

Lab 7: Machine Learning 1

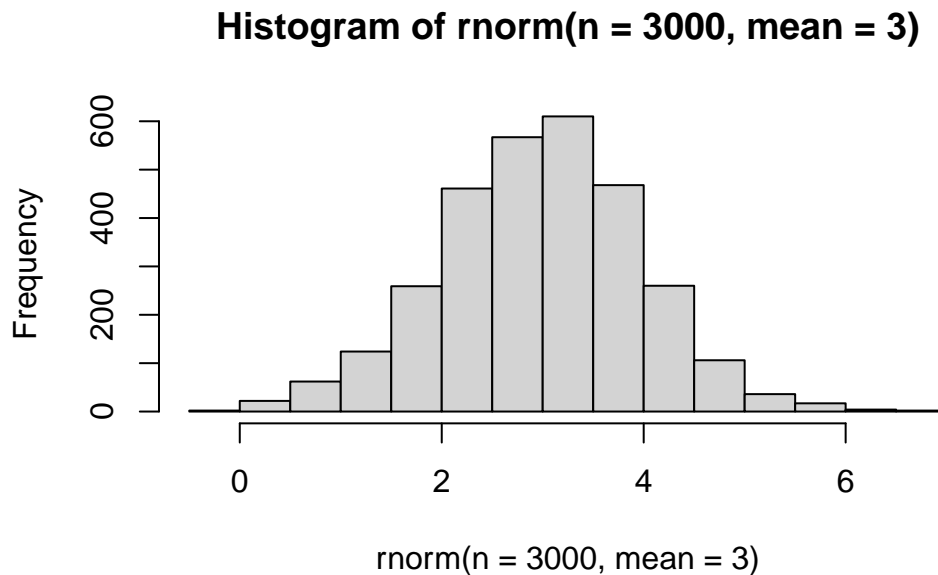
Trevor Hoang (A16371830)

Today we will explore unsupervised machine learning methods including clustering and dimensionality reduction methods.

Let's start by making up some data (Where we know there are clear groups) that we can use to test out different clustering methods.

We can use the `rnorm()` function to help us here:

```
hist(rnorm(n=3000, mean=3))
```



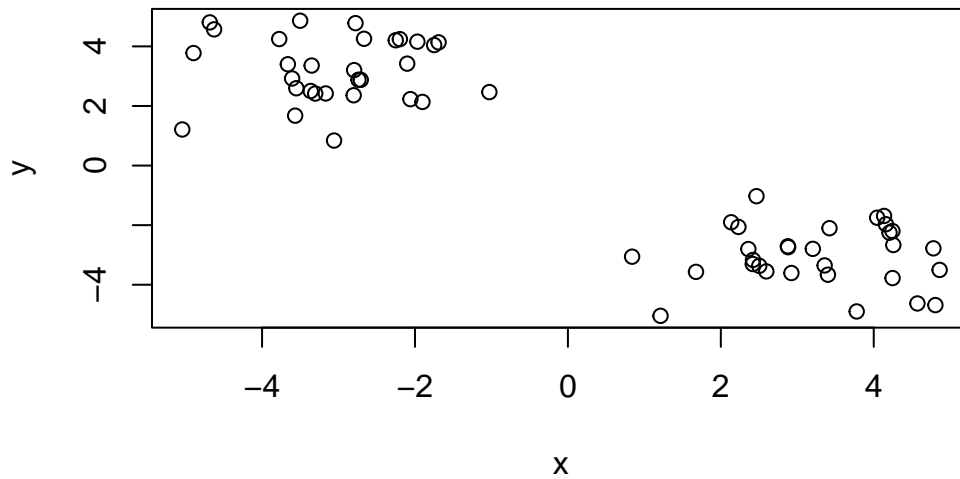
Make data with 2 “clusters”

```
x = c(rnorm(30, mean=-3),
      rnorm(30, mean=+3))

z= cbind(x=x,y=rev(x))
head(z)
```

```
      x      y
[1,] -1.968585 4.158959
[2,] -2.197726 4.244085
[3,] -2.802102 2.359855
[4,] -2.666248 4.255850
[5,] -2.251875 4.206102
[6,] -1.691315 4.135072
```

```
plot(z)
```



K-means clustering

The main function in “base” R for K-means clustering is called `kmeans()`

Q. How many points lie in each cluster?

k\$size

[1] 30 30

Q. What component of our results tells us about the cluster membership (i.e. which point lies in which cluster)?

```
k$cluster
```

[illegible]

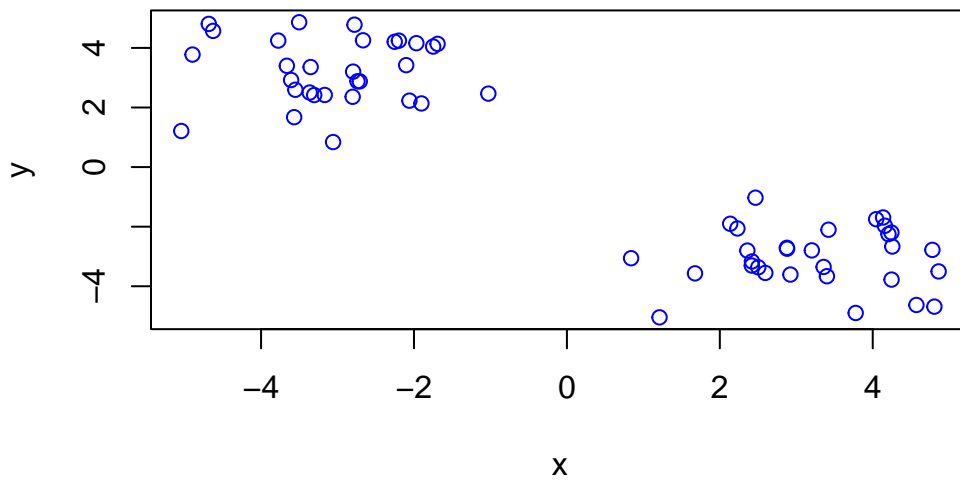
Q. Center of each cluster?

