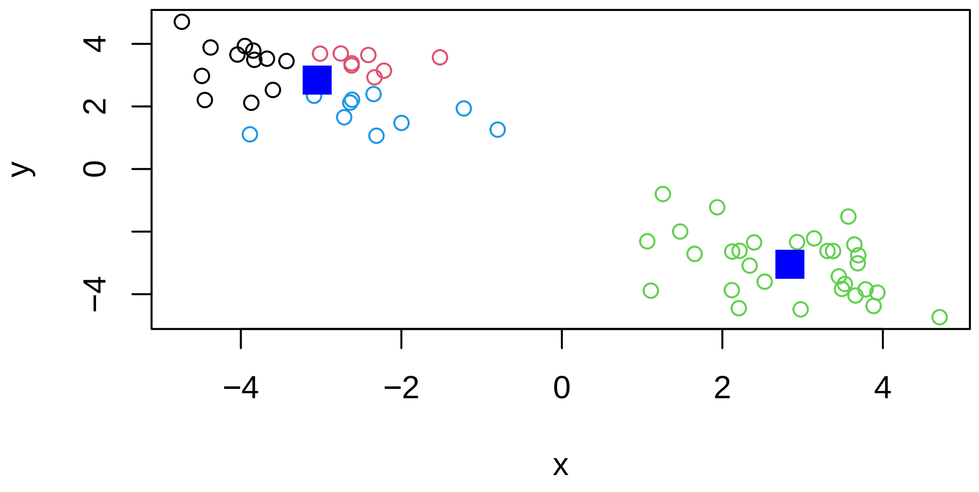
```
plot(x, col=k$cluster)
points(k$centers, col="blue", pch=15)
```



Q. Run kmeans() again and this time produce 4 clusters and call your result object 'k4' and make a results figure like above?

```r
k4 <- kmeans(x, center = 4)
```

```r
plot(x, col=k4$cluster)
points(k$centers, col="blue", pch=15, cex=2)
```



> **Key-point:** Kmeans will always return the clustering that we ask for (this is the "K" or "centers" in K-means)!

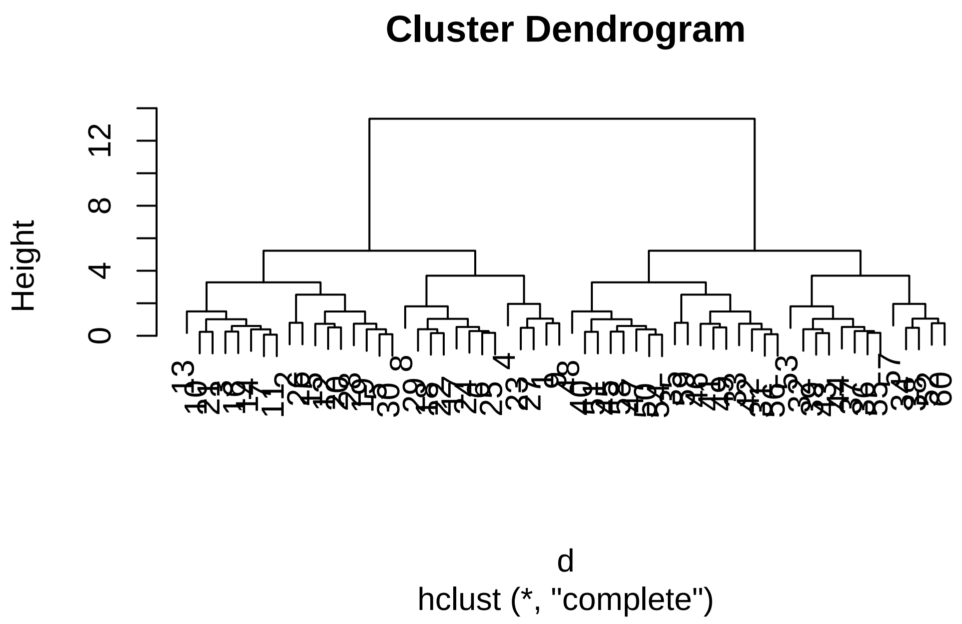
```r
k$tot.withinss
```

```
[1] 112.5653
```

Hierarchial clustering

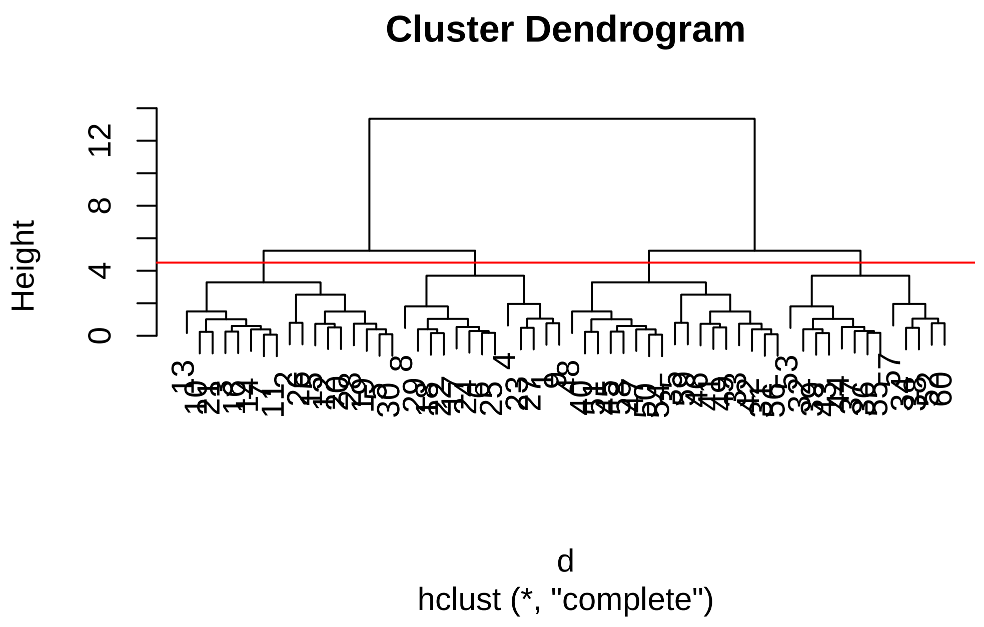
The main function for hierarchical clustering in base R is called `hclust()`. One of the main differences with respect to the `kmeans()` function is that you can not just pass your input data directly to `hclust()` - it needs a "distance matrix" as input. We can get this from lots of places including the `dist()` function.

```r
d <- dist(x)
hc <- hclust(d)
plot(hc)
```



We can “cut” the dendrogram or “tree” at a given height to yield our “clusters”. For this we use the function `cutree()`

```
plot(hc)
abline(h=4.5, col="red")
```



