Homework-3.R

oyino

2020-09-16

```r
#install.packages("mlbench")
library(mlbench)
#install.packages("Rtsne")
library("Rtsne")
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
#install.packages("caret")
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
#install.packages("MASS")
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
#install.packages("magrittr")
#library(magrittr)
#install.packages("devtools")
#library(devtools)
#install.packages("ggbiplot")
#library(ggbiplot)
#install_github("vqv/ggbiplot")
library("Seurat")
```

```
## Registered S3 method overwritten by 'spatstat':
##   method     from
##   print.boxx cli
```

```r
library("FactoMineR")
#install.packages("factoextra")
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
library(MASS)
library(umap)


data("Glass")
str(Glass)
```

```
## 'data.frame':    214 obs. of  10 variables:
##  $ RI  : num  1.52 1.52 1.52 1.52 1.52 ...
##  $ Na  : num  13.6 13.9 13.5 13.2 13.3 ...
##  $ Mg  : num  4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...
##  $ Al  : num  1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...
##  $ Si  : num  71.8 72.7 73 72.6 73.1 ...
##  $ K   : num  0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...
##  $ Ca  : num  8.75 7.83 7.78 8.22 8.07 8.07 8.17 8.24 8.3 8.4 ...
##  $ Ba  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Fe  : num  0 0 0 0 0 0.26 0 0 0 0.11 ...
##  $ Type: Factor w/ 6 levels "1","2","3","5",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
#Glass = matrix(Glass)

Glass [duplicated(Glass),]
```

```
##         RI    Na   Mg   Al    Si    K   Ca Ba Fe Type
## 40 1.52213 14.21 3.82 0.47 71.77 0.11 9.57  0  0    1
```

```r
#to find the duplicate row
Glass_Unique = Glass [!duplicated(Glass),]
#remove the duplicate row
Glass_matrix = as.matrix(Glass_Unique[,-10])
#store the new dataframe as a matrix
```

```r
corMat = cor(Glass_matrix)
#the correlation matrix for the predictor variables
#in the Glass data set.
#this shows the collinearity in the data and to see highly colinear data.
#corMat

#aii)
eigen(corMat)
```

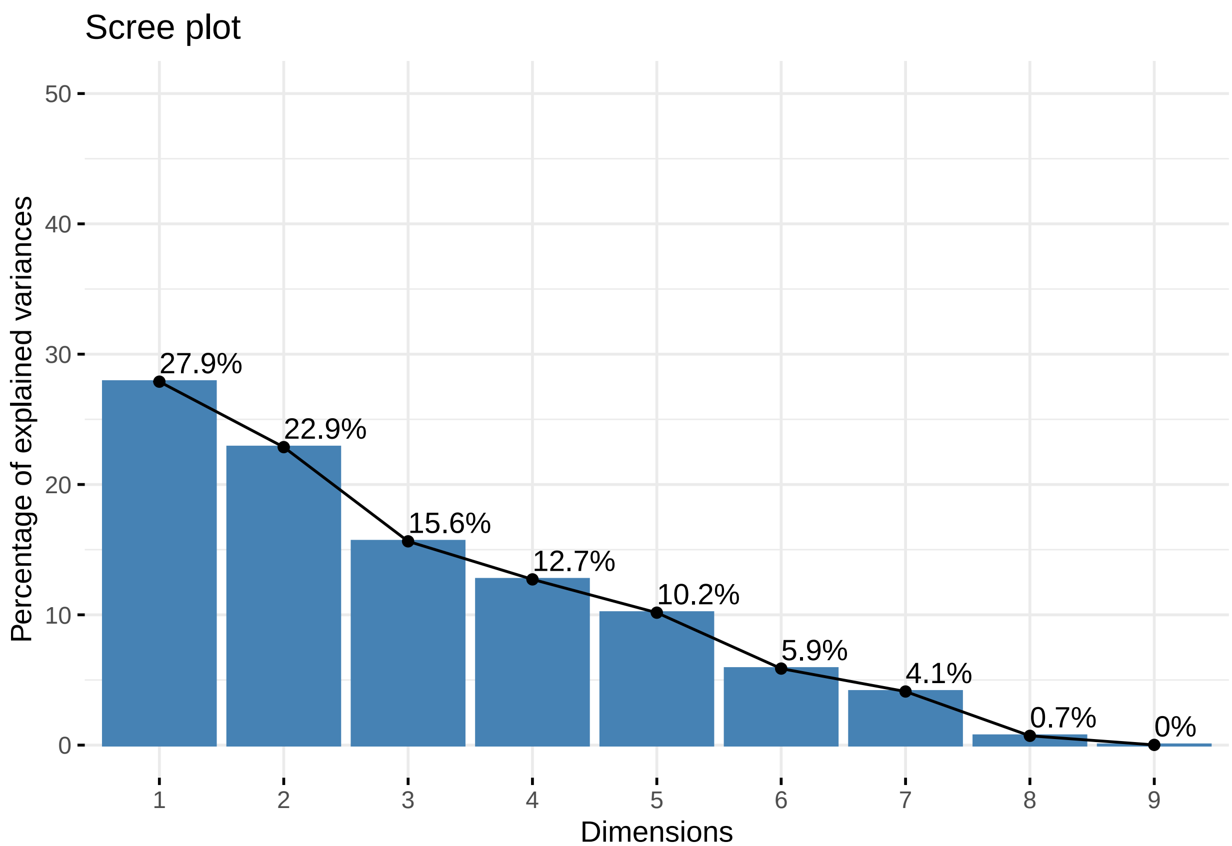
```
## eigen() decomposition
## $values
## [1] 2.510152168 2.058169337 1.407484057 1.144693344 0.914768873 0.528593040
## [7] 0.370262639 0.064267543 0.001608997
##
## $vectors
##             [,1]        [,2]        [,3]       [,4]        [,5]        [,6]
##  [1,]  0.5432231 -0.28911804 -0.08849541  0.1479796  0.07670808 -0.11455615
##  [2,] -0.2676141 -0.26909913  0.36710090  0.5010669 -0.14626769  0.55790564
##  [3,]  0.1093261  0.59215502 -0.02295318  0.3842440 -0.11610001 -0.30585293
##  [4,] -0.4269512 -0.29636272 -0.32602906 -0.1488756 -0.01720068  0.02014091
##  [5,] -0.2239232  0.15874450  0.47979931 -0.6394962 -0.01763694 -0.08850787
##  [6,] -0.2156587  0.15305116 -0.66349177 -0.0733491  0.30154622  0.24107648
##  [7,]  0.4924367 -0.34678973  0.01380151 -0.2743430  0.18431431  0.14957911
##  [8,] -0.2516459 -0.48262056 -0.07649040  0.1299431 -0.24970936 -0.65986429
##  [9,]  0.1912640  0.06089167 -0.27223834 -0.2252596 -0.87828176  0.24066617
##              [,7]        [,8]        [,9]
##  [1,] -0.08223530  0.75177166 -0.02568051
##  [2,] -0.15419352  0.12819398  0.31188932
##  [3,]  0.20691746  0.07799332  0.57732740
##  [4,]  0.69982052  0.27334224  0.19041178
##  [5,] -0.20945417  0.38077660  0.29747147
##  [6,] -0.50515516  0.11064442  0.26075531
##  [7,]  0.09984144 -0.39885229  0.57999243
##  [8,] -0.35043794 -0.14497643  0.19853265
##  [9,] -0.07120579  0.01650505  0.01459278
```

```r
#the eigen valuesand values for the glass data set. SHown below
#aiii)
pca = prcomp(Glass_matrix, scale= T)
#the pca would project the dimensions of the dataset into smaller dimension.
#this preserves the majority of the inofrmation
#from the data and also eases visualization of data relationship

#aiv)
#The eigenvalues of the correlation matrix
#are the squares of the standard deviation i.e variances
#of the principal components themselves are same as eigenvectors
#of the correlation matrix. Even though the signs
#maybe opposite as is the case here.

#av)
#pca
pca_columns = pca$rotation
```

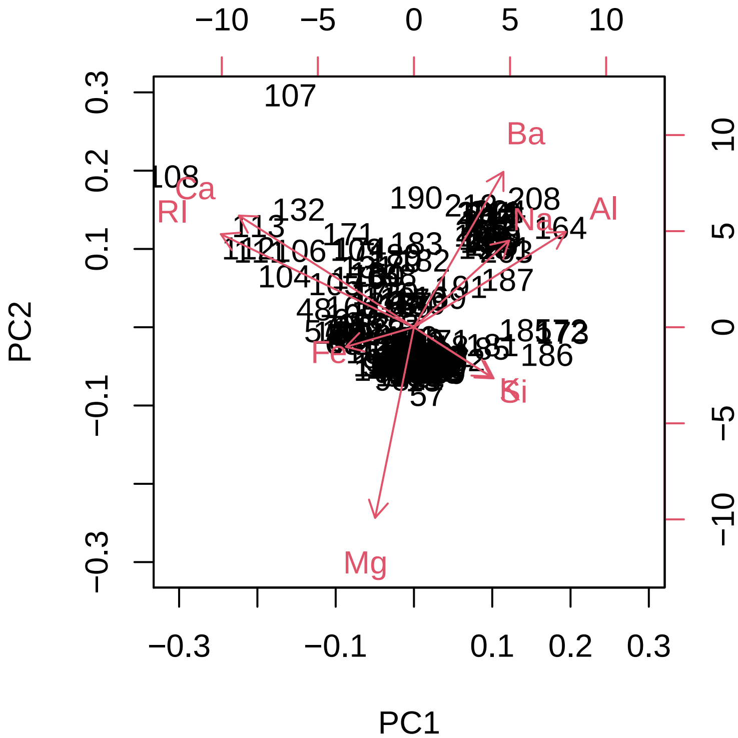
```r
PC1 = pca_columns[,1]
PC2 = pca_columns[,2]
PC1%*%PC2
```

```
##                 [,1]
## [1,] -3.243933e-16
```