k\$centers

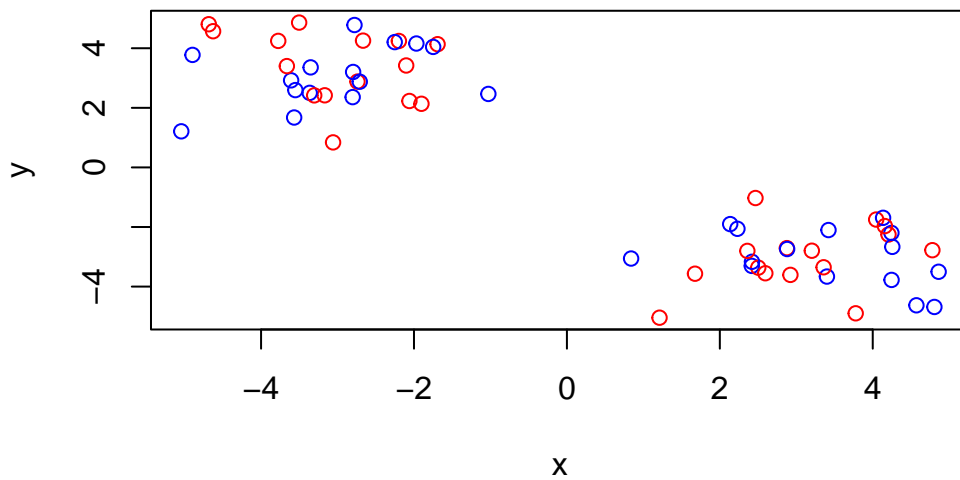
	x	y
1	-3.020213	3.233496
2	3.233496	-3.020213

Q. Put this result info together and make a little “base R” plot of our clustering result. Also add the cluster center points to this plot

```
plot(z, col="blue")
```

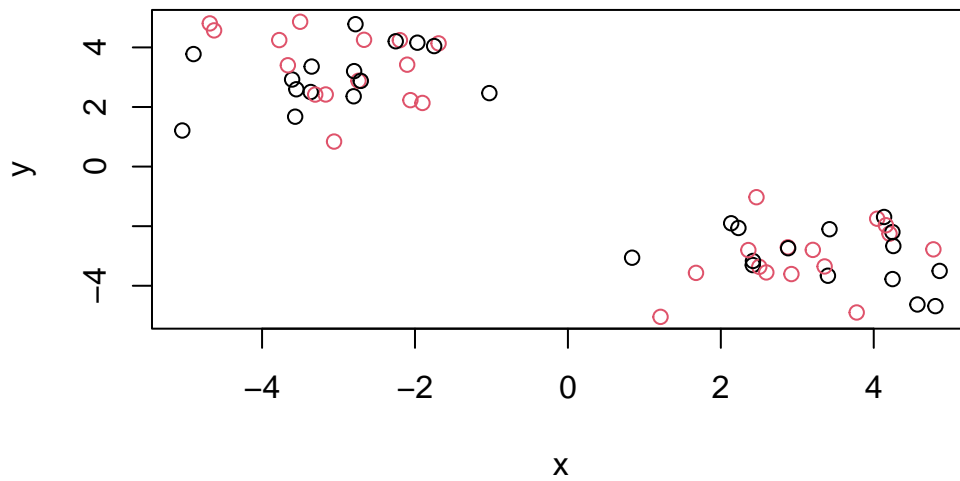


```
plot(z, col=c("blue", "red"))
```



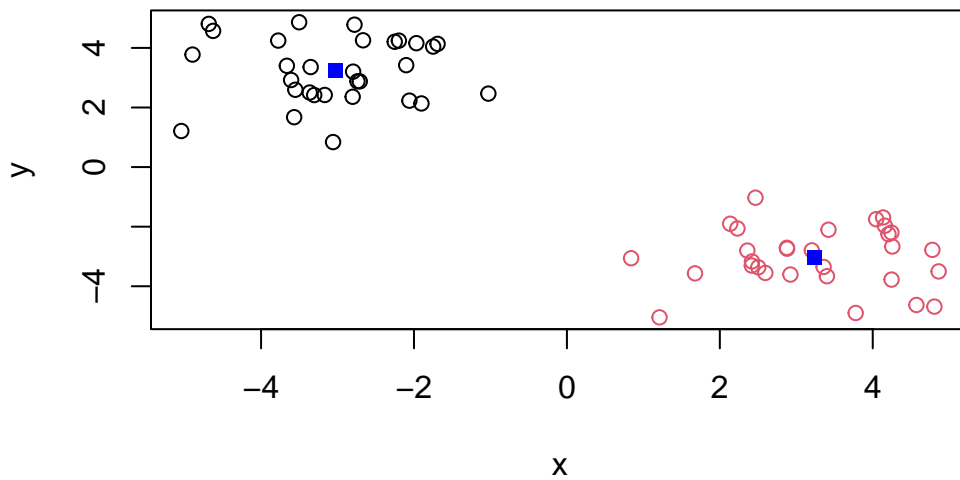
You can color by number.

```
plot(z, col=c(1, 2))
```



Plot colored by cluster membership:

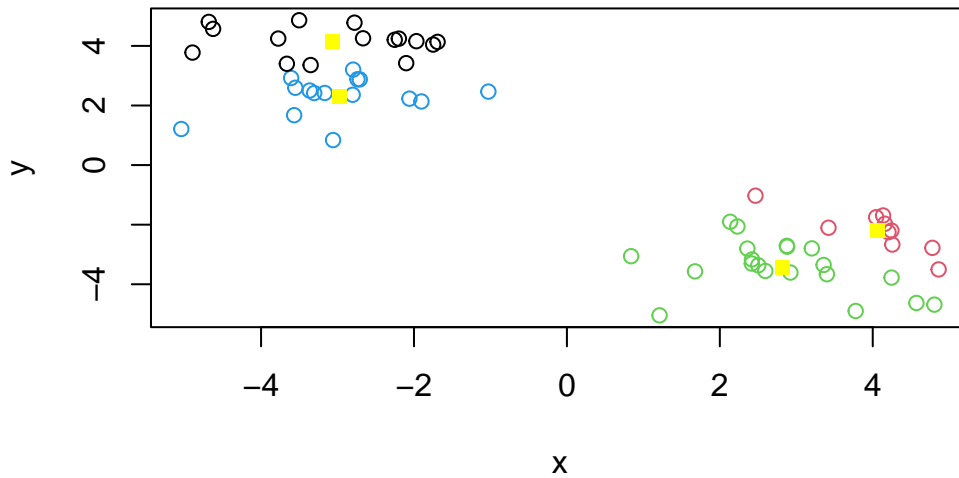
```
plot(z, col=k$cluster)
points(k$centers, col="blue", pch=15)
```



Q. Run kmeans on our input `z` and define 4 clusters making the same result visualization plot as above(plot of `z` colored by cluster membership).

```
k2=kmeans(z, centers=4)

plot(z, col=k2$cluster)
points(k2$centers, col="yellow", pch=15)
```