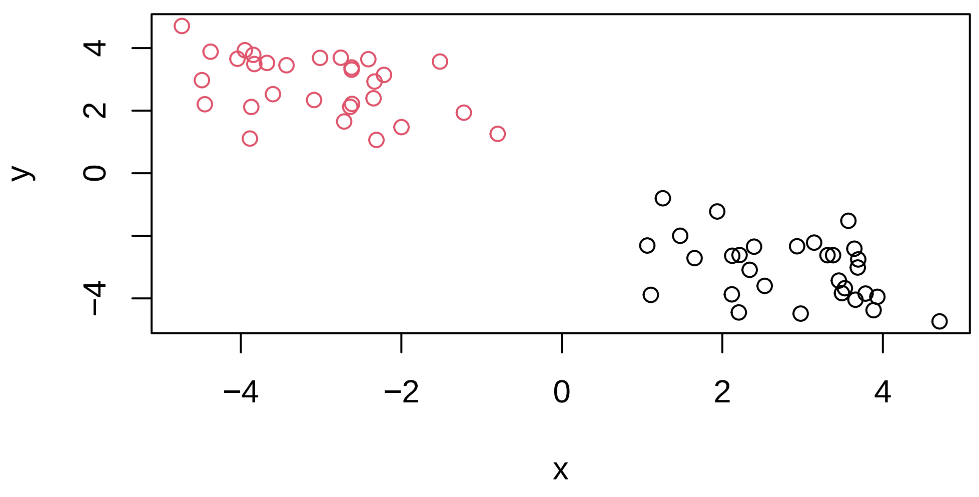
```r
grps <- cutree(hc, h=10)
```

```r
grps
```

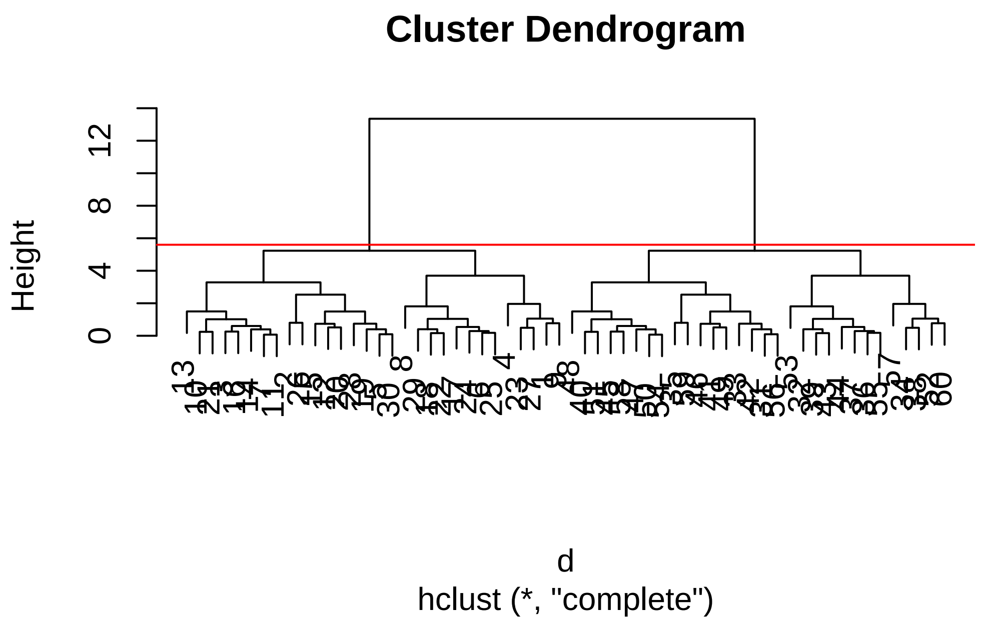
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Q. Plot our data `x` colored by the clustering result from `hclust()` and `cutree()`?

```r
plot(x, col=grps)
```



```r
plot(hc)
abline(h=5.6, col="red")
```



```
grps <- cutree(hc, h=5.6)
```

Principal Component Analysis (PCA)

PCA is a popular dimensional reduction technique that is widely used in bioinformatics.

PCA of UK food data

Read data on food consumption in the UK

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

```
7       Fresh_potatoes      720   874      566       1033
8             Fresh_Veg      253   265      171        143
9             Other_Veg      488   570      418        355
10 Processed_potatoes      198   203      220        187
11       Processed_Veg      360   365      337        334
12          Fresh_fruit     1102  1137      957        674
13              Cereals     1472  1582     1462       1494
14            Beverages       57    73       53         47
15          Soft_drinks     1374  1256     1572       1506
16     Alcoholic_drinks      375   475      458        135
17       Confectionery       54    64       62         41
```

It looks like the row names are not set properly. We can fix this.

```
rownames(x) <- x[,1]
x <- x[,-1]
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
Fresh_potatoes          720   874      566      1033
Fresh_Veg               253   265      171       143
Other_Veg               488   570      418       355
Processed_potatoes      198   203      220       187
Processed_Veg           360   365      337       334
Fresh_fruit            1102  1137      957       674
Cereals                1472  1582     1462      1494
Beverages                57    73       53        47
Soft_drinks            1374  1256     1572      1506
Alcoholic_drinks        375   475      458       135
Confectionery            54    64       62        41
```

A better way to do this is fix the row names assignment at import time:

```
read.csv(url, row.names = 1)
```

```
                    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
Fresh_potatoes          720   874      566      1033
Fresh_Veg               253   265      171       143
Other_Veg               488   570      418       355
Processed_potatoes      198   203      220       187
Processed_Veg           360   365      337       334
Fresh_fruit            1102  1137      957       674
Cereals                1472  1582     1462      1494
Beverages                57    73       53        47
Soft_drinks            1374  1256     1572      1506
Alcoholic_drinks        375   475      458       135
Confectionery            54    64       62        41
```

> 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 colums.

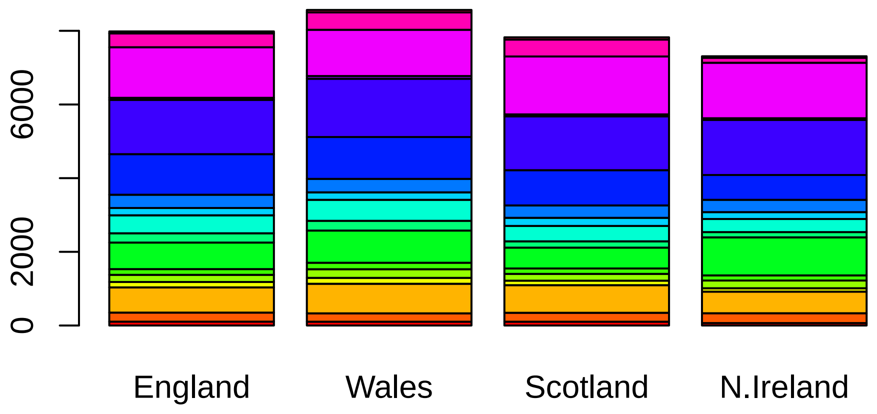
```
dim(x)
```

```
[1] 17  4
```

> 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? I prefer the read.csv(url, row.names = 1)

> Q3: Changing what optional argument in the above barplot() function results in the following plot? Changing beside=T to beside=F, or deleting it.

