```
knitr::opts_chunk$set(echo = TRUE)
load("lattice.RData")
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.2
```

```
## Warning in as.POSIXlt.POSIXct(Sys.time()): unable to identify current timezone 'H':
## please set environment variable 'TZ'
```

```
library(devtools)
```

```
## Warning: package 'devtools' was built under R version 4.1.2
```

```
## Loading required package: usethis
```

```
## Warning: package 'usethis' was built under R version 4.1.2
```

```
library(ade4)
```

```
## Warning: package 'ade4' was built under R version 4.1.2
```

```
library(ggbiplot)
```

```
## Loading required package: plyr
```

```
## Warning: package 'plyr' was built under R version 4.1.2
```

```
##
## Attaching package: 'plyr'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     ozone
```

```
## Loading required package: scales
```

```
## Warning: package 'scales' was built under R version 4.1.2
```

```
## Loading required package: grid
```

```
library(viridis)
```

```
## Warning: package 'viridis' was built under R version 4.1.2
```

```
## Loading required package: viridisLite
```

```
## Warning: package 'viridisLite' was built under R version 4.1.2
```

```
##
## Attaching package: 'viridis'

## The following object is masked from 'package:scales':
##
##     viridis_pal
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.2

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1
## v purrr   0.3.4

## Warning: package 'tibble' was built under R version 4.1.2

## Warning: package 'tidyr' was built under R version 4.1.2

## Warning: package 'readr' was built under R version 4.1.2

## Warning: package 'purrr' was built under R version 4.1.2

## Warning: package 'dplyr' was built under R version 4.1.2

## Warning: package 'stringr' was built under R version 4.1.2

## Warning: package 'forcats' was built under R version 4.1.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::arrange()    masks plyr::arrange()
## x readr::col_factor() masks scales::col_factor()
## x purrr::compact()    masks plyr::compact()
## x dplyr::count()      masks plyr::count()
## x purrr::discard()    masks scales::discard()
## x dplyr::failwith()   masks plyr::failwith()
## x dplyr::filter()     masks stats::filter()
## x dplyr::id()         masks plyr::id()
## x dplyr::lag()        masks stats::lag()
## x dplyr::mutate()     masks plyr::mutate()
## x dplyr::rename()     masks plyr::rename()
## x dplyr::summarise()  masks plyr::summarise()
## x dplyr::summarize()  masks plyr::summarize()
```

```
library(dplyr)
library(ggrepel)
```

```
## Warning: package 'ggrepel' was built under R version 4.1.2
```

```r
head(hamster,5)
```

```
##       lung    heart    liver     spleen   kidney    testes
## 1 0.969500 1.352002 8.185496 0.22550020 1.348003 0.3415002
## 2 1.141003 1.499403 8.544507 0.27049980 1.322103 0.4170000
## 3 0.872000 1.268503 6.823009 0.04999999 0.897001 1.2200020
## 4 0.739900 1.270203 9.067509 0.11299980 1.278003 0.8145000
## 5 1.139302 1.496203 9.621993 0.11980000 1.415003 1.1568030
```

```r
ham_noscale = prcomp(hamster, scale. = FALSE)
```

```r
ham_noscale$rotation[,1:2]
```

```
##                PC1           PC2
## lung   -0.04844187 -1.400244e-02
## heart  -0.07583873 -1.194474e-01
## liver  -0.99409891 -2.013637e-02
## spleen -0.01213091 -1.658741e-05
## kidney -0.04990582 -4.195610e-02
## testes -0.03211681  9.916504e-01
```