Hierarchical Clustering

The main function in base R for this is called `hclust()` it will take as input a distance matrix (key point is that you can't just give your "raw" data as input - you first have to calculate a distance matrix from your data).

```
d = dist(z)
hc = hclust(d)
hc
```

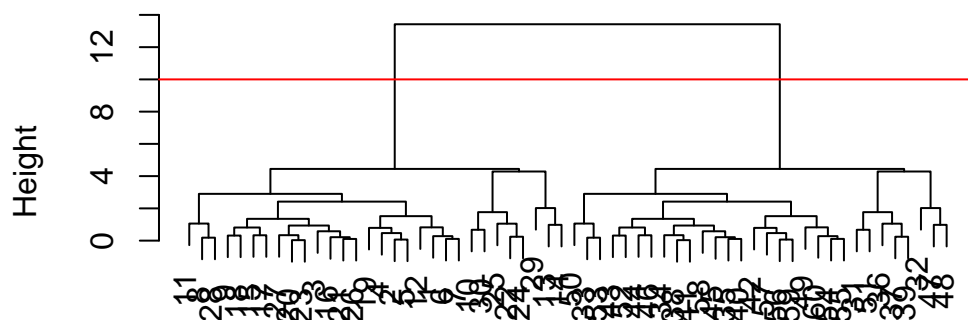
Call:

```
hclust(d = d)
```

```
Cluster method : complete
Distance       : euclidean
Number of objects: 60
```

```
plot(hc)
abline(h=10, col="red")
```


Cluster Dendrogram

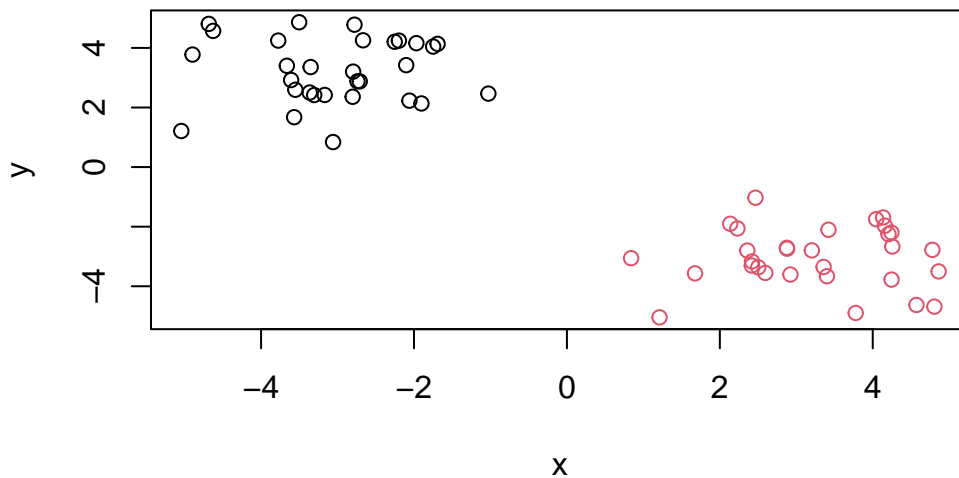


d
hclust (*, "complete")

Once I inspect the “tree” I can “cut” the tree to yield my groupings or clusters. The function to do this is called `cutree()`.

```
grps =cutree(hc, h=10)
```

```
plot(z, col=(grps))
```



##Hands on with Principal Component Analysis (PCA)

Let's examin some 17 dimensional data detailing food consumption in the US (England, Scotland, Wales and N. Ireland). Are these countries eating habits different or similar and if so how?

```
url = "https://tinyurl.com/UK-foods"
food = read.csv(url, row.names=1)
food
```

	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?

```
nrow(food)
```

```
[1] 17
```

```
ncol(food)
```

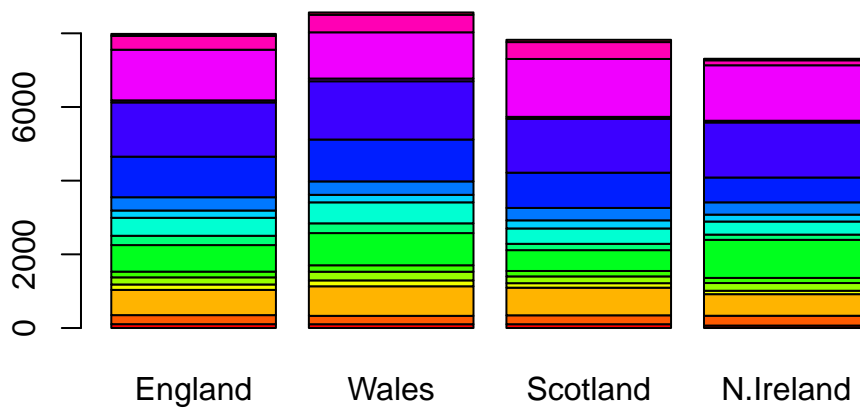
```
[1] 4
```

```
dim(food)
```

```
[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 dim function as it gives you both the rows and the columns in one command.

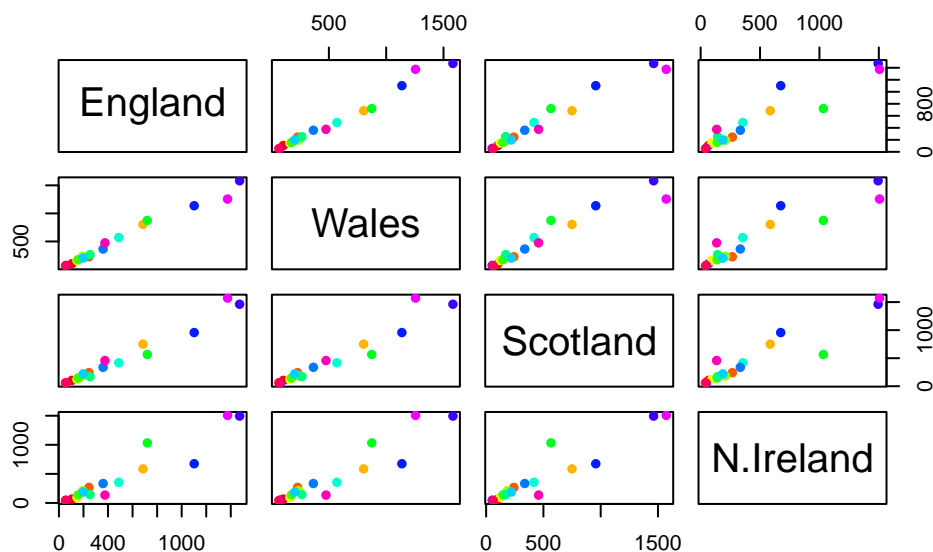
```
barplot(as.matrix(food), beside=F, col=rainbow(nrow(food)))
```



Q3: Changing what optional argument in the above `barplot()` function results in the following plot? Changing the `beside` argument allows you to get a stacked barplot.

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(food, col=rainbow(17), pch=16)
```



If the point lies on the straight line, there are similar amounts of the food category consumed in both countries.