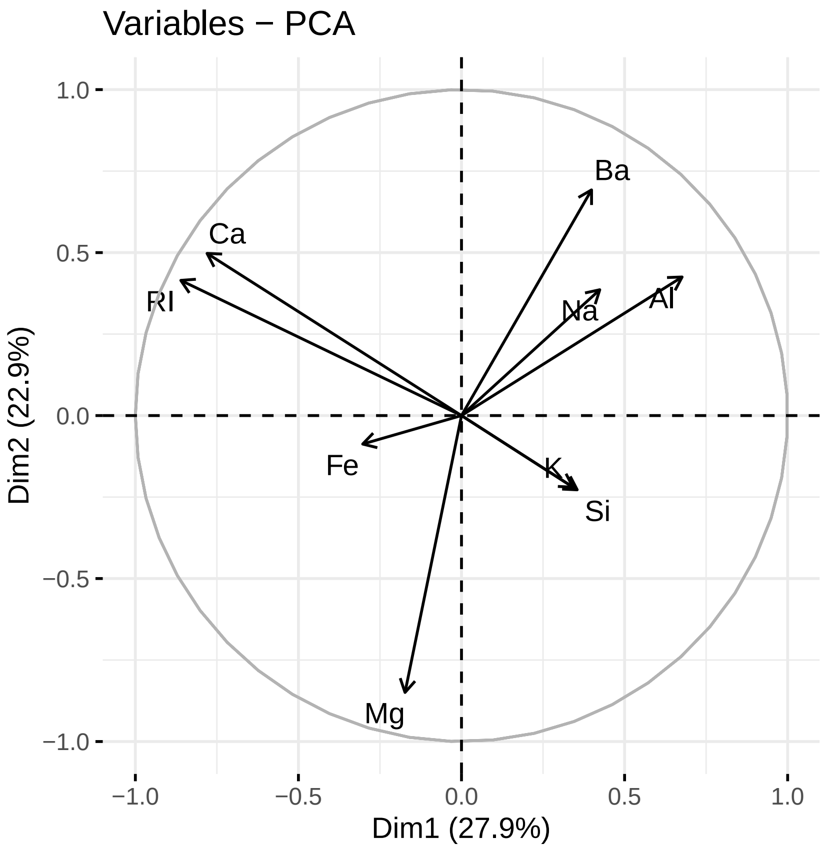
```r
# two vectors are orthogonal when the inner product is 0. The inner products of
#PC1 and PC2 was approximately zero whcih means they are orthogonal

#1b
fviz_eig(pca, addlabels = TRUE, ylim = c(0, 50))
```



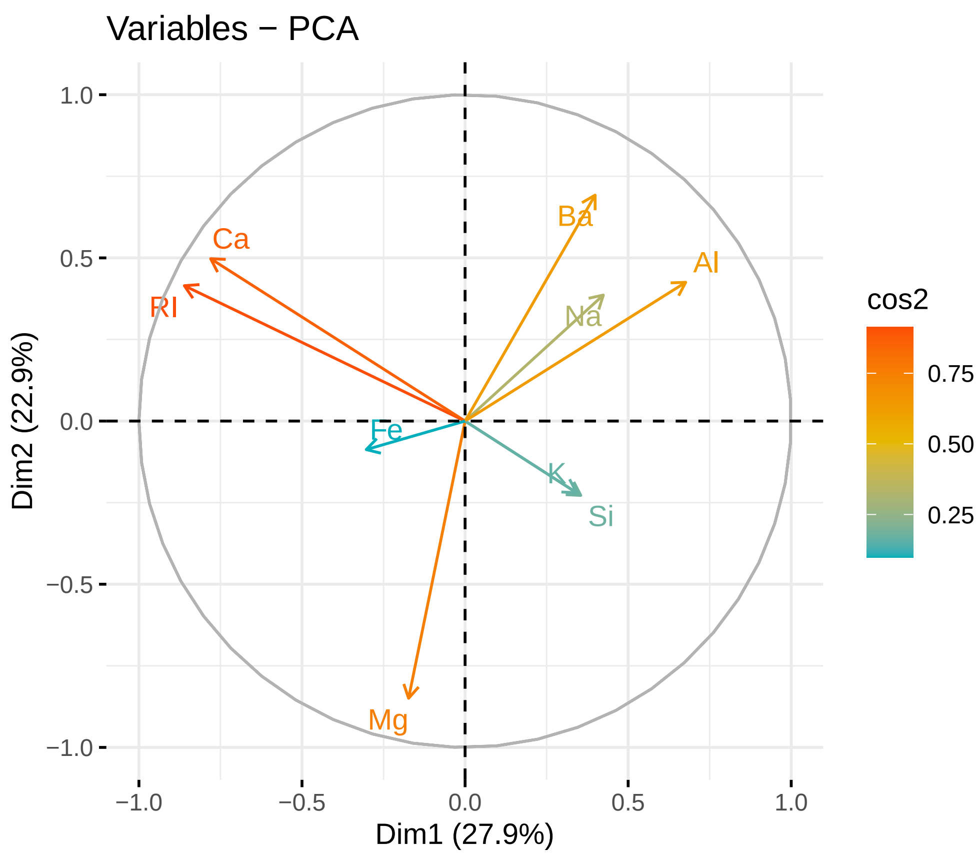

```r
#The diagram shows the number of dimensions that the data set was reduced to
#and the variance explained by each of the dimensions.
#The first dimension explains the most varinace (27.9%). The first 5 dimensions
#explains 89.3% variance of the original data.

biplot(pca)
```



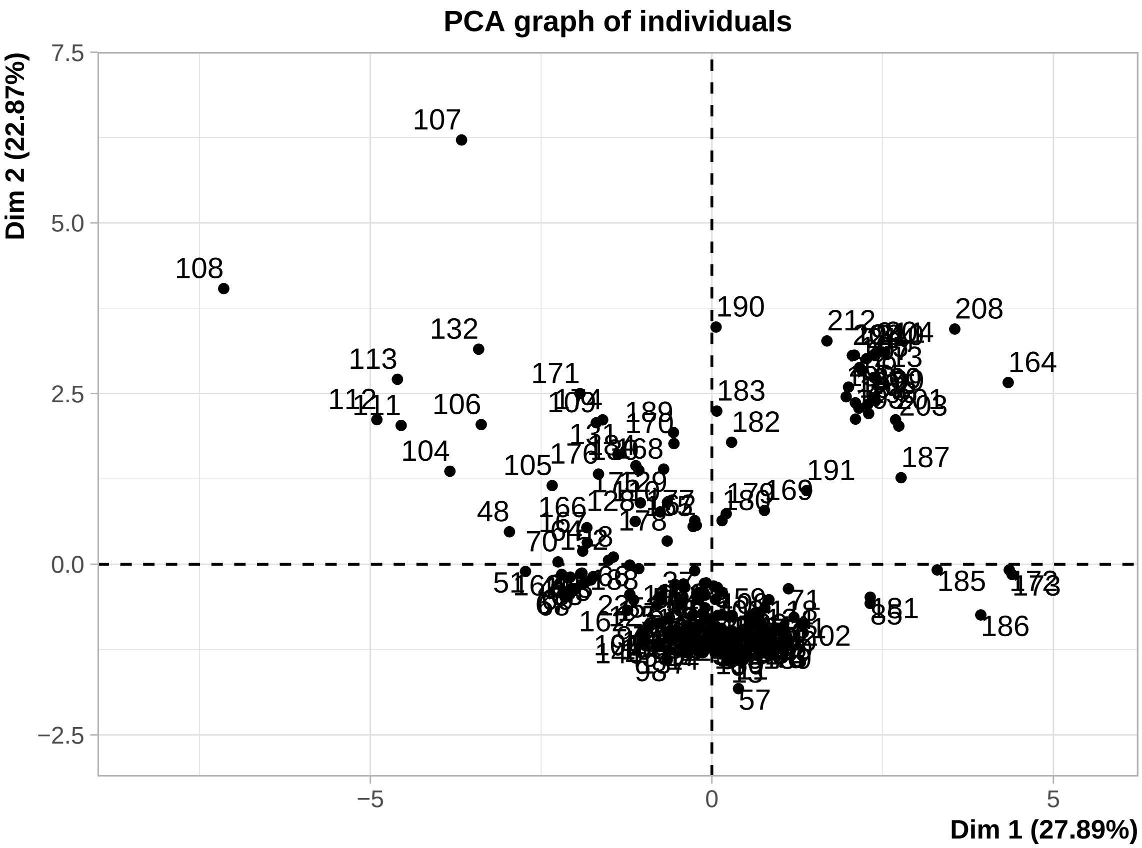

```
#this does not show a good represenatatio of the data due to the large number of
#individual parameters. Color would also make the plot look better.

fviz_pca_var(pca, col.var = "black", repel = TRUE)
```