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

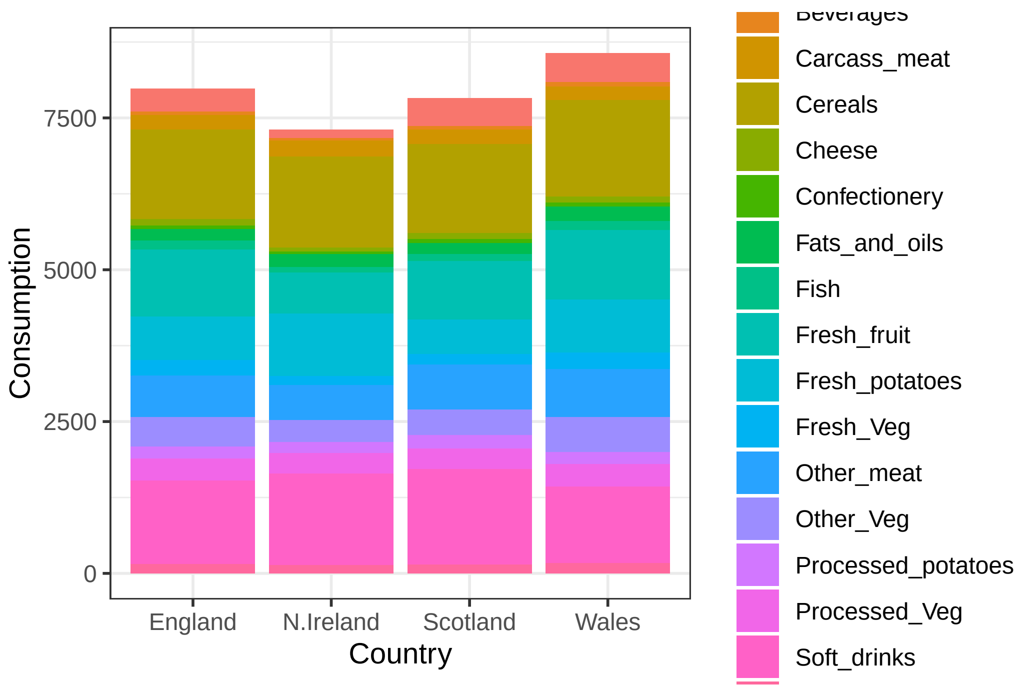


```r
library(tidyr)

# Convert data to long format for ggplot with `pivot_longer()`
x_long <- x |>
          tibble::rownames_to_column("Food") |>
          pivot_longer(cols = -Food,
                       names_to = "Country",
                       values_to = "Consumption")
dim(x_long)
```

```
[1] 68  3
```

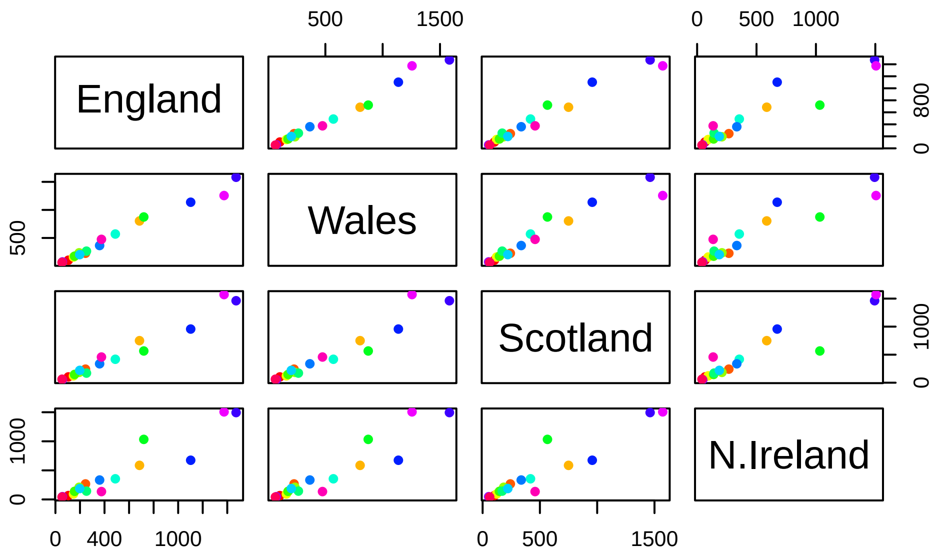
Q4: Changing what optional argument in the above ggplot() code results in a stacked barplot figure? Change dodge to stack

```r
library(ggplot2)

ggplot(x_long) +
  aes(x = Country, y = Consumption, fill = Food) +
  geom_col(position = "stack") +
  theme_bw()
```



Q5: We can use the pairs() function to generate all pairwise plots for our countries. 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? A point on the diagonal of a given plot represents a value where the variable on the x-axis is equal to the variable on the y-axis. Each country is being compared to/plotted against itself, so the off-diagnoal plots show relationships between different countries.

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

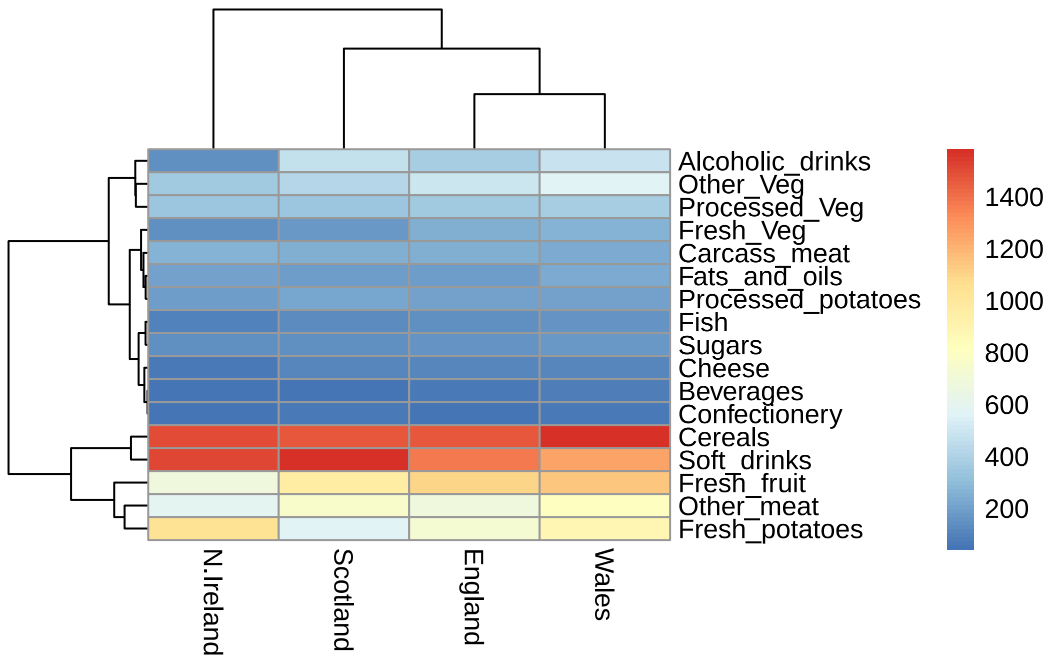


Heatmap

We can instaill the **pheatmap** package with the `install.packages()` command that we used previously. Remember that we always run this in the console and not a code chunk in our quarto document.