Looking at these types of “pairwise plots” can be helpful but it does not scale well and kind of sucks! There must be a better way

Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set? The main differences between N. Ireland and the other countries is that they eat much less of the blue food than the other regions.

PCA To the rescue!

The main function for PCA in base R is called `prcomp()`. This function wants the transpose of our input data - i.e. the important foods as columns and the countries as rows.

```
pca =prcomp(t(food))
summary(pca)
```

Importance of components:

	PC1	PC2	PC3	PC4
Standard deviation	324.1502	212.7478	73.87622	3.176e-14
Proportion of Variance	0.6744	0.2905	0.03503	0.000e+00
Cumulative Proportion	0.6744	0.9650	1.00000	1.000e+00

Let's see what is in our PCA result object `pca`

```
attributes(pca)
```

```
$names
```

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

```
$class
```

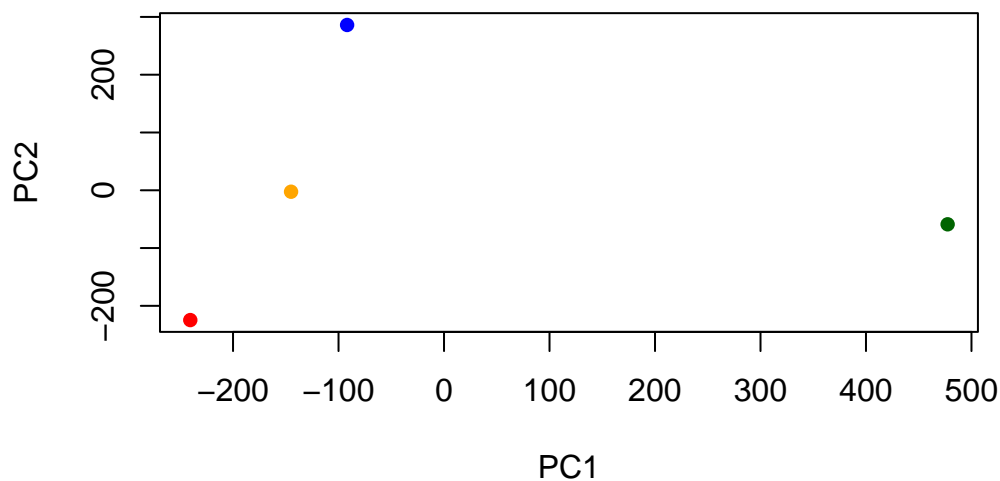
```
[1] "prcomp"
```

The `pca$x` result object is where we will focus first as this details how the countries are related to each other in terms of our new “axis” (aka “PCs”, “eigenvectors”, etc.)

```
head(pca$x)
```

	PC1	PC2	PC3	PC4
England	-144.99315	-2.532999	105.768945	-4.894696e-14
Wales	-240.52915	-224.646925	-56.475555	5.700024e-13
Scotland	-91.86934	286.081786	-44.415495	-7.460785e-13
N.Ireland	477.39164	-58.901862	-4.877895	2.321303e-13

```
plot(pca$x[,1],pca$x[,2], col=c("orange","red","blue","darkgreen"), pch=16,  
     xlab="PC1", ylab="PC2")
```



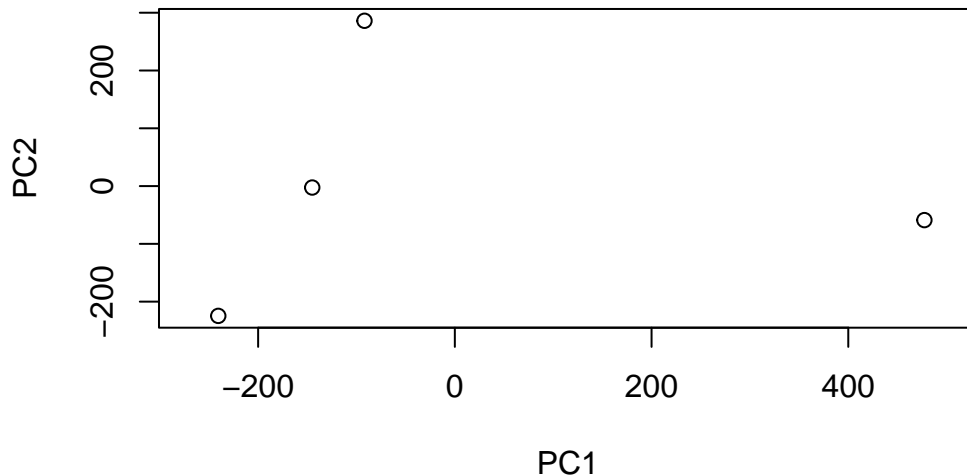
We can look at the so-called PC “loadings” result object to see how the original foods contribute to our new PCs (i.e. how the original variables contribute to our new better PC variables)

```
pca$rotation[,1]
```

Cheese	Carcass_meat	Other_meat	Fish
-0.056955380	0.047927628	-0.258916658	-0.084414983
Fats_and_oils	Sugars	Fresh_potatoes	Fresh_Veg
-0.005193623	-0.037620983	0.401402060	-0.151849942
Other_Veg	Processed_potatoes	Processed_Veg	Fresh_fruit
-0.243593729	-0.026886233	-0.036488269	-0.632640898
Cereals	Beverages	Soft_drinks	Alcoholic_drinks
-0.047702858	-0.026187756	0.232244140	-0.463968168
Confectionery			
-0.029650201			

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], col=c("orange", "red", "blue", "green"), colnames(x))
```



Calculating how much of the 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
```