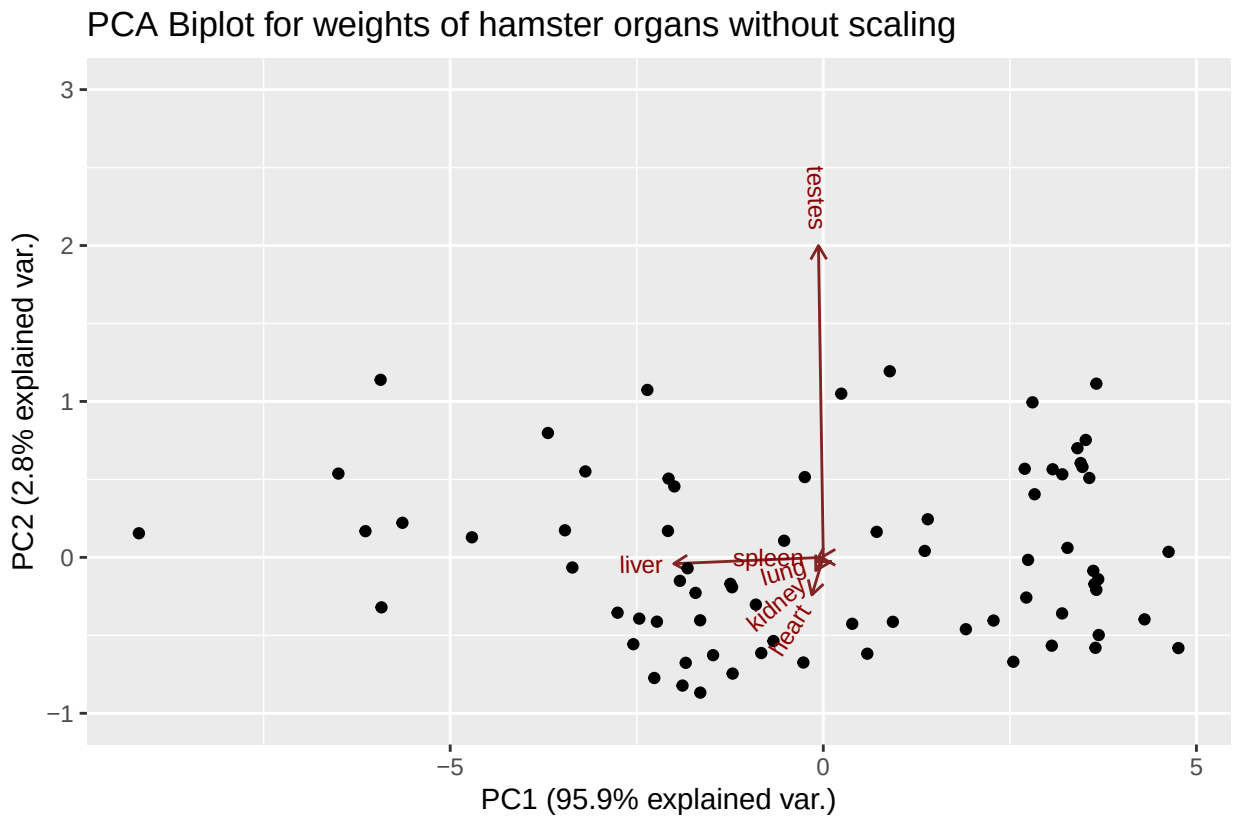
```r
ham_noscale2 = prcomp(hamster, scale. = TRUE)
ham_noscale2$rotation[,1:2]
```

```
##                PC1         PC2
## lung   -0.40885549 -0.05704641
## heart  -0.46129490 -0.20800130
## liver  -0.48564405  0.08125510
## spleen -0.41489143  0.09751789
## kidney -0.45260553 -0.09380103
## testes -0.08489638  0.96362358
```

```r
summary(hamster)
```

```
##       lung            heart            liver            spleen
##  Min.   :0.3216   Min.   :0.4134   Min.   : 1.626   Min.   :0.0293
##  1st Qu.:0.6220   1st Qu.:0.8039   1st Qu.: 3.262   1st Qu.:0.0657
##  Median :0.8416   Median :1.0690   Median : 6.439   Median :0.0896
##  Mean   :0.8457   Mean   :1.0581   Mean   : 6.299   Mean   :0.1035
##  3rd Qu.:1.0260   3rd Qu.:1.2763   3rd Qu.: 8.307   3rd Qu.:0.1283
##  Max.   :1.6521   Max.   :1.8390   Max.   :15.492   Max.   :0.2959
##      kidney           testes
##  Min.   :0.8199   Min.   :0.3082
##  1st Qu.:1.0751   1st Qu.:0.6387
##  Median :1.2291   Median :0.9761
##  Mean   :1.2256   Mean   :1.0900
##  3rd Qu.:1.3642   3rd Qu.:1.4862
##  Max.   :1.7381   Max.   :2.4244
```