> Q6. Based on the pairs and heatmap figures, which countries cluster together and what does this suggest about their food consumption patterns? Can you easily tell what the main differences between N. Ireland and the other countries of the UK in terms of this data-set? England, Scotland, and Wales cluster together, suggesting similar food consumption patterns. Based on the color intensity of the heat map, it can be observed that N.Ireleand has higher consumption of fresh fruit, fresh potatoes while lower consumption of alcoholic drinks compared to the other UK countries.

```r
library(pheatmap)
pheatmap( as.matrix(x) )
```



Of all these plots really only the `pairs()` plot was useful. This however took a bit of work to interpret and will not scale when I am looking at much bigger data sets.

PCA the rescue

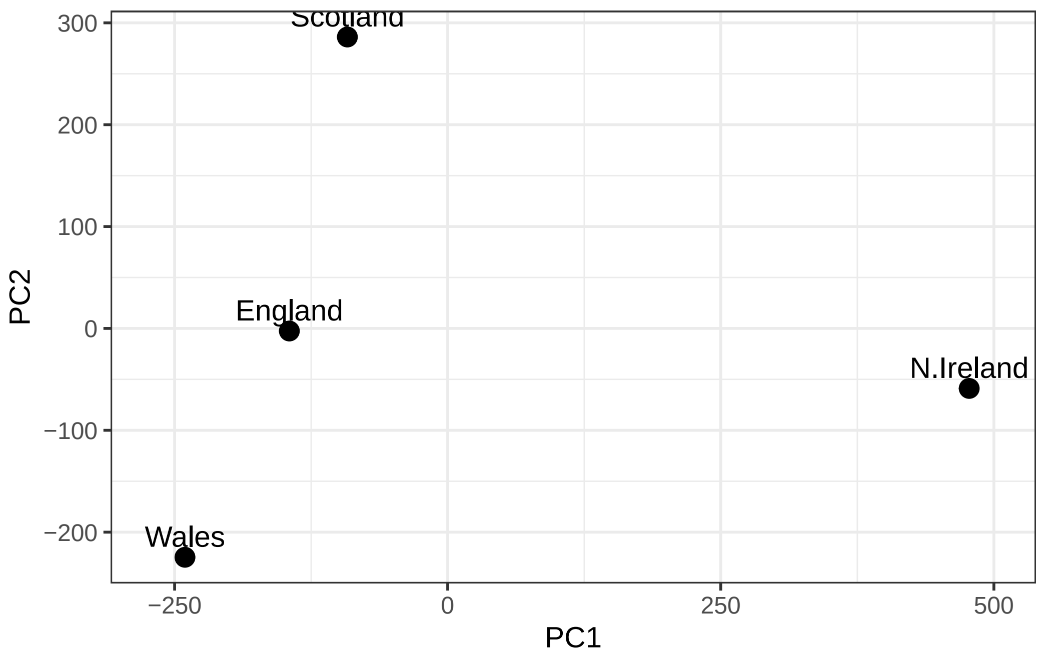
The main function in "base R" for PCA is called `prcomp()`.

```
pca <- prcomp( t(x) )
summary(pca)
```

```
Importance of components:
                            PC1      PC2      PC3     PC4
Standard deviation     324.1502 212.7478 73.87622 2.7e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.0e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.0e+00
```

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

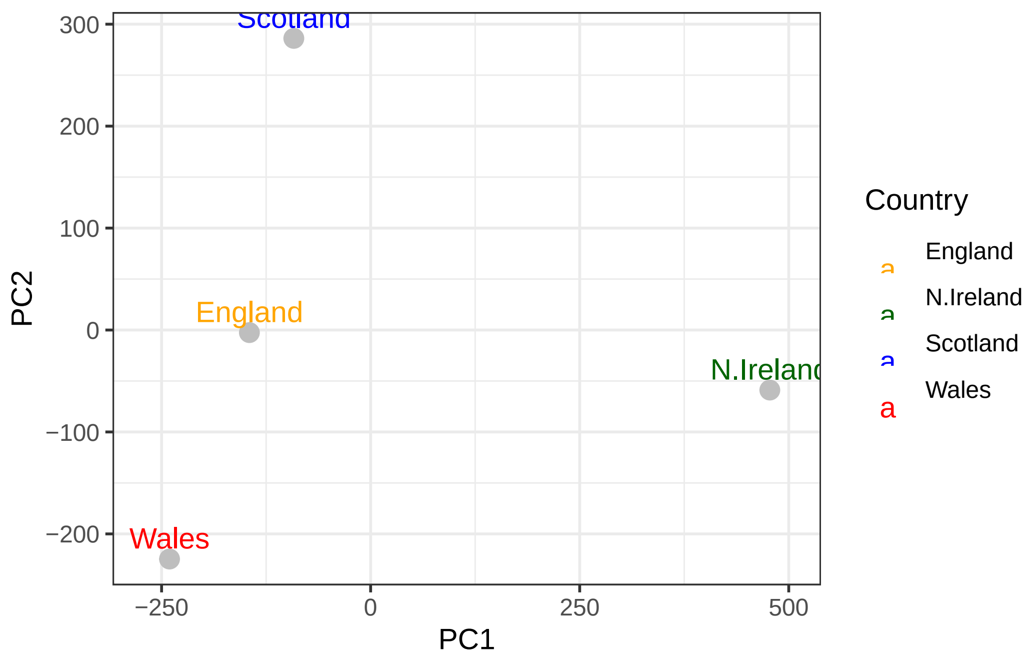
```
# Create a data frame for plotting
df <- as.data.frame(pca$x)
df$Country <- rownames(df)

# Plot PC1 vs PC2 with ggplot
ggplot(pca$x) +
  aes(x = PC1, y = PC2, label = rownames(pca$x)) +
  geom_point(size = 3) +
  geom_text(vjust = -0.5) +
  xlim(-270, 500) +
  xlab("PC1") +
  ylab("PC2") +
  theme_bw()
```