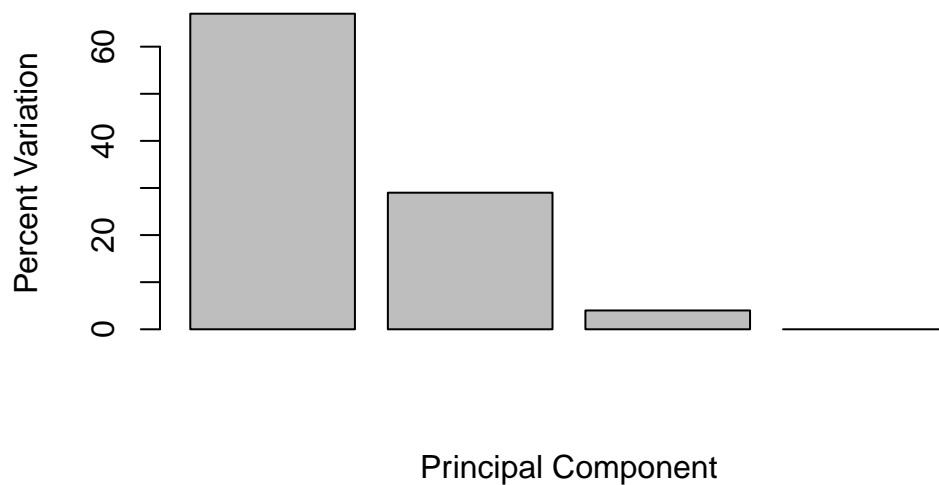
The second row

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

	PC1	PC2	PC3	PC4
Standard deviation	324.15019	212.74780	73.87622	3.175833e-14
Proportion of Variance	0.67444	0.29052	0.03503	0.000000e+00
Cumulative Proportion	0.67444	0.96497	1.00000	1.000000e+00

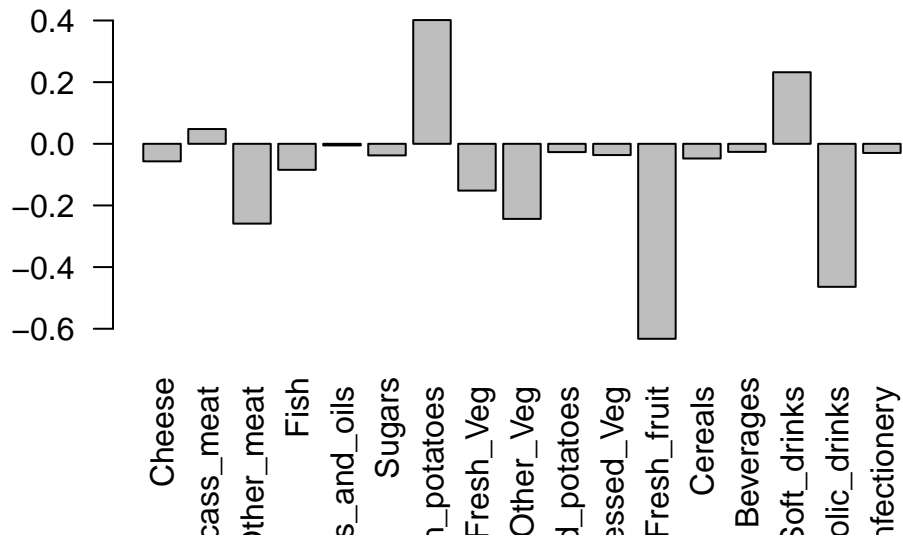
Plot of variances

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



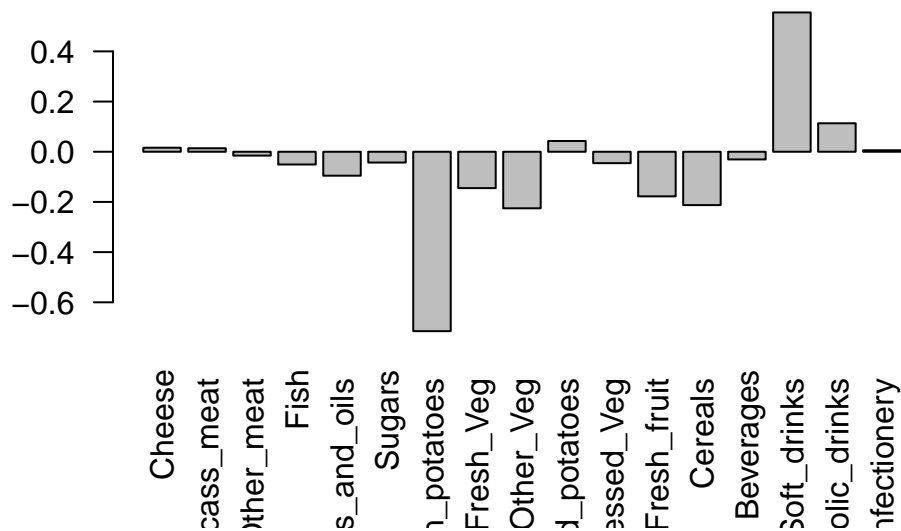
Creating a biplot to summarize how much of the original variables influence our PCs


```
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?

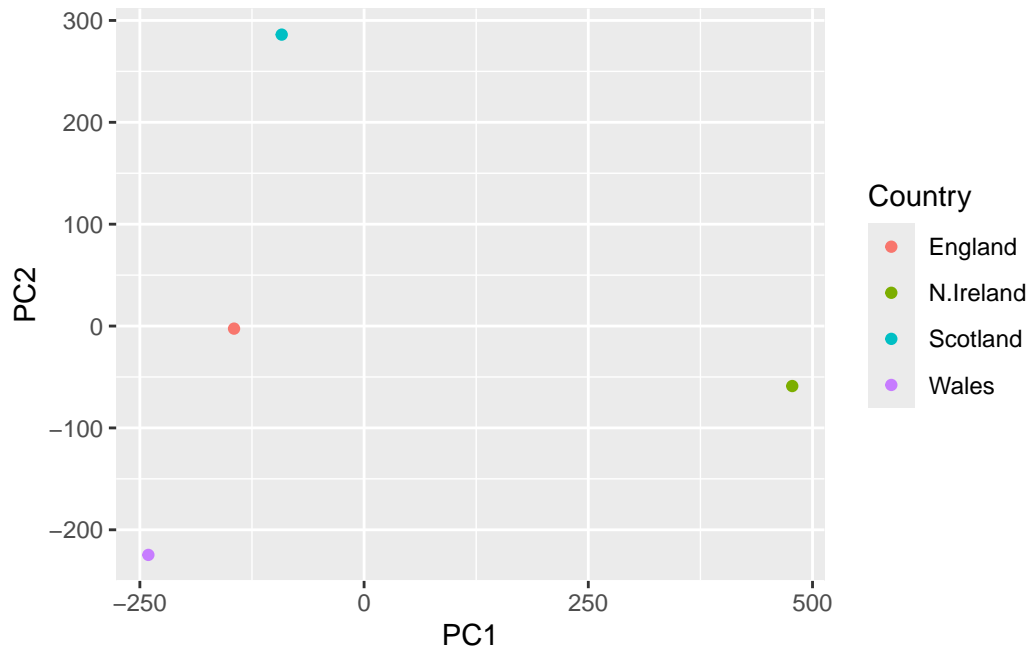
```
par(mar=c(10,3,0.35,0))
barplot(pca$rotation[,2], las=2)
```



Using ggplot2 for figures

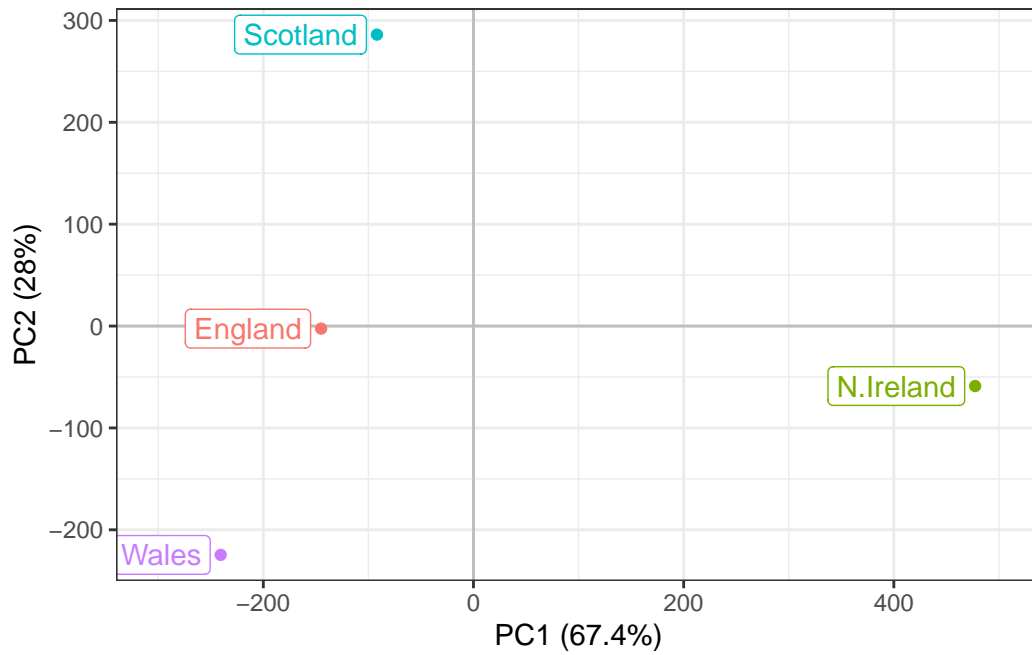
```
library(ggplot2)
df = as.data.frame(pca$x)
df_lab = tibble::rownames_to_column(df, "Country")