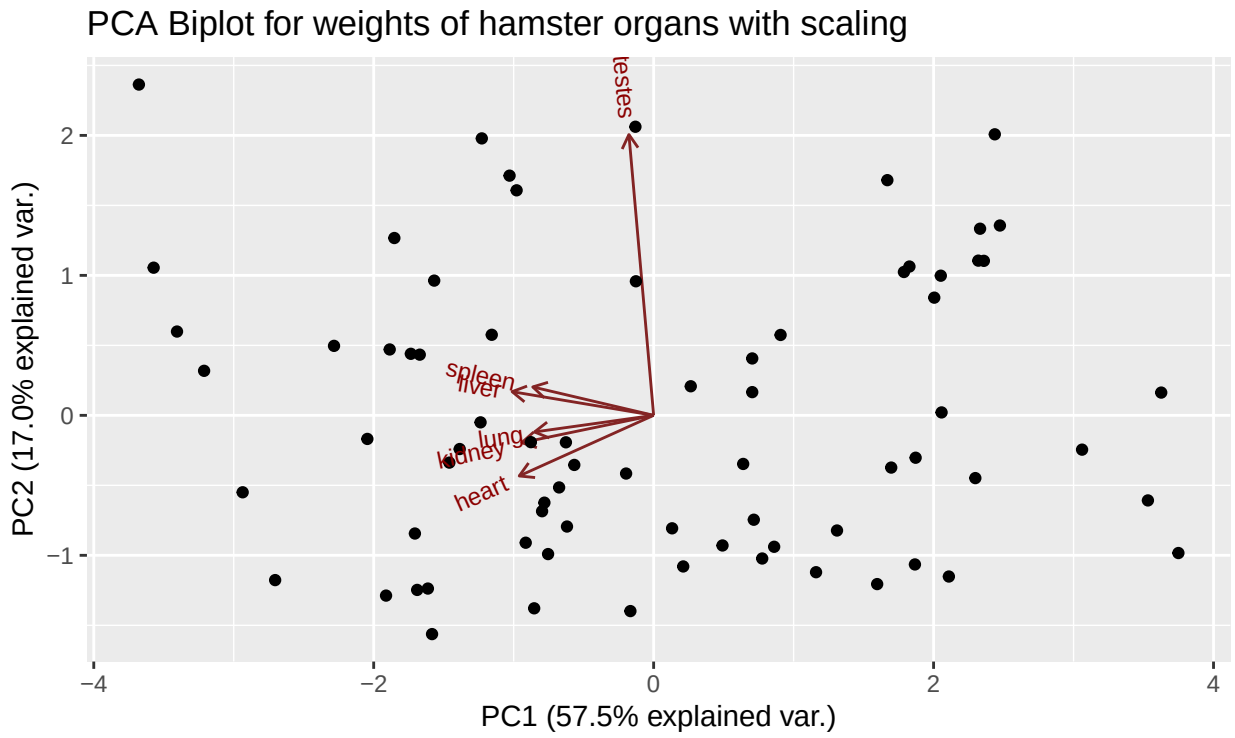
```
## scale = 0 means form biplot
ggbiplot(ham_noscale, scale = 0) + ggtitle("PCA Biplot for weights of hamster organs without scaling") +
```



Q.1)

1. For the above plot, we applied PCA on the hamster dataset without scaling the data. We see that all variables are on different scales.
2. The first principal component i.e. PC1 shows an explained variance of almost 96%.
3. However we see that Testes shows the most variation followed by liver.
4. Since we are not scaling the features, we won't be able to take the variation of other features like kidney, heart, spleen and lung.
5. This results in capturing most of the variation from Testes and Liver. We can solve this problem if we scale all features.

```
## scale = 0 means form biplot
ggbiplot(ham_noscale2, scale = 0) + ggtitle("PCA Biplot for weights of hamster organs with scaling")
```



6. In the above PCA biplot we applied PCA on the scaled data for all organ weights. 7. Compared to the earlier plot wherein we were capturing variation from only two features, here we can capture variation of all the features using two principal components 8. Thus we are able to capture variance from all features or dimensions by projecting it on principal components.