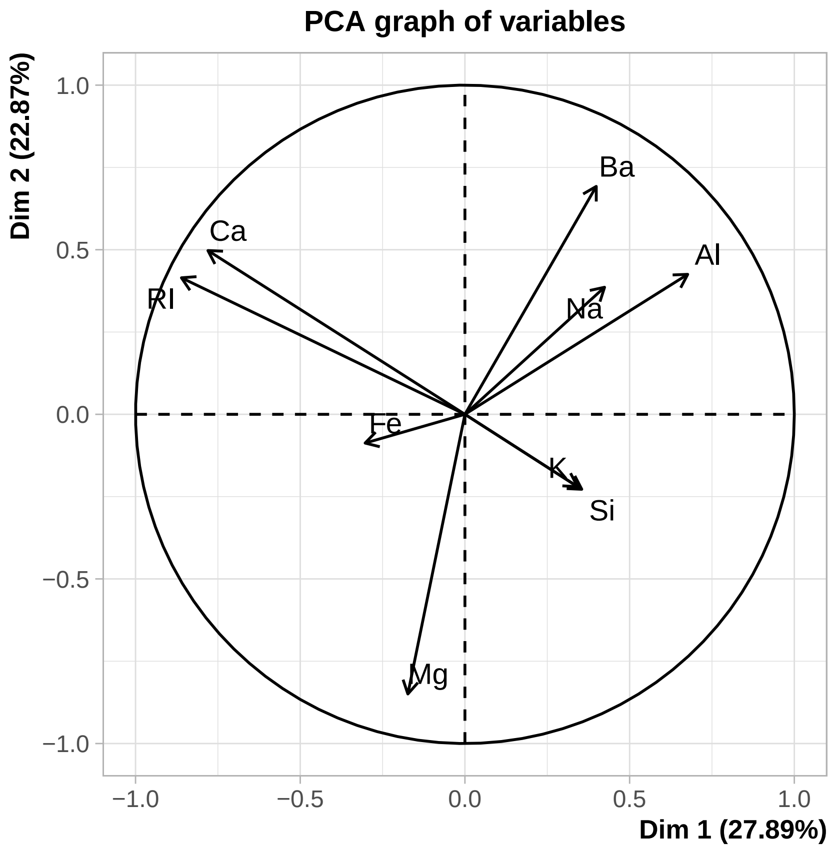



```
#variable correlation plot.The distance of the varibale from the center of the circle
#represent the quality of the variables on the factor map.
#Variables that are far away are well represented i.e with longer arrow length.
#i.e the the longer the arrow lines, the more influence
#the variable has on the principal component.

fviz_pca_var(pca, col.var = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE # Avoid text overlapping
)
```




```r
#The cos2 values are used to estimate the quality of the representation
#The closer a variable is to the circle of correlations, the better its
#representation on the factor map
#(and the more important it is to interpret these components)
#Variables that are closed to the center of the plot are
#less important for the first components.
#variables with high correlations are colored red, blue for the low effects
#and orange for average

#pca
#the shows all the pCs sorted accoring to the importance
#and weight of each of the variables.


glass.pca <- PCA(Glass_matrix, scale.unit = TRUE)
```


PCA graph of individuals
Dim 2 (22.87%)
Dim 1 (27.89%)
7.5
5.0
2.5
0.0
−2.5
−5
0
5
107
108
132
113
112
111
106
104
171
109
189
170
183
182
190
212
208
164
187
191
169
105
48
185
172
173
186
181
51
68
102
57




```r
glass.desc <- dimdesc(glass.pca)
# Description of dimension 1
glass.desc$Dim.1
```

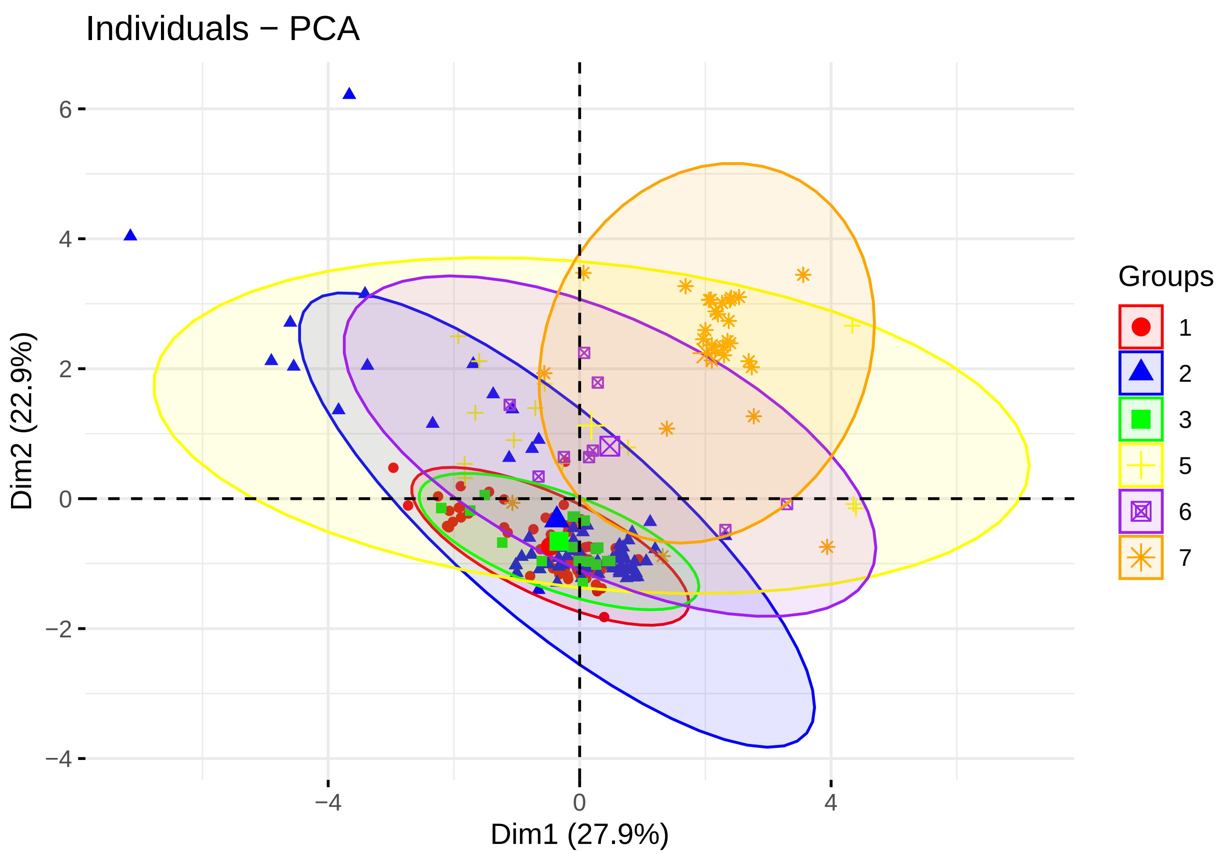
```
## $quanti
##    correlation      p.value
## Al   0.6764384 7.586117e-30
## Na   0.4239934 1.055021e-10
## Ba   0.3986941 1.573157e-09
## Si   0.3547719 1.029914e-07
## K    0.3416780 3.188243e-07
## Mg  -0.1732104 1.133441e-02
## Fe  -0.3030284 6.704662e-06
## Ca  -0.7801902 7.177285e-45
## RI  -0.8606534 9.028843e-64
##
## attr(,"class")
## [1] "condes" "list "
```

```r
#the first PC represents 27.89% of the variance of the data.
#the table shows the % of each of the variable
#explained by PC1. tHE RI variable has the largest contribution to PC1
#from the biplot plot, the variable with high
#contribution increase or decrease across the horizontal section.
```

```r
glass.desc$Dim.2
```

```
## $quanti
##    correlation      p.value
## Ba   0.6923830 9.737651e-32
## Ca   0.4975157 1.015288e-14
## Al   0.4251715 9.251141e-11
## RI   0.4147780 2.897971e-10
## Na   0.3860583 5.585204e-09
## K   -0.2195721 1.259104e-03
## Si  -0.2277400 8.130739e-04
## Mg  -0.8495246 1.609808e-60
##
## attr(,"class")
## [1] "condes" "list "
```

```r
#the second PC represents 22.87% of the variance of the data.
#the table shows the % of each of the variable explained by PC1.
#tHE Mg variable has the largest contribution to PC1
#from the bilpot plot,  the variable with high
#contribution increase or decrease across the vertical section.



#Glass_Unique$Type
type = as.factor(Glass_Unique$Type)

fviz_pca_ind(glass.pca,
             geom.ind = "point", # show points only (nbut not "text")
             col.ind = Glass_Unique$Type, # color by groups
             palette = c("red", "blue", "green", "yellow", "purple", "orange"),
             addEllipses = TRUE, # Concentration ellipses
             legend.title = "Groups"
)
```




```
#this plot shows the types of glass and the groups of each of the glass type.
#The red represents the type 1, blue color represents Type 2, green color
#represents Type 3, Yellow color represents Type 5, Purple color represents 6,
#Orange color represents Type 7 glass.

#The glass type and cluster overlap on the PC1 and PC2 which shows that the
#glass type share similar properties.


rbind(
  SD = sqrt(pca$sd^2),
  per_var = (pca$sd^2/sum(pca$sd^2))*100,
  per_var_cum = (cumsum(pca$sd^2)/sum(pca$sd^2))*100)
```

```
##                  [,1]      [,2]      [,3]      [,4]       [,5]      [,6]
## SD           1.584346  1.434632  1.186374  1.069903  0.9564355  0.727044
## per_var     27.890580 22.868548 15.638712 12.718815 10.1640986  5.873256
## per_var_cum 27.890580 50.759128 66.397840 79.116655 89.2807531 95.154009
##                   [,7]       [,8]         [,9]
## SD           0.6084921  0.2535104   0.04011231
## per_var      4.1140293  0.7140838   0.01787775
## per_var_cum 99.2680384 99.9821223 100.00000000
```

```
#the dimension can be reduced to 5 dimensions
#this tables shows the standard deviation represented by
```

```
#each of the PC and the standard deviation
#decreases as the PC number increases.
#This is also represented in the percentage variance explained by each of the
#principal comp. 9 PCs explain 100 variation is the data
#but 5 PCs represents 89.3% which is very good representation.

#biii
#yes, The dimension can be reduced to 5 PCs and
#the still preserve 89.3% of the data.

#1c
preproc.param = Glass %>% preProcess(method = c("center", "scale"))
#preprocess does mean centering and scaling and
#it is not affected by factor level. It ignores them

#transform the data using the estimated parameters
transformed  = preproc.param %>% predict(Glass)

#transformed
#fit the model

lda.model  = lda(Type ~., data = transformed)
#lda.model
#The purpose of the linear discriminant analysis is to find
#combination of the variables that give
#best possible separation between groups (glass Type) in our data set.
#group probability means the initial data proportion



Coefficients_of_linear_discriminants = lda.model$scaling
#the coefficient that shows the proportion of each of the
#variables represented by the LDA, to get the .
Coefficients_of_linear_discriminants
```

```
##             LD1        LD2       LD3          LD4        LD5
## RI  0.94656386 0.08925658 1.0811807  0.749671704 -2.4436288
## Na  1.94450927 2.58461557 0.3753751  5.654449980  1.9588285
## Mg  1.06793259 4.30684515 2.2687400  9.880261458  4.0391676
## Al  1.66643308 0.86111013 1.0996248  3.204929873  0.4678828
## Si  1.89891675 2.32855629 1.3187565  5.841781630  0.7406973
## K   1.02491652 1.21439159 0.8387922  5.267175041  1.8398285
## Ca  1.43213378 3.37701884 0.9215204  9.530340500  5.2814448
## Ba  1.15061274 1.71202472 1.2910288  3.201342577  2.1915962
## Fe -0.04983573 0.02110900 0.1171805 -0.004360319 -0.1269549
```