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.

```
# Create a data frame for plotting
df <- as.data.frame(pca$x)
df$Country <- rownames(df)

my_cols <- c("Wales"="red", "England"="orange", "Scotland"="blue", "N.Ireland"="darkgreen")
```

```r
# Plot PC1 vs PC2 with ggplot
ggplot(df, aes(x = PC1, y = PC2, label = Country, color = Country)) +
  geom_point(size = 3, col="grey") +
  geom_text(vjust = -0.5) +
  scale_color_manual(values = my_cols) +
  xlim(-270, 500) +
  xlab("PC1") +
  ylab("PC2") +
  theme_bw()
```



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

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

```r
z <- summary(pca)
z$importance
```

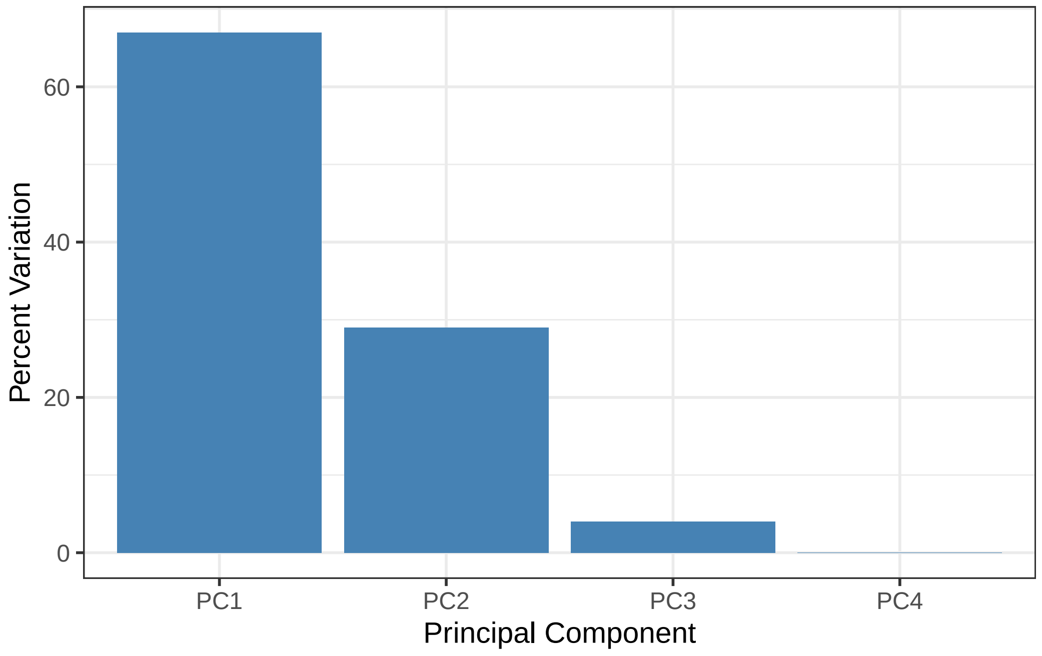
```
                             PC1       PC2      PC3          PC4
Standard deviation     324.15019 212.74780 73.87622 2.699876e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

Q. How much variance is captured in the first PC? 67.4%

Q. How many PCs do I need to capture at least 90% of the total variance in the dataset? Two PCs capture 96.5% of the total variance.

Q. Plot our main PCA result. Folks can call this different things depending on their field of study e.g. "PC plot", "ordienation plot", "Score plot", "PC1 vs. PC2 plot"...

```r
# Create scree plot with ggplot
variance_df <- data.frame(
  PC = factor(paste0("PC", 1:length(v)), levels = paste0("PC", 1:length(v))),
  Variance = v
)

ggplot(variance_df) +
  aes(x = PC, y = Variance) +
  geom_col(fill = "steelblue") +
  xlab("Principal Component") +
  ylab("Percent Variation") +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 0))
```



```r
attributes(pca)
```

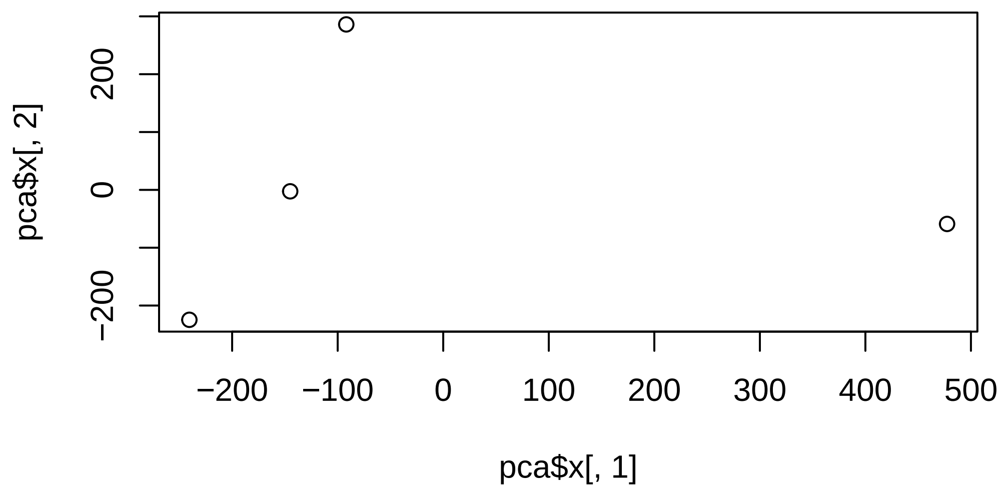
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

To generate our PCA score plot we want the `pca$x` component of the result object.

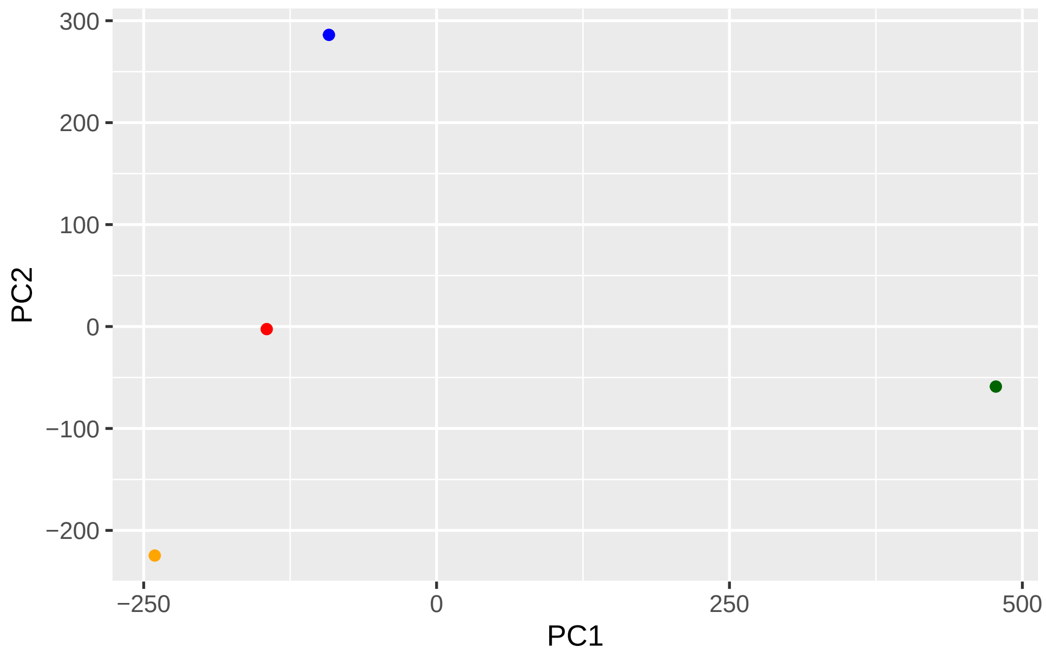
```r
pca$x
```