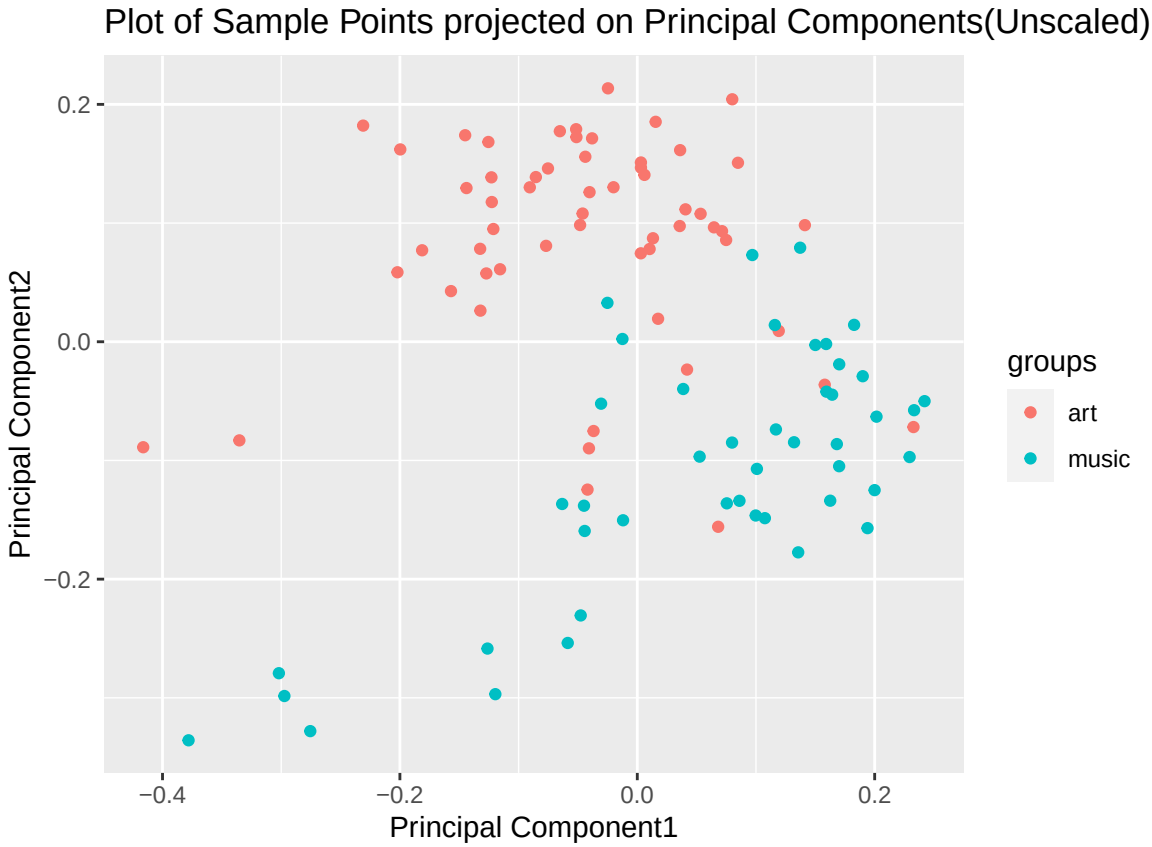
Q.2)

```
nyt_pca = read.csv("nyt_articles.csv")
```

```
nyt_pca1 = nyt_pca[,!sapply(names(nyt_pca), function(col) {all(nyt_pca[,col]==0) |
                                        !is.numeric(nyt_pca[,col])})]
```

```
nyt_noscale = prcomp(nyt_pca1, scale. = FALSE)
```