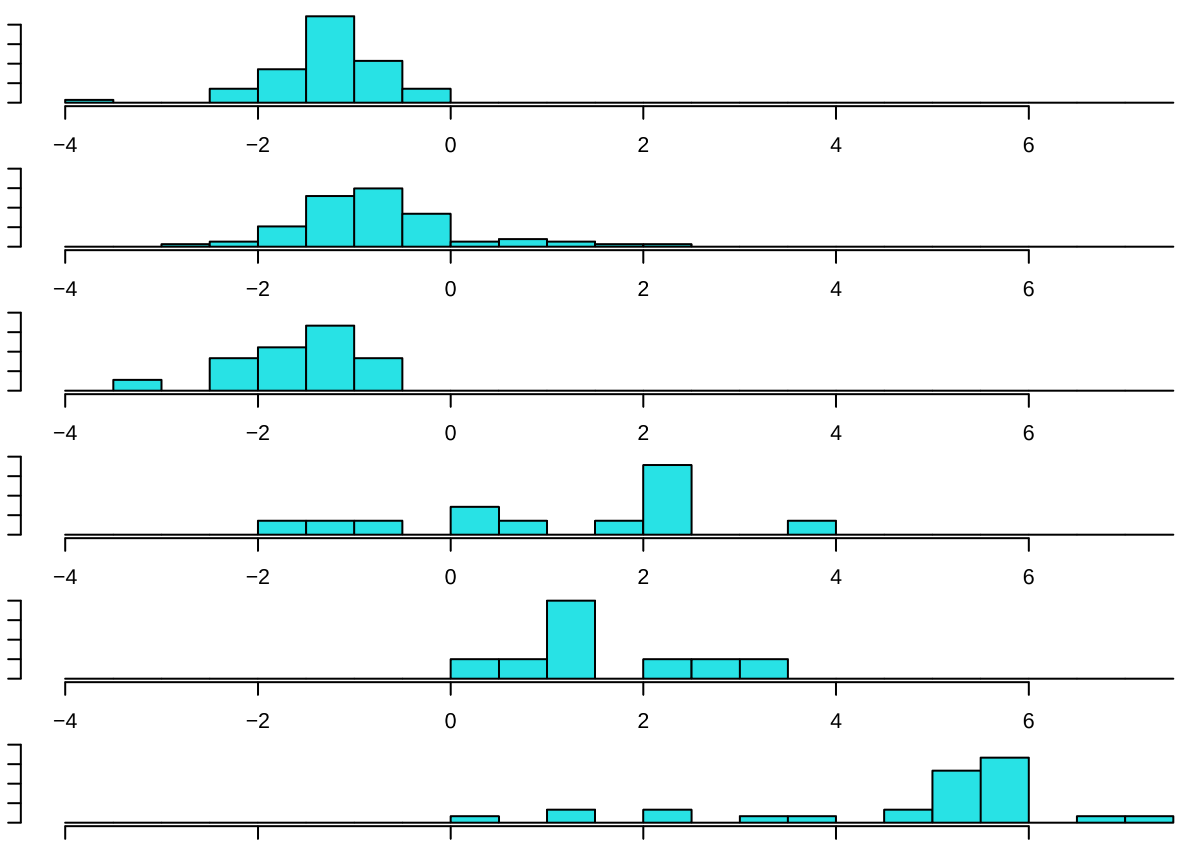
```
one_lda = Coefficients_of_linear_discriminants[,1]
one_lda
```

```
##          RI          Na          Mg          Al          Si           K
##  0.94656386  1.94450927  1.06793259  1.66643308  1.89891675  1.02491652
##          Ca          Ba          Fe
##  1.43213378  1.15061274 -0.04983573
```

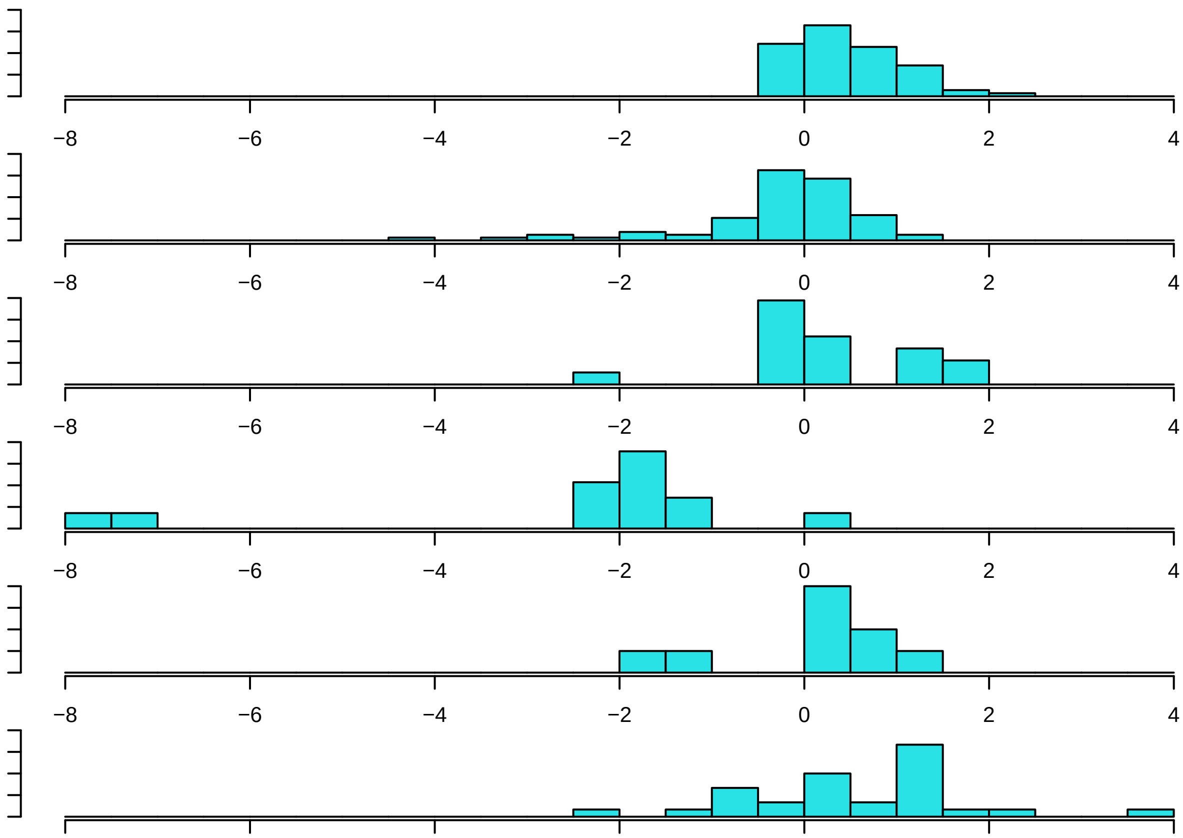
```r
#the first LDA1, it contains 0.8145 of the data.
#The linear discriminant function from the result in above is
#0.94*RI+1.944Na+1.068*Mg+1.667*Al+1.8989*Si+1.024*K+
#1.432*Ca+1.15*Ba-0.0498*Fe


#(the variable returned by the lda() function) is the percentage
#separation achieved by each discriminant function.
#LDA1 independently achieve a separation of 0.8145.

two_lda = Coefficients_of_linear_discriminants[,2]
two_lda
```

```
##         RI         Na         Mg         Al         Si          K         Ca
## 0.08925658 2.58461557 4.30684515 0.86111013 2.32855629 1.21439159 3.37701884
##         Ba         Fe
## 1.71202472 0.02110900
```

```r
#LDA2 independently achieve a separation of 0.1169.

lda.model_values = predict(lda.model)
#linear regression model for the hitogram

par(mar=c(1, 1, 1, 1))
ldahist(lda.model_values$x[,1], g = type)
```



```r
#From the graph above, we have histogram from LD1,
#the type 1,2,3 have some overlap between them
#and we can see that the separation between 1, 2, 3 and the other three Species
#is quite small  with some overlap. On the contrary, there is a certain amount
#of overlapping between type 5,6,7. We already said that the tight
#percentage of separation archived by LD1 is 81.45%, that is there is some
# clear separation from the histogram above.

#Now, we can try to do the same for LD2.

ldahist(lda.model_values$x[,2], g = type)
```




```r
#the distance between the separation between the type is very small and we can
#see the overlapping between all the types of glasses with no clear separation.
#this is expected due to the smaller separation of 11.69% achieved by LDA.
#The LDA does not provide a good separtion between the glass types.

#tsne_out = Rtsne(Glass[,-10], perplexity=5, theta = 0.0,
                  #max_iter = 2000, num_threads = 6)

#df = data.frame(x=tsne_out$Y[,1], y= tsne_out$Y[,2], type = Glass[,10])
#ggplot(data=df, aes(x=x, y=y, group=type, color=type))+geom_point()
```