```
                 PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945  1.612425e-14
Wales     -240.52915 -224.646925 -56.475555  4.751043e-13
Scotland   -91.86934  286.081786 -44.415495 -6.044349e-13
N.Ireland  477.39164  -58.901862  -4.877895  1.145386e-13
```

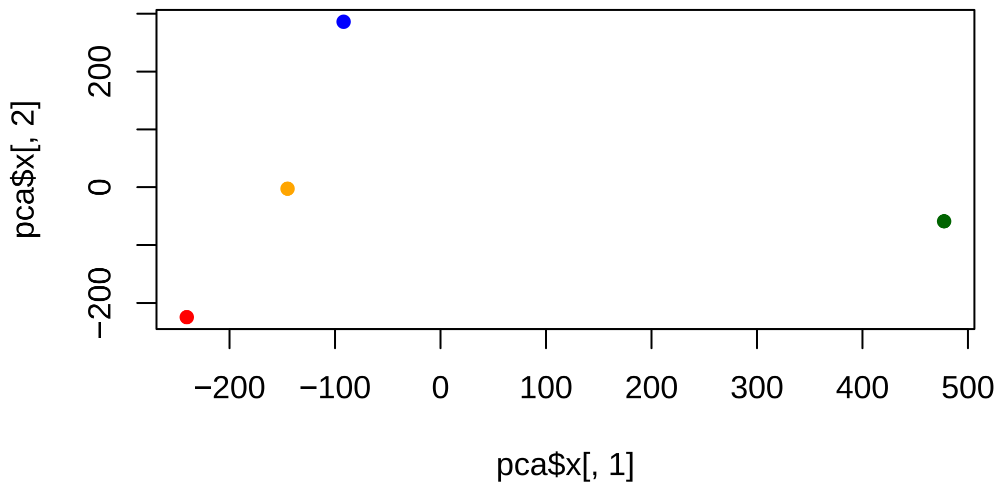
```r
plot(pca$x[,1], pca$x[,2])
```



```r
library(ggplot2)

ggplot(pca$x) +
  aes(PC1, PC2) +
  geom_point(col=my_cols)
```



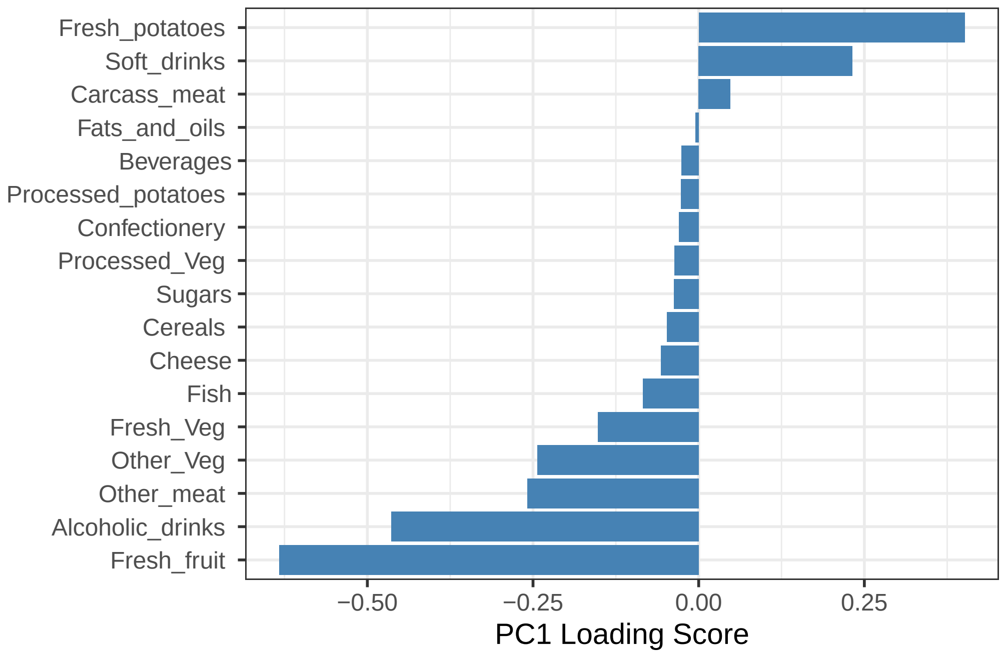
```r
my_cols <- c("orange", "red", "blue", "darkgreen")
plot(pca$x[,1], pca$x[,2], col=my_cols, pch=16)
```



Digging deeper (variable loadings)

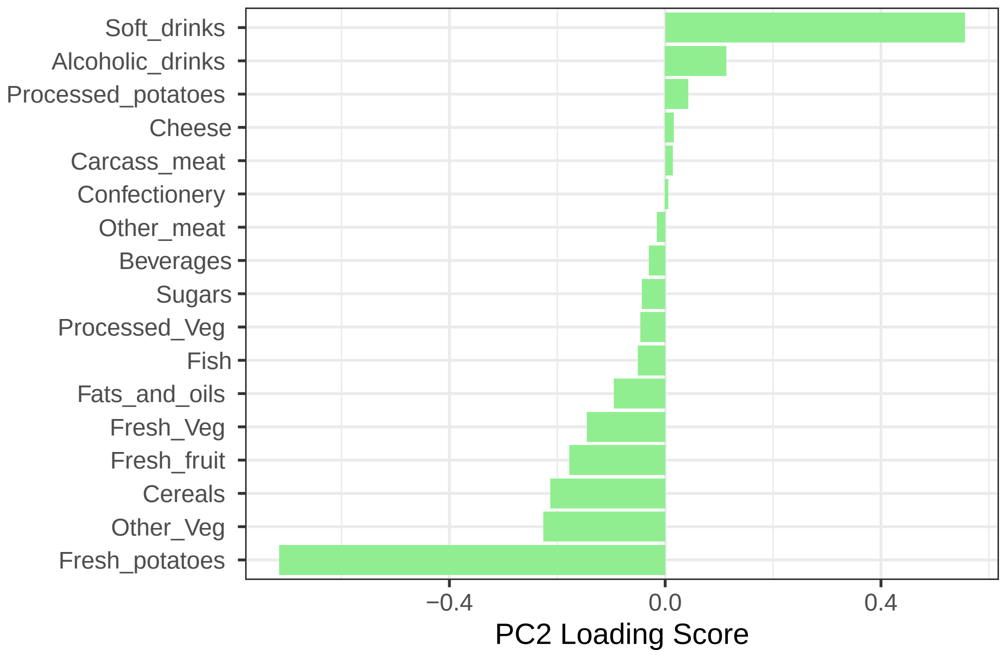
How do the original variables (i.e., the 17 different foods) contribute to our new PCs?