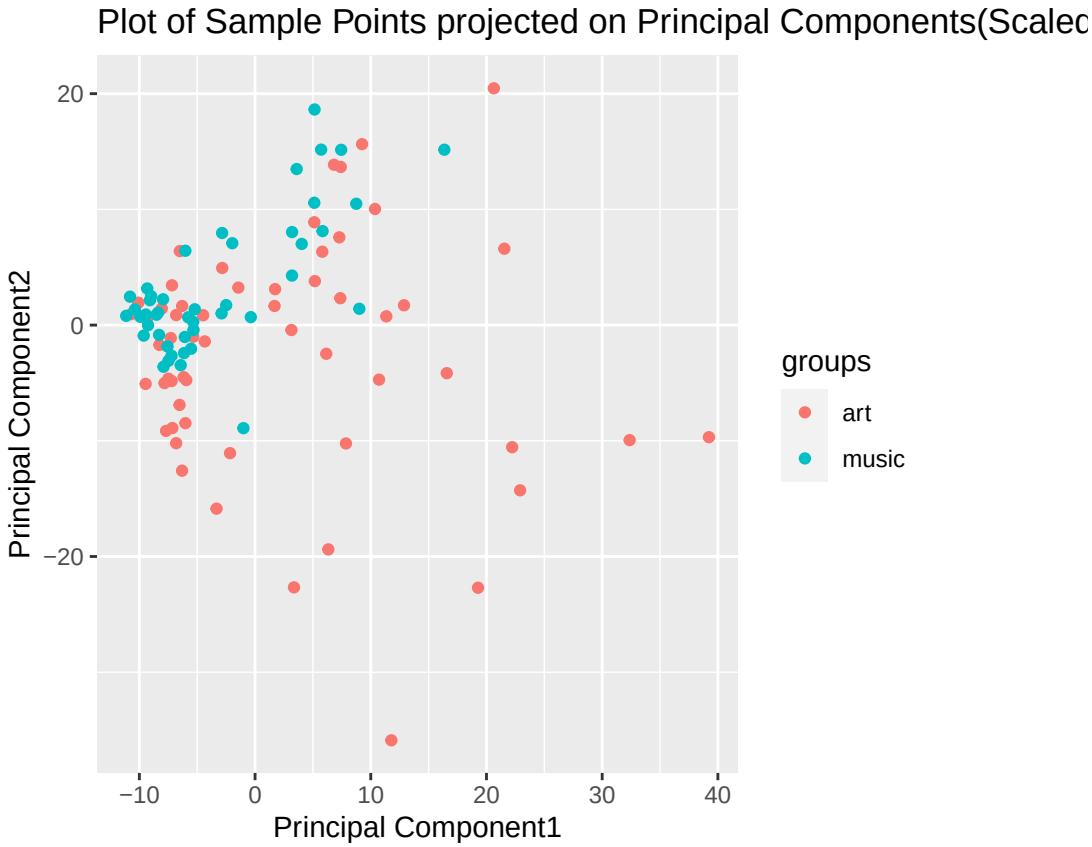
```
ggbiplot(nyt_noscale, scale = 0, var.axes = FALSE, groups = nyt_pca$class.labels) + ggtitle("Plot of Sa
  xlab("Principal Component1") + ylab("Principal Component2")
```



1. In the above PCA plot for sample points, we did not apply scaling on the features. 2. We see that we are able to distinguish art and music articles because of the variation of the features from the data. that is captured by our two main principal components 3. We have differentiated the art and music articles on the basis of seperate colors to represent these clusters.

```
nyt_noscale2 = prcomp(nyt_pca1 , scale. = TRUE)
```