#First Basic Plot
ggplot(df_lab)+
  aes(PC1, PC2, col=Country)+
  geom_point()
```



Making the graph look nicer

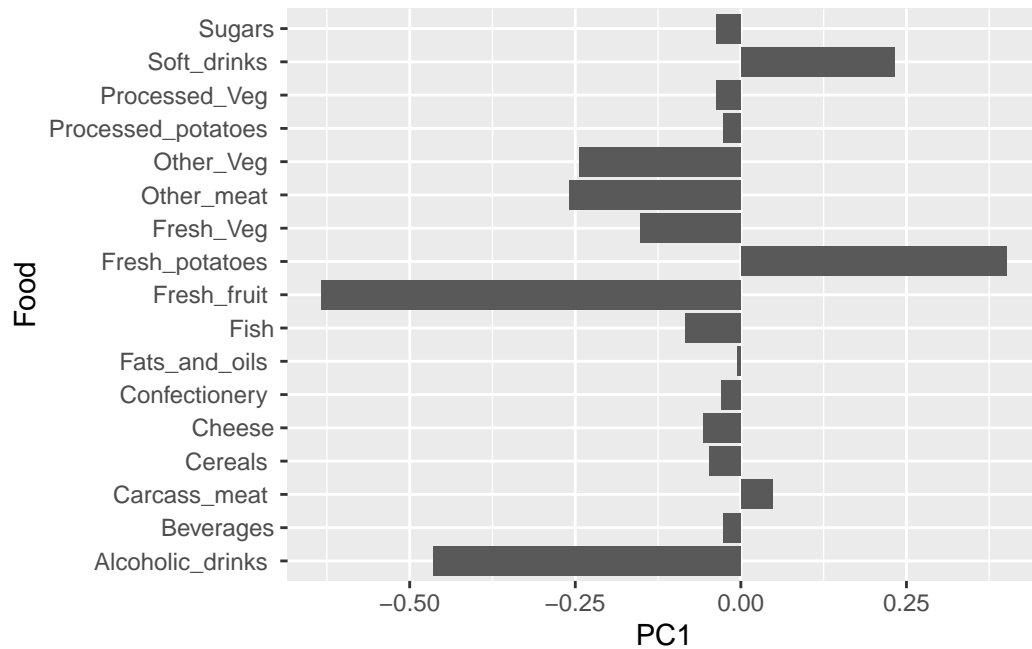
```
ggplot(df_lab) +  
  aes(PC1, PC2, col=Country, label=Country) +  
  geom_hline(yintercept = 0, col="gray") +  
  geom_vline(xintercept = 0, col="gray") +  
  geom_point(show.legend = FALSE) +  
  geom_label(hjust=1, nudge_x = -10, show.legend = FALSE) +  
  expand_limits(x = c(-300,500)) +  
  xlab("PC1 (67.4%)") +  
  ylab("PC2 (28%)") +  
  theme_bw()
```



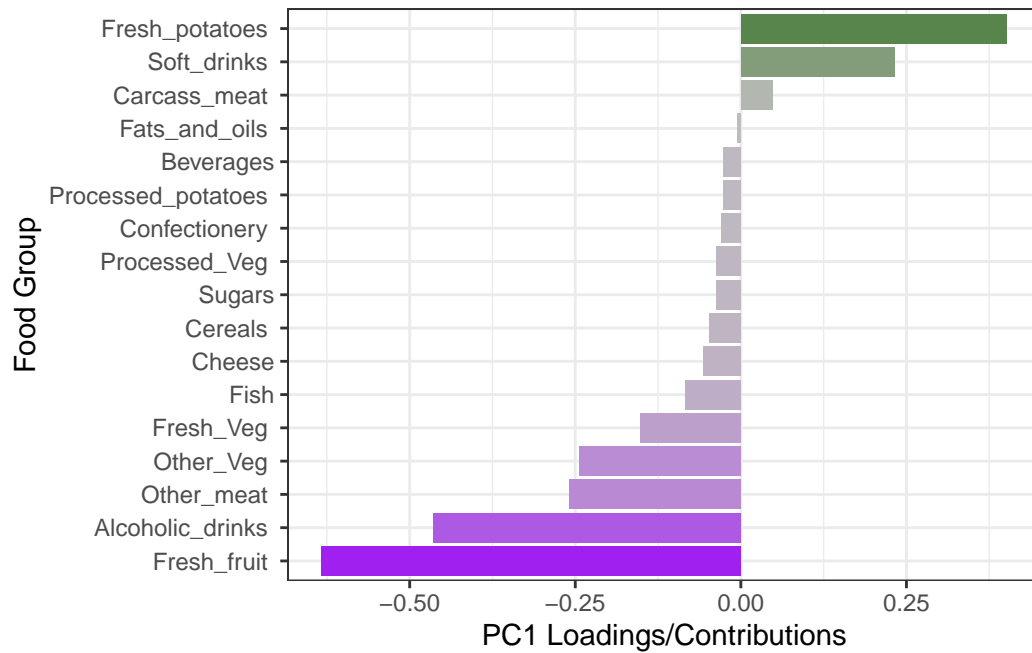
Recreating loadings and PC contribution figures