```
## Lets focus on PC1 as it accounts for > 90% of variance
ggplot(pca$rotation) +
  aes(x = PC1,
      y = reorder(rownames(pca$rotation), PC1)) +
  geom_col(fill = "steelblue") +
  xlab("PC1 Loading Score") +
  ylab("") +
  theme_bw() +
  theme(axis.text.y = element_text(size = 9))
```



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

```
ggplot(pca$rotation) +
  aes(x = PC2,
      y = reorder(rownames(pca$rotation), PC2)) +
  geom_col(fill = "lightgreen") +
  xlab("PC2 Loading Score") +
  ylab("") +
  theme_bw() +
  theme(axis.text.y = element_text(size = 9))
```



PCA of RNA-seq data

```r
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
```

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

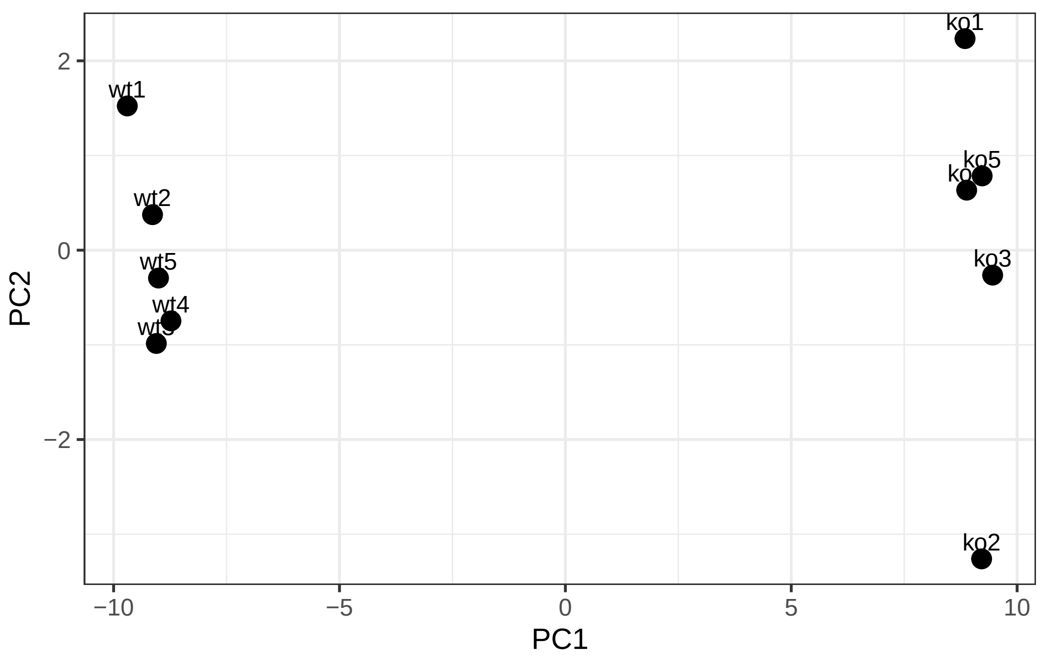
```
[1] 100
```

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

```
[1] 10
```

> Q10: How many genes and samples are in this data set? How many PCs do you think it will take to have a useful overview of this data set (see below)? 100 genes and 10 samples. Based on the PC1 vs PC2 plot, the first two PCs already seem to separate the two samples meaningfully.

```r
## Again we have to take the transpose of our data
pca <- prcomp(t(rna.data), scale=TRUE)

# Create data frame for plotting
df <- as.data.frame(pca$x)
df$Sample <- rownames(df)

## Plot with ggplot
ggplot(df) +
  aes(x = PC1, y = PC2, label = Sample) +
  geom_point(size = 3) +
  geom_text(vjust = -0.5, size = 3) +
  xlab("PC1") +
  ylab("PC2") +
  theme_bw()
```

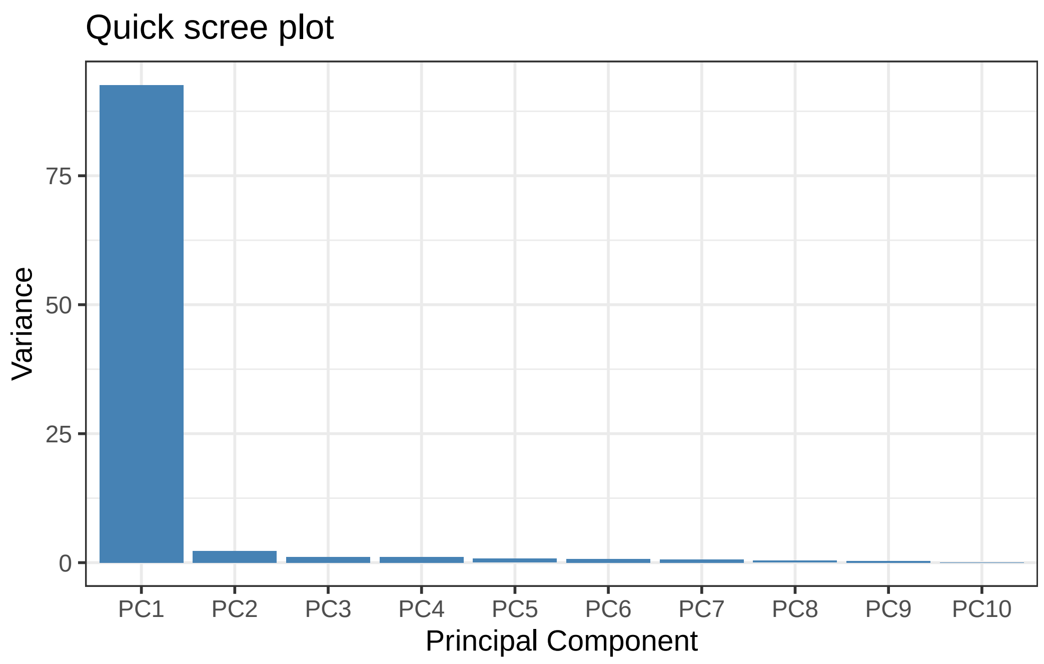