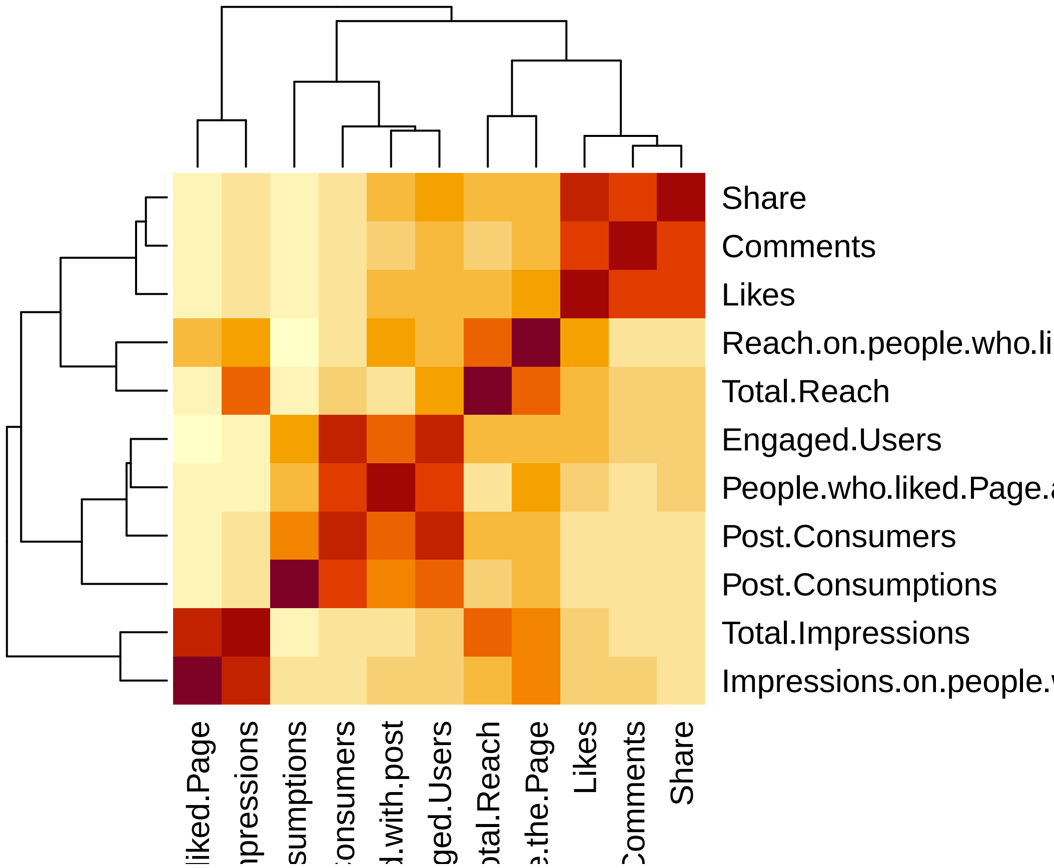
```r
#2
FB_metric = read.csv(file = 'FB-metrics.csv',
                     header = TRUE, sep = ",",
                     quote = "\"" , dec = ".",
                     fill =TRUE, comment.char = "")
#FB_metric
#summary(FB_metric)

FB_metric_new = FB_metric[,8:18]
#extract the 11 columns needed for the analysis

#str(FB_metric_new)
#the summary of the each columns
FB_pca = prcomp(FB_metric_new, scale= T)
#model the pca on the 11 variable dataframe to reduce the dimension.
summary(FB_pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.4300 1.3091 1.2707 0.79219 0.68482 0.55935 0.38343
## Proportion of Variance 0.5368 0.1558 0.1468 0.05705 0.04263 0.02844 0.01337
## Cumulative Proportion  0.5368 0.6926 0.8394 0.89646 0.93910 0.96754 0.98090
##                            PC8     PC9    PC10    PC11
## Standard deviation     0.35002 0.27073 0.11706 0.02340
## Proportion of Variance 0.01114 0.00666 0.00125 0.00005
## Cumulative Proportion  0.99204 0.99870 0.99995 1.00000
```

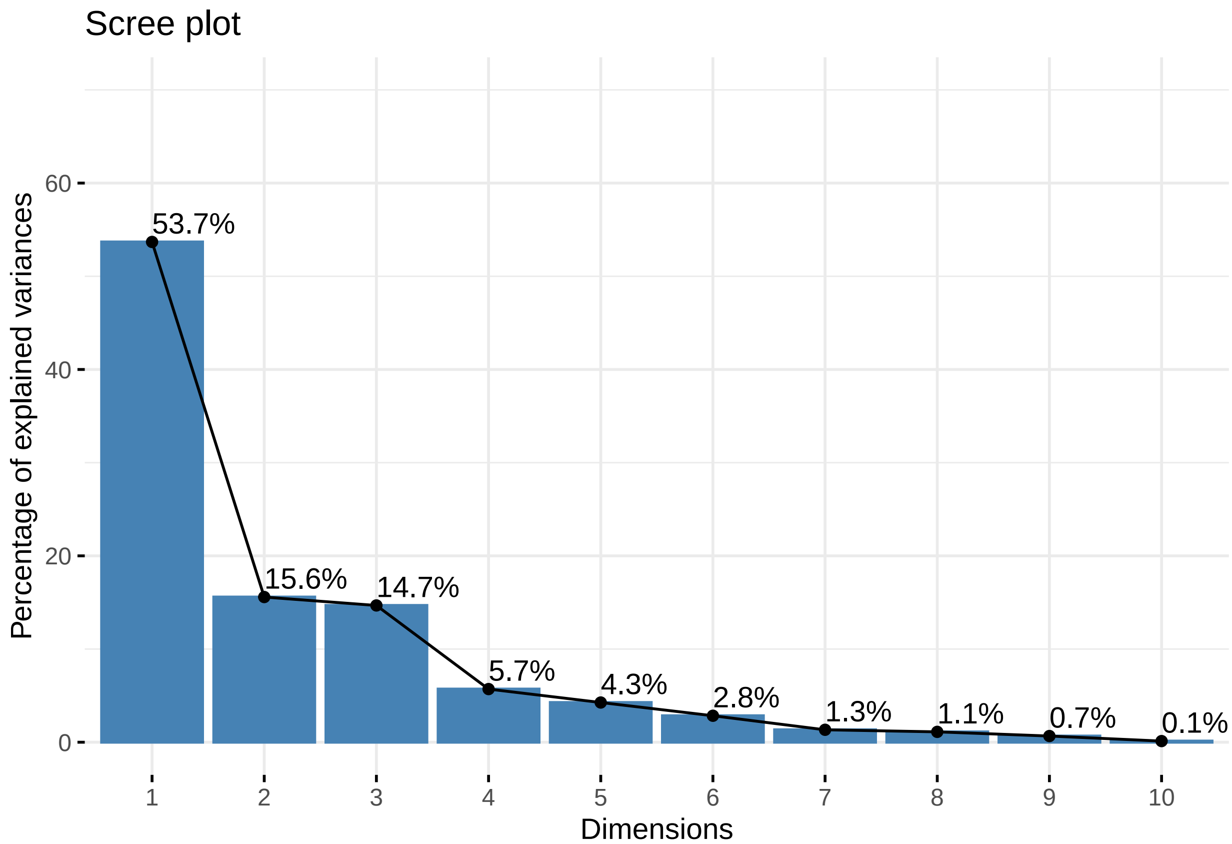
```r
#gives the summary. the first three PCs give a cumulative of 84%

rbind(
  SD = sqrt(FB_pca$sd^2),
  per_var = (FB_pca$sd^2/sum(FB_pca$sd^2))*100,
  per_var_cum = (cumsum(FB_pca$sd^2)/sum(FB_pca$sd^2))*100)
```

```
##                  [,1]      [,2]      [,3]       [,4]       [,5]      [,6]
## SD           2.430039  1.309138  1.270658  0.7921942  0.6848185  0.559352
## per_var     53.682618 15.580374 14.677926  5.7051963  4.2634213  2.844315
## per_var_cum 53.682618 69.262991 83.940917 89.6461136 93.9095348 96.753850
##                   [,7]       [,8]       [,9]      [,10]        [,11]
## SD           0.3834277  0.3500222  0.2707292  0.1170572 2.339721e-02
## per_var      1.3365164  1.1137774  0.6663120  0.1245672 4.976631e-03
## per_var_cum 98.0903668 99.2041441 99.8704562 99.9950234 1.000000e+02
```