```
ld <- as.data.frame(pca$rotation)
ld_lab <- tibble::rownames_to_column(ld, "Food")

ggplot(ld_lab) +
  aes(PC1, Food) +
  geom_col()
```

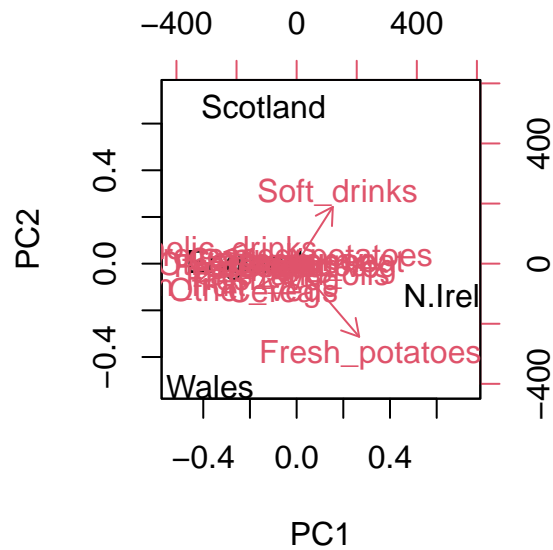


```
ggplot(ld_lab) +
  aes(PC1, reorder(Food, PC1), bg=PC1) +
  geom_col() +
  xlab("PC1 Loadings/Contributions") +
  ylab("Food Group") +
  scale_fill_gradient2(low="purple", mid="gray", high="darkgreen", guide=NULL) +
  theme_bw()
```



Creating biplots could be a nother way to see information together with main PCA plots.

```
biplot(pca)
```



##RNA-seq data

Reading and importing the data

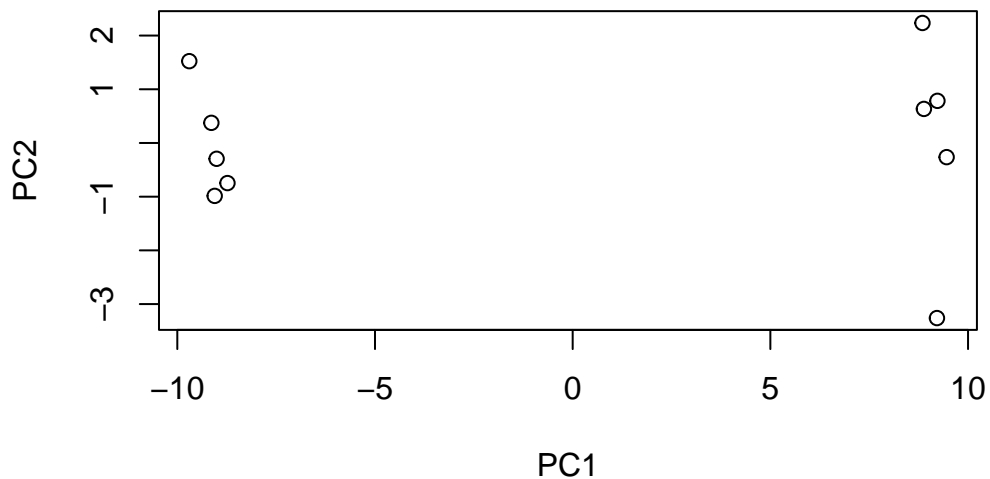
```
url2 <- "https://tinyurl.com/expression-CSV"
rna.data <- read.csv(url2, row.names=1)
head(rna.data)
```

	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? There are 6 genes and 10 samples.

Creating a PCA of the data and plotting

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



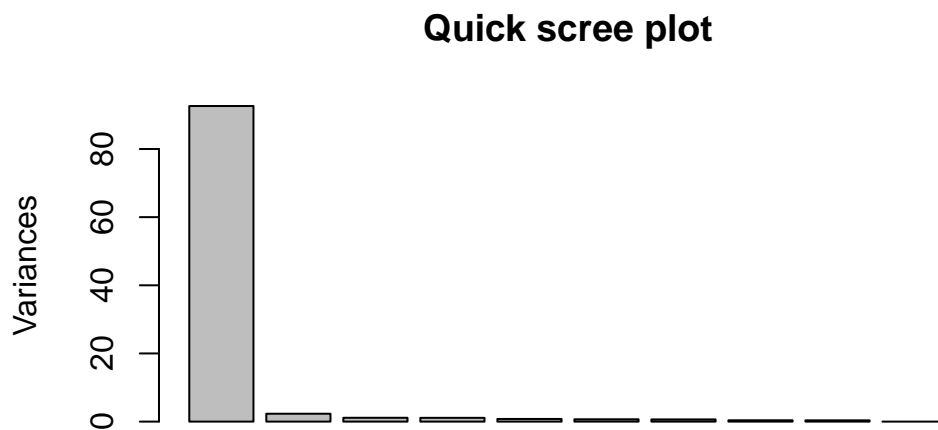
```
summary(rpca)
```

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	3.457e-15
Proportion of Variance	0.00385	0.00364	0.000e+00
Cumulative Proportion	0.99636	1.00000	1.000e+00

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



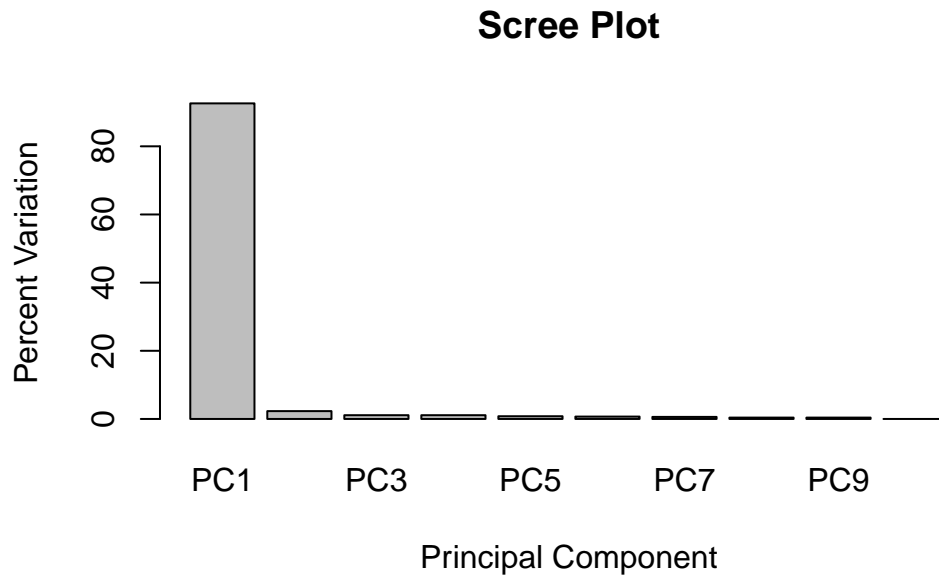
Calculating the contribution to variance in the original data