```r
summary(pca)
```

```
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 3.39e-15
Proportion of Variance 0.00385 0.00364 0.00e+00
Cumulative Proportion  0.99636 1.00000 1.00e+00
```

```r
# Calculate variance explained
pca.var <- pca$sdev^2
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)

# Create scree plot data
scree_df <- data.frame(
  PC = factor(paste0("PC", 1:10), levels = paste0("PC", 1:10)),
  Variance = pca.var[1:10]
)
```

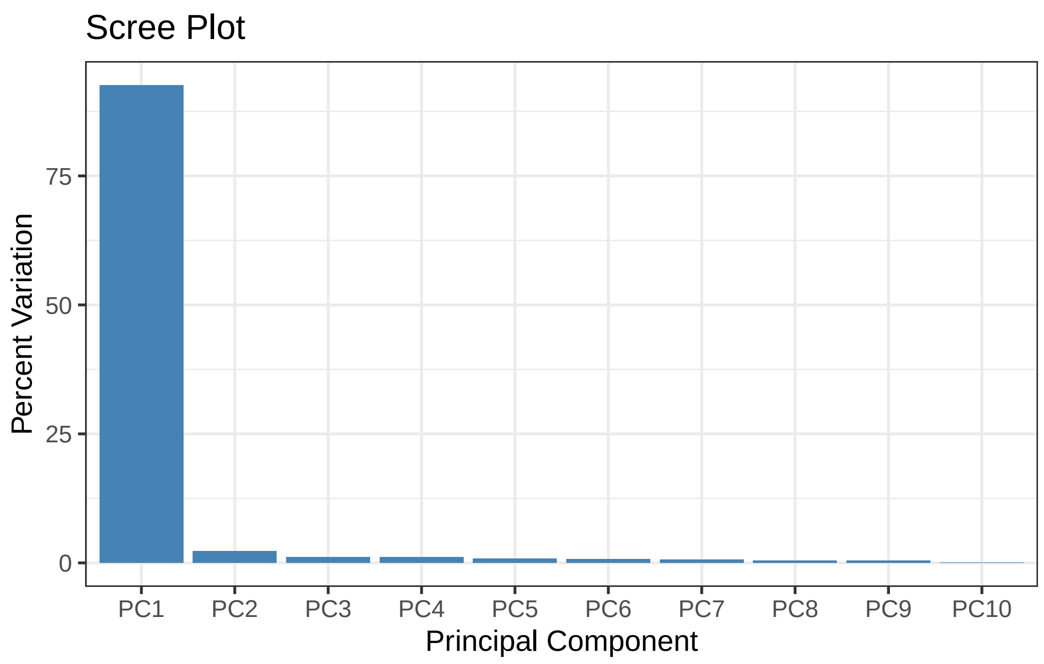
```r
ggplot(scree_df) +
  aes(x = PC, y = Variance) +
  geom_col(fill = "steelblue") +
  ggtitle("Quick scree plot") +
  xlab("Principal Component") +
  ylab("Variance") +
  theme_bw()
```



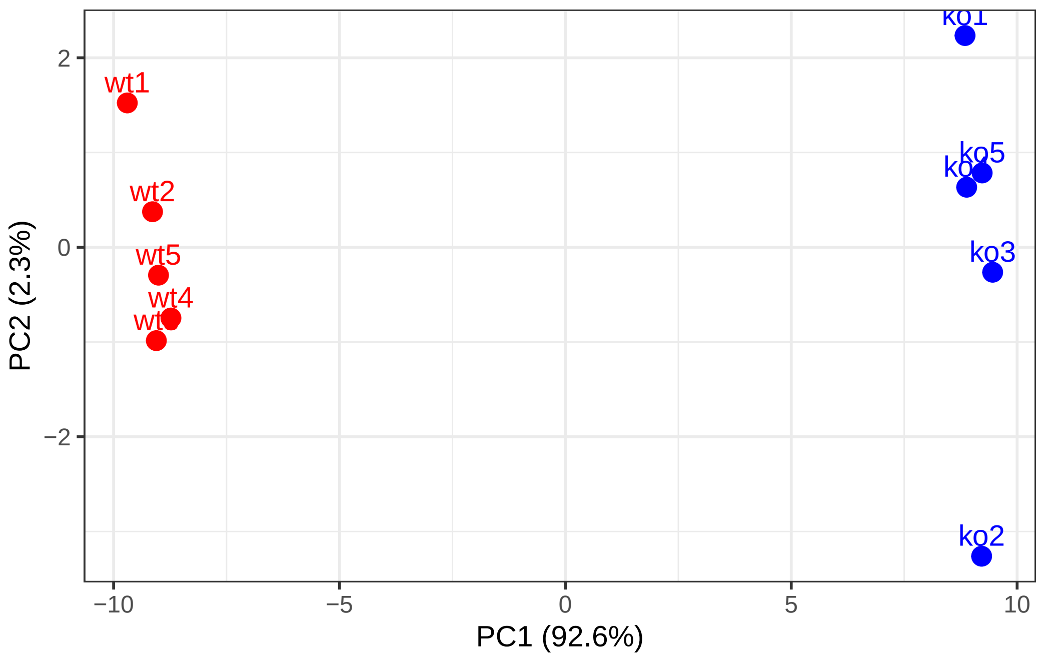
```r
## Percent variance is often more informative to look at
pca.var.per
```

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

```r
# Create percent variance scree plot
scree_pct_df <- data.frame(
  PC = factor(paste0("PC", 1:10), levels = paste0("PC", 1:10)),
  PercentVariation = pca.var.per[1:10]
)

ggplot(scree_pct_df) +
  aes(x = PC, y = PercentVariation) +
  geom_col(fill = "steelblue") +
  ggtitle("Scree Plot") +
  xlab("Principal Component") +
  ylab("Percent Variation") +
  theme_bw()
```



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

# Add condition to data frame
df$condition <- substr(df$Sample, 1, 2)
df$color <- colvec

ggplot(df) +
  aes(x = PC1, y = PC2, color = color, label = Sample) +
  geom_point(size = 3) +
  geom_text(vjust = -0.5, hjust = 0.5, show.legend = FALSE) +
  scale_color_identity() +
  xlab(paste0("PC1 (", pca.var.per[1], "%)")) +
  ylab(paste0("PC2 (", pca.var.per[2], "%)")) +
  theme_bw()
```


ko1
wt1
ko5
ko4
wt2
ko3
wt5
wt4
wt3
ko2
2
0
−2
−10
−5
0
5
10
PC2 (2.3%)
PC1 (92.6%)