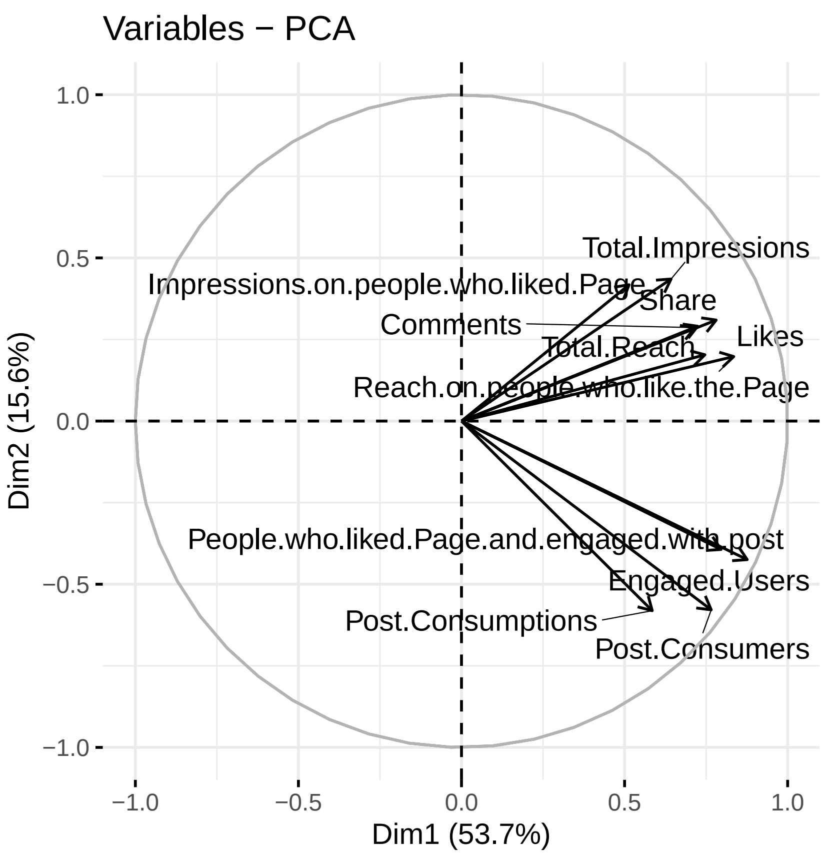
```r
heatmap(cor(FB_metric_new))
```



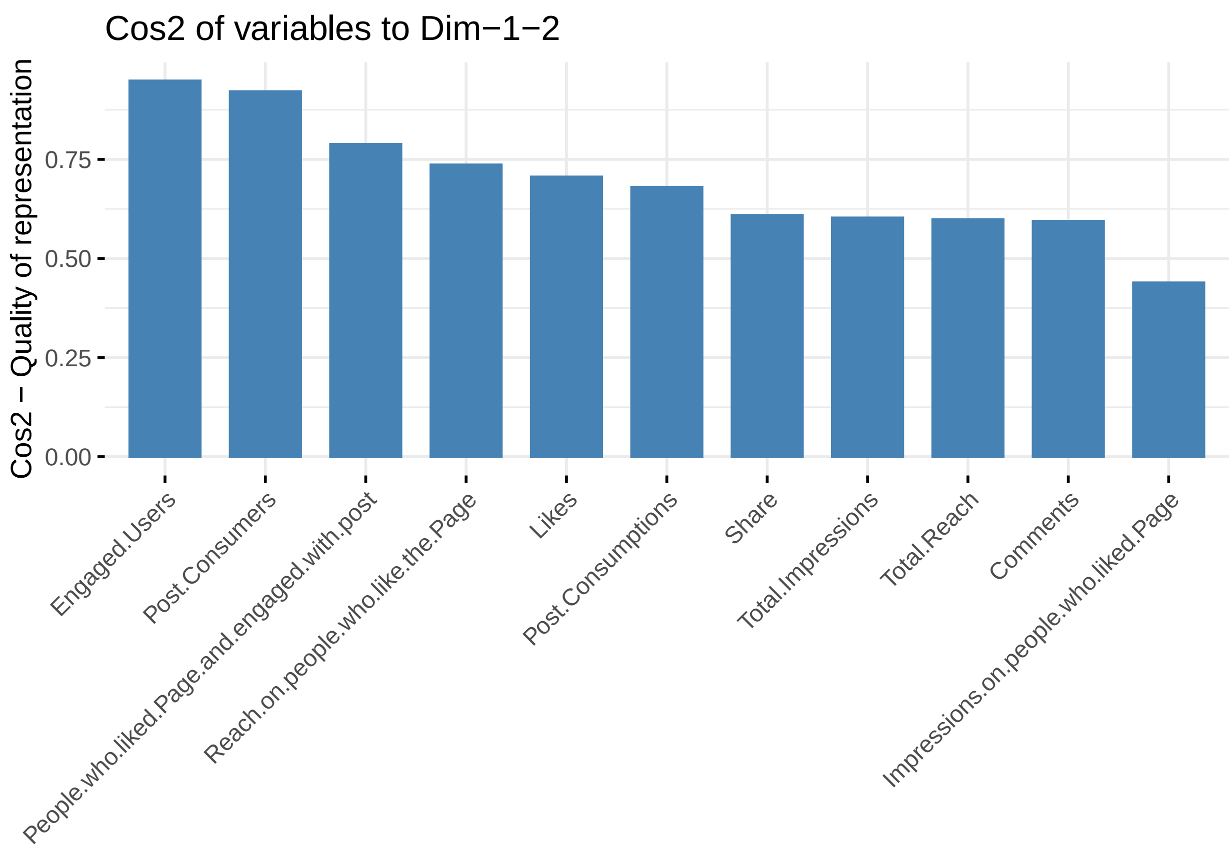

```
#shows the closely correlated variables in the data. Total impression and
#reach on people who liked the page and engagged the post are highly correlated.
corMat_FB = cor(FB_metric_new)
#corMat_FB

fviz_eig(FB_pca, addlabels = TRUE, ylim = c(0, 70))
```




```r
#the first three dimensions represents 84% of the data.
#4 dimensions represents 89.7%. The data can be well represented with
#4 PCs

fviz_pca_var(FB_pca, col.var = "black", repel = TRUE)
```



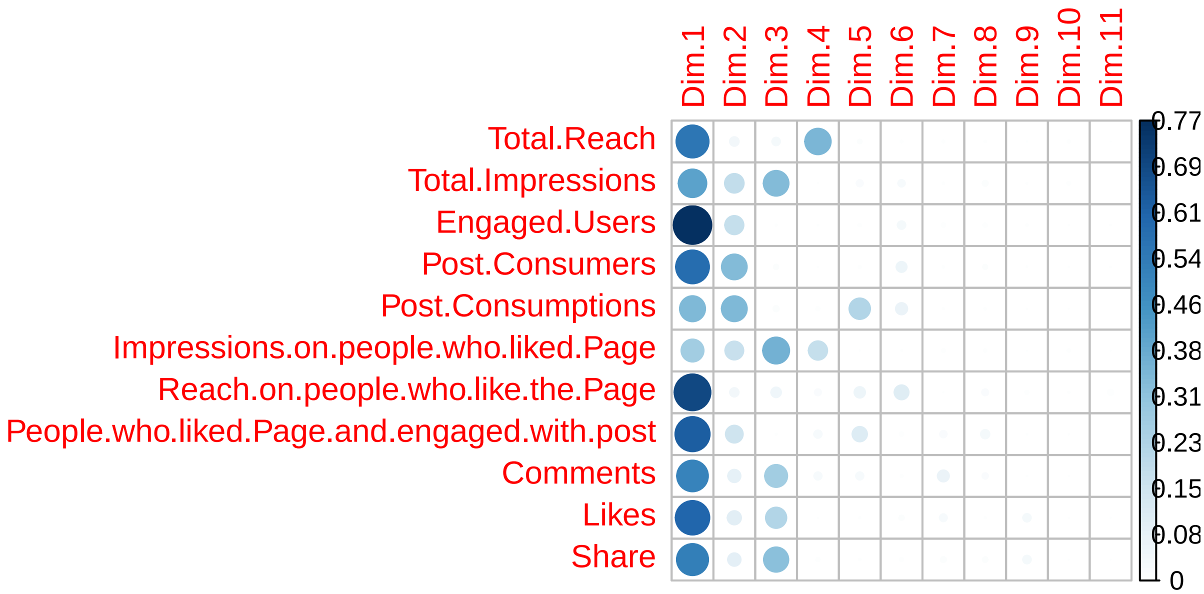

```r
#the bilpot shows hows the variables are represented on the PCA biplot.
#The longer length arrow have a higher weight/effect on the PC 1 and PC2.
#the PC1 has more effect on the the representation. This is shown on the plot
#as more of the variables increase towards the horizontal direction.

fviz_cos2(FB_pca, choice = "var", axes = 1:2)
```



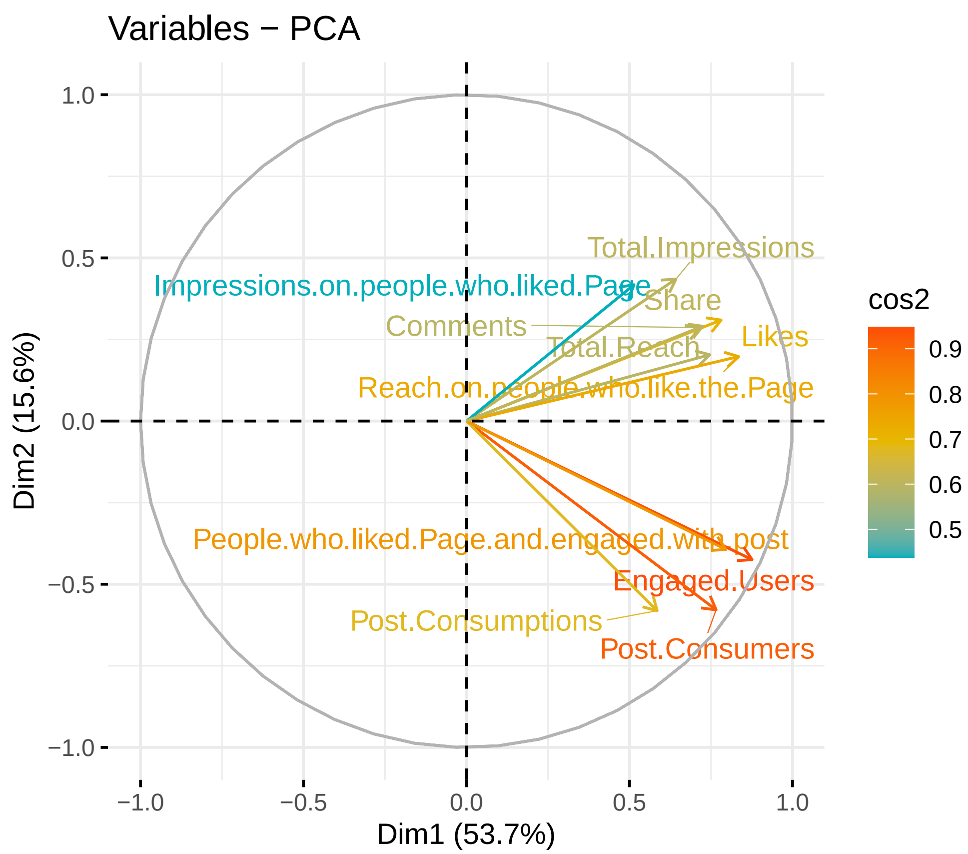

```r
#visualize the cos2 to represent the quality of
#the variables on factor map. square cosine, squared coordinates.
# The number of engaged users has the highest effect and impressions
#on people has the lowest effect on the PC1 and PC2.

library("corrplot")
```

```
## corrplot 0.84 loaded
```