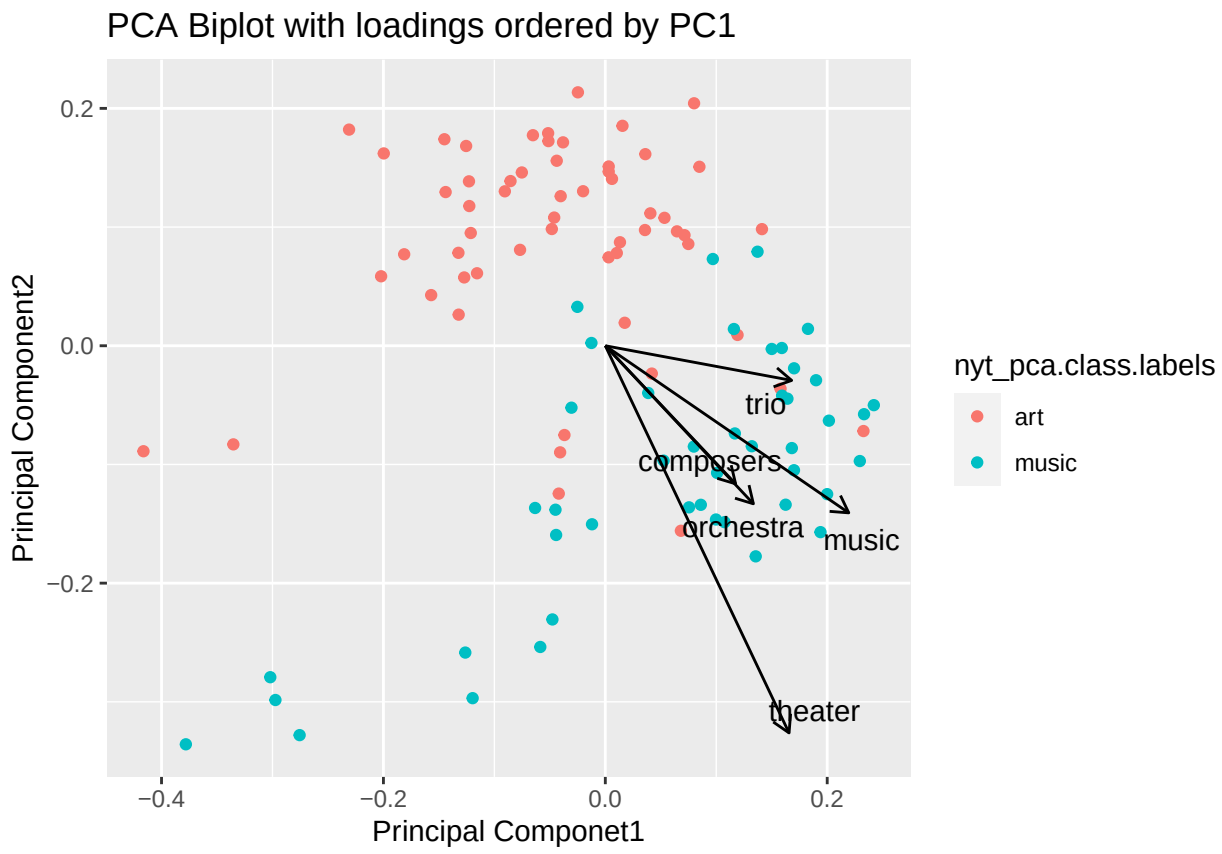
```
ggbiplot(nyt_noscale2, scale = 0, var.axes = FALSE, groups = nyt_pca$class.labels)+ ggtitle("Plot of Sa
  xlab("Principal Component1") + ylab("Principal Component2")
```