```
rpca.var <- rpca$sdev^2  
  
rpca.var.per <- round(rpca.var/sum(rpca.var)*100, 1)  
rpca.var.per
```

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



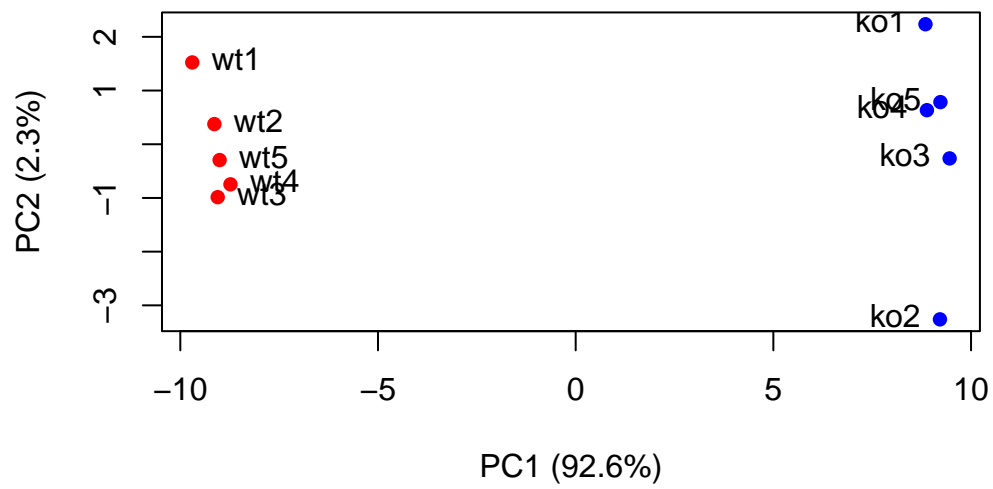
```
barplot(rpca.var.per, main="Scree Plot",
        names.arg = paste0("PC", 1:10),
        xlab="Principal Component", ylab="Percent Variation")
```



```
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

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

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

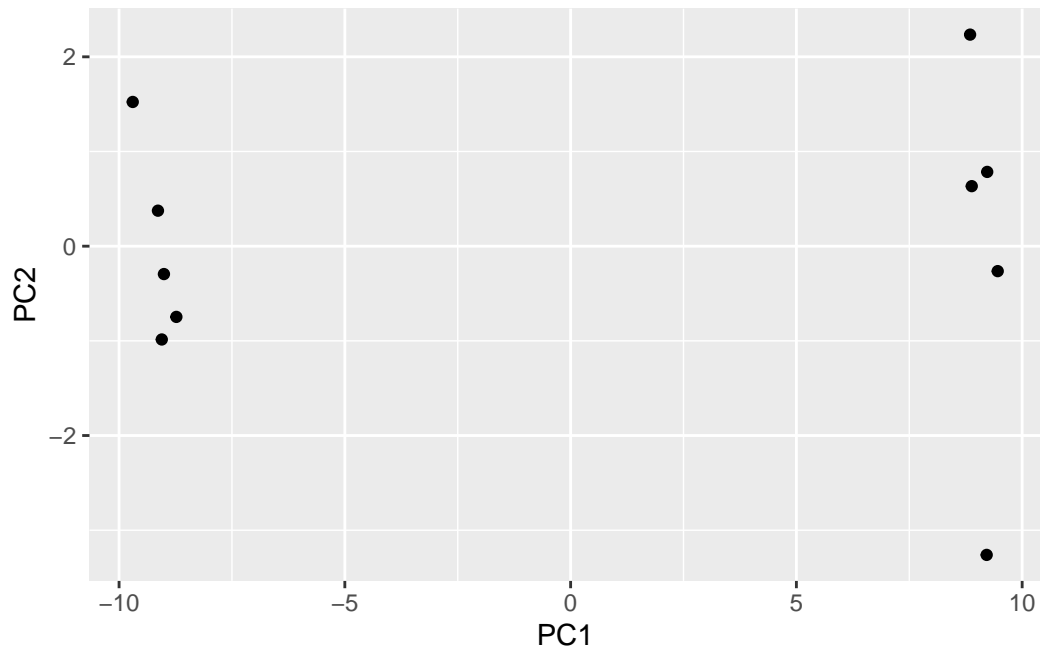


Using ggplot to make these graphs

```
library(ggplot2)

rdf <- as.data.frame(rpca$x)

ggplot(rdf) +
  aes(PC1, PC2) +
  geom_point()
```



```
rdf$samples <- colnames(rna.data)
rdf$condition <- substr(colnames(rna.data),1,2)

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