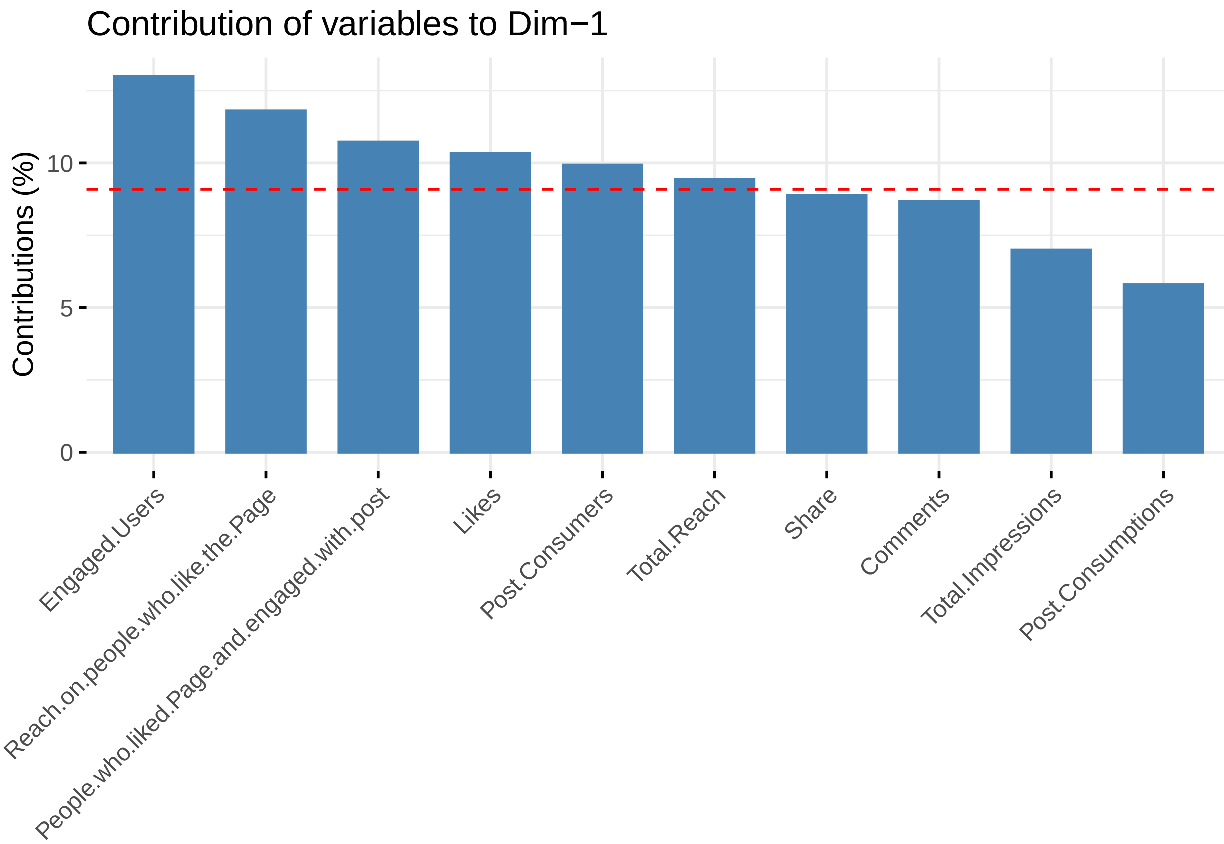
```r
var = get_pca_var(FB_pca)
corrplot(var$cos2, is.corr=FALSE)
```



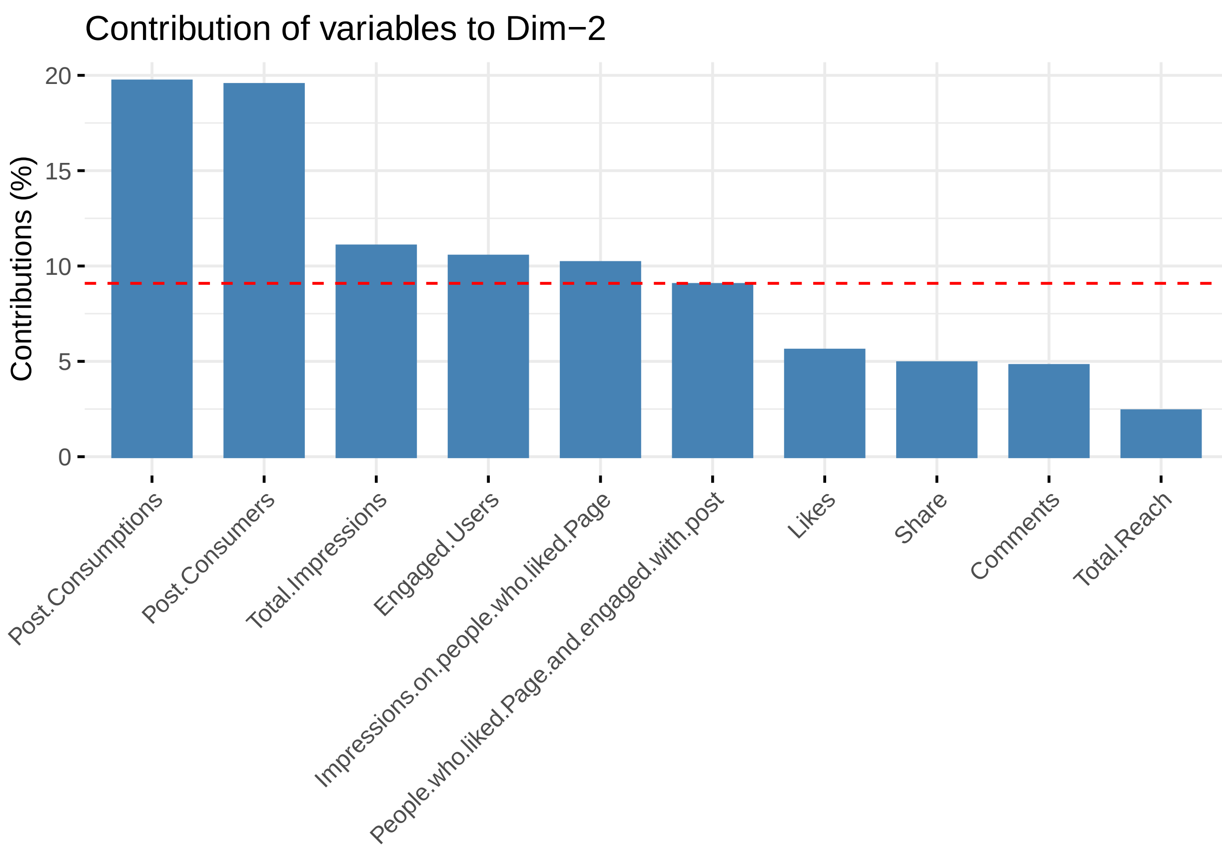

```r
fviz_pca_var(FB_pca, col.var = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE # Avoid text overlapping
)
```



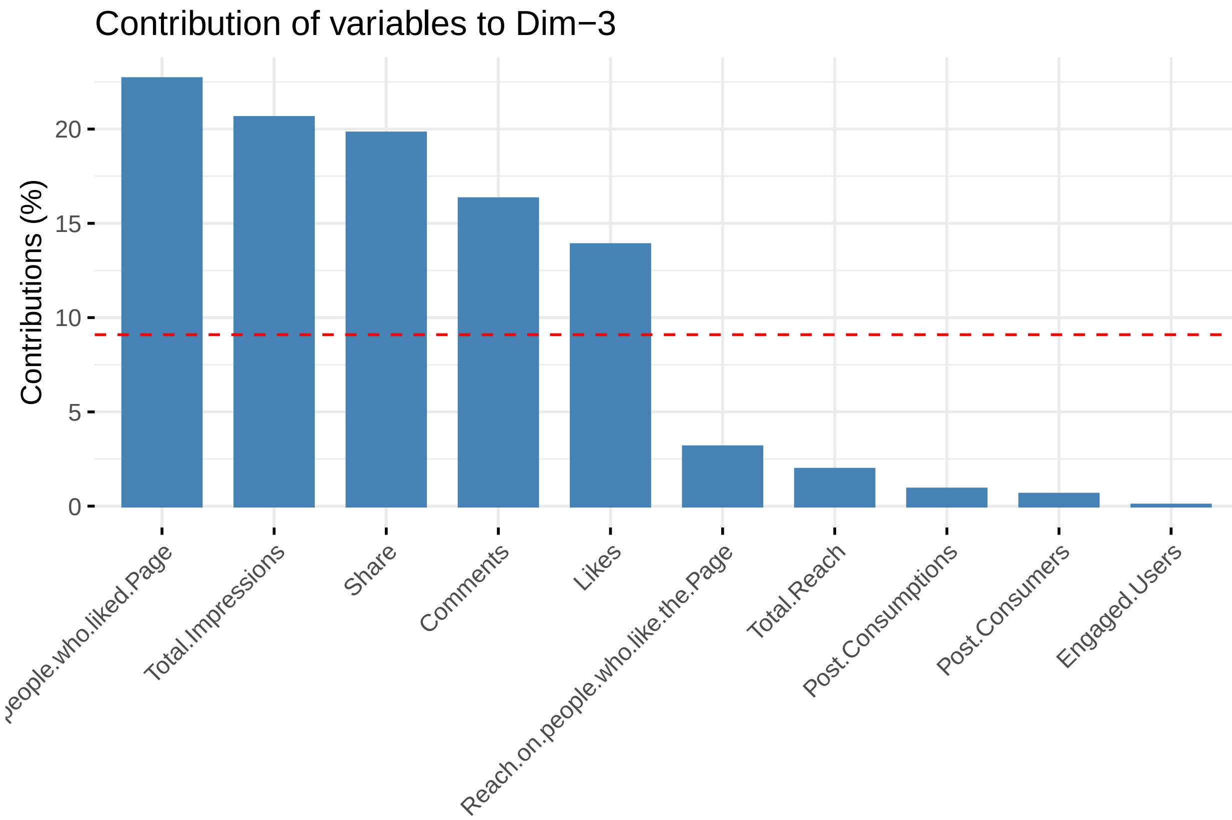

```r
#The cos2 values are used to estimate the quality of the representation
#The closer a variable is to the circle of correlations, the better its
#representation on the factor map
#(and the more important it is to interpret these components)
#Variables that are closed to the center of the plot are
#less important for the first components.
#variables with high correlations are colored red, blue for the low effects
#and orange for average. The varibales with the highest imapact is the
#post consumers and the engaged users.

#scatter plot matrix for the PCS

# Contributions of individual variables to PC1
fviz_contrib(FB_pca, choice = "var", axes = 1, top = 10)
```