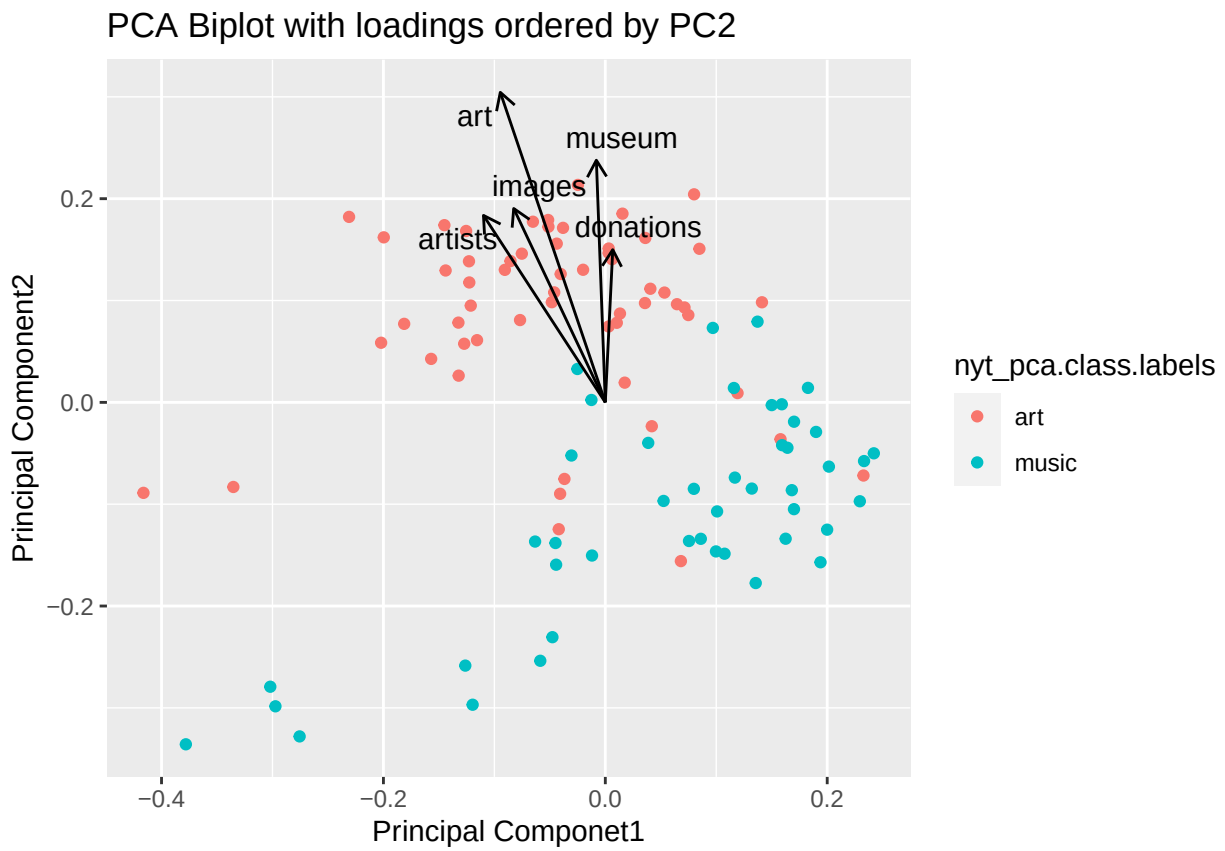
4. In the above PCA plot for sample points, we applied scaling of the features.
5. Here we see that the clusters for art and music articles are focussed towards same location and as a result we are not able to differentiate between the two article classes.
6. Since the data is already normalized there is no need to apply normalization the second time to the data
7. Thus the scale is same for all features and therefore using PCA on unscaled features is more beneficial.
8. We can say that using our main two principal components, we are able to capture variation in features with respect to art and music articles.

Q.3)

```
rot <-data.frame(nyt_noscale$rotation[,1:2])
rot_0<-rot[order(-rot$PC1),][1:5,]
rot_1<-rot[order(-rot$PC2),][1:5,]
nyt_x <- data.frame(nyt_noscale$x,nyt_pca$class.labels)
ggplot() +
    geom_point(aes(x = PC1, y = PC2,color=nyt_pca.class.labels), data = nyt_x) +
    geom_segment(aes(xend = PC1*2, yend = PC2*2, x = 0, y = 0), data = rot_0,
                 arrow = arrow(length = unit(0.1, "inches"))) +
    geom_text_repel(aes(x = PC1*2, y = PC2*2, label = row.names(rot_0)), data = rot_0)+
    labs(x = "Principal Componet1", y = "Principal Component2",
         title = "PCA Biplot with loadings ordered by PC1")
```




```
ggplot() +
    geom_point(aes(x = PC1, y = PC2,color=nyt_pca.class.labels), data = nyt_x) +
    geom_segment(aes(xend = PC1*2, yend = PC2*2, x = 0, y = 0), data = rot_1,
                 arrow = arrow(length = unit(0.1, "inches"))) +
    geom_text_repel(aes(x = PC1*2, y = PC2*2, label = row.names(rot_1)), data = rot_1) +
    labs(x = "Principal Componet1", y = "Principal Component2",
         title = "PCA Biplot with loadings ordered by PC2")
```



We created two biplots. One biplot was used for plotting largest loadings according to principal component1 i.e PC1 and other biplot was used for plotting largest loadings according to principal component2 i.e PC2

1. From the plot of largest loadings with respect to PC1, we see that variation of features that are most commonly found in music articles is captured by PC1.
2. From the plot of largest loadings with respect to PC2, we see that variation of features that are most commonly found in art articles is captured by PC2.
3. From this, we can conclude that PC1 and PC2 capture variation from only a particular type of article class at a time. So music and art articles must be having some different words.