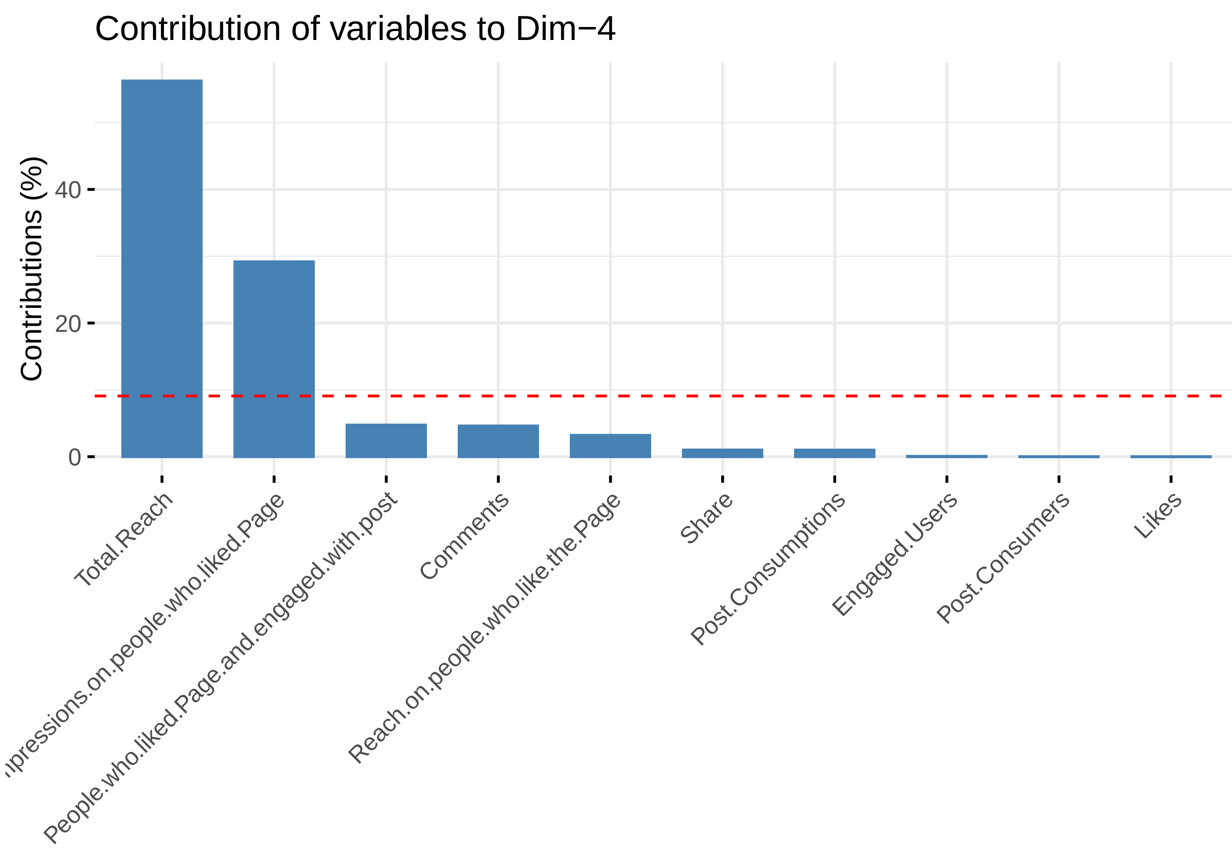

```
# Contributions of variables to PC2
fviz_contrib(FB_pca, choice = "var", axes = 2, top = 10)
```



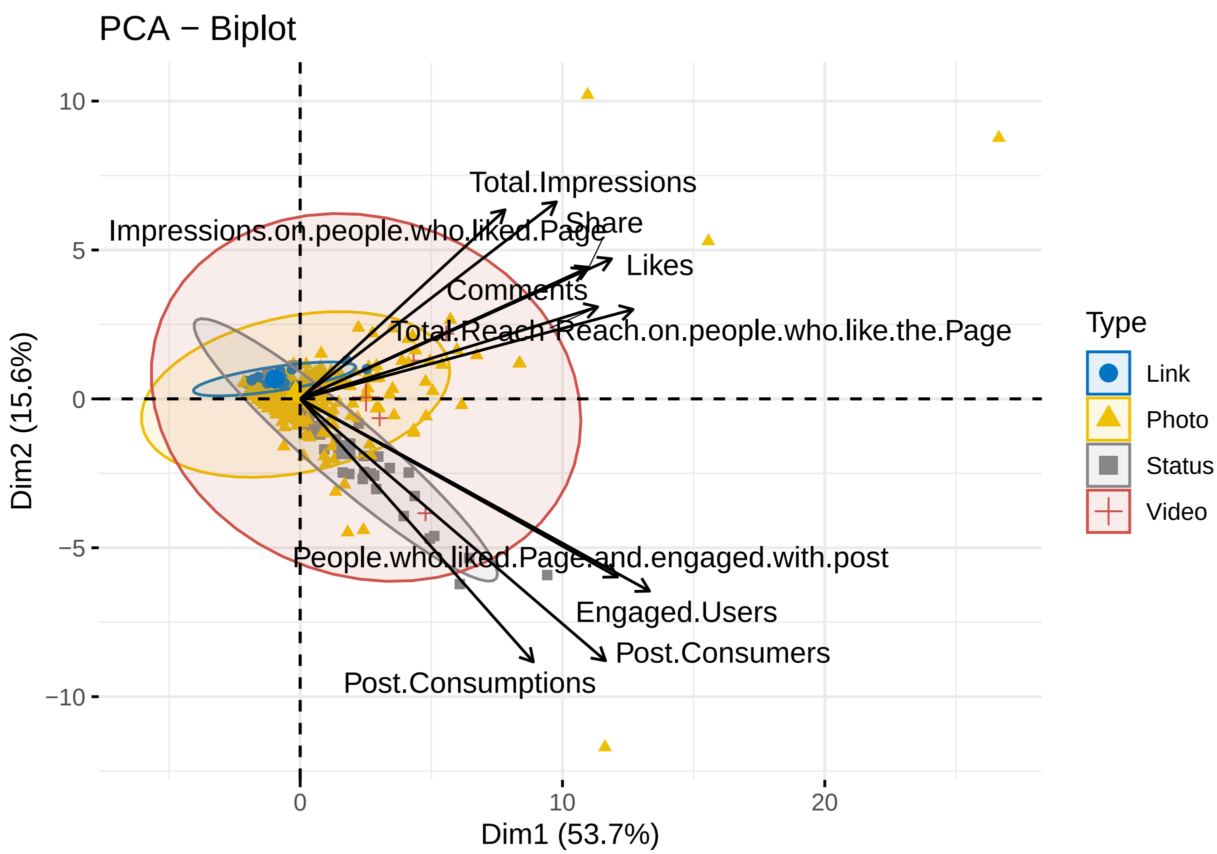

```r
# Contributions of variables to PC3
fviz_contrib(FB_pca, choice = "var", axes = 3, top = 10)
```



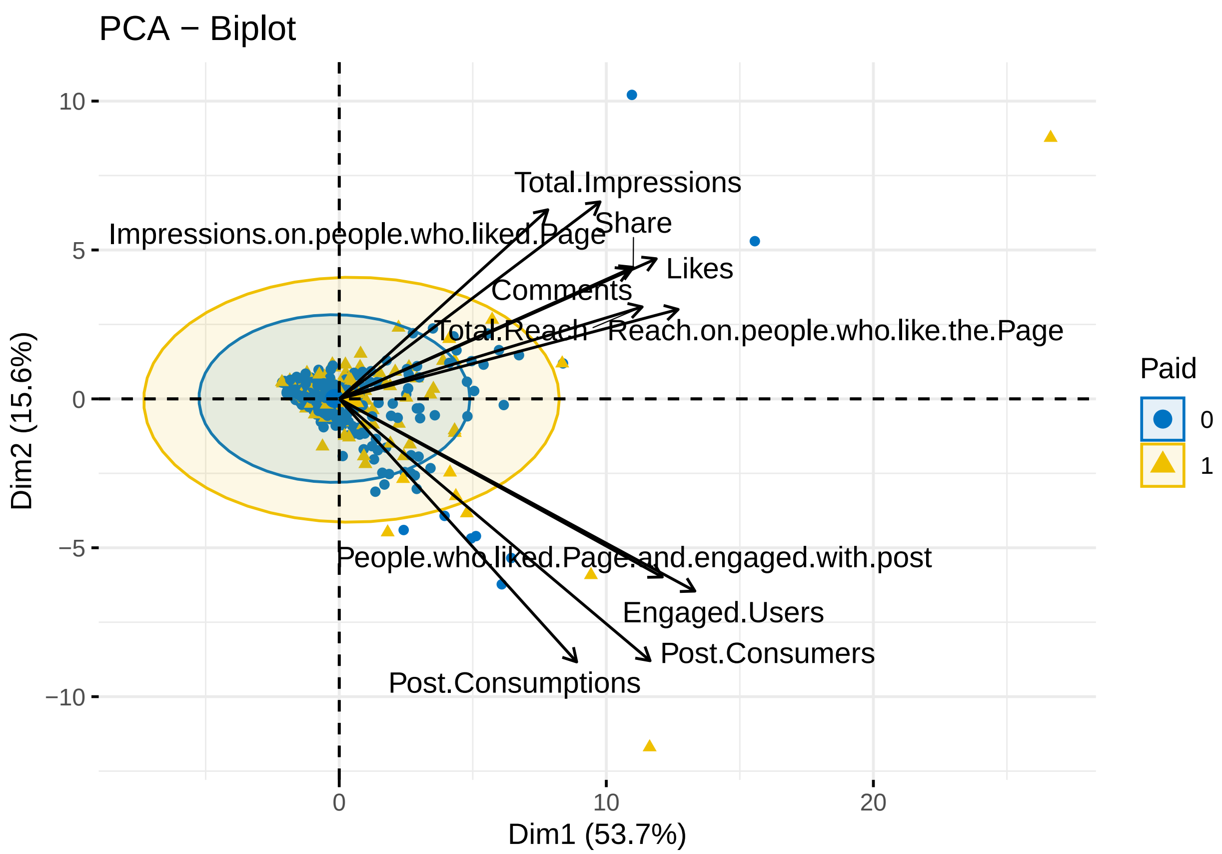

```
# Contributions of variables to PC4
fviz_contrib(FB_pca, choice = "var", axes = 4, top = 10)
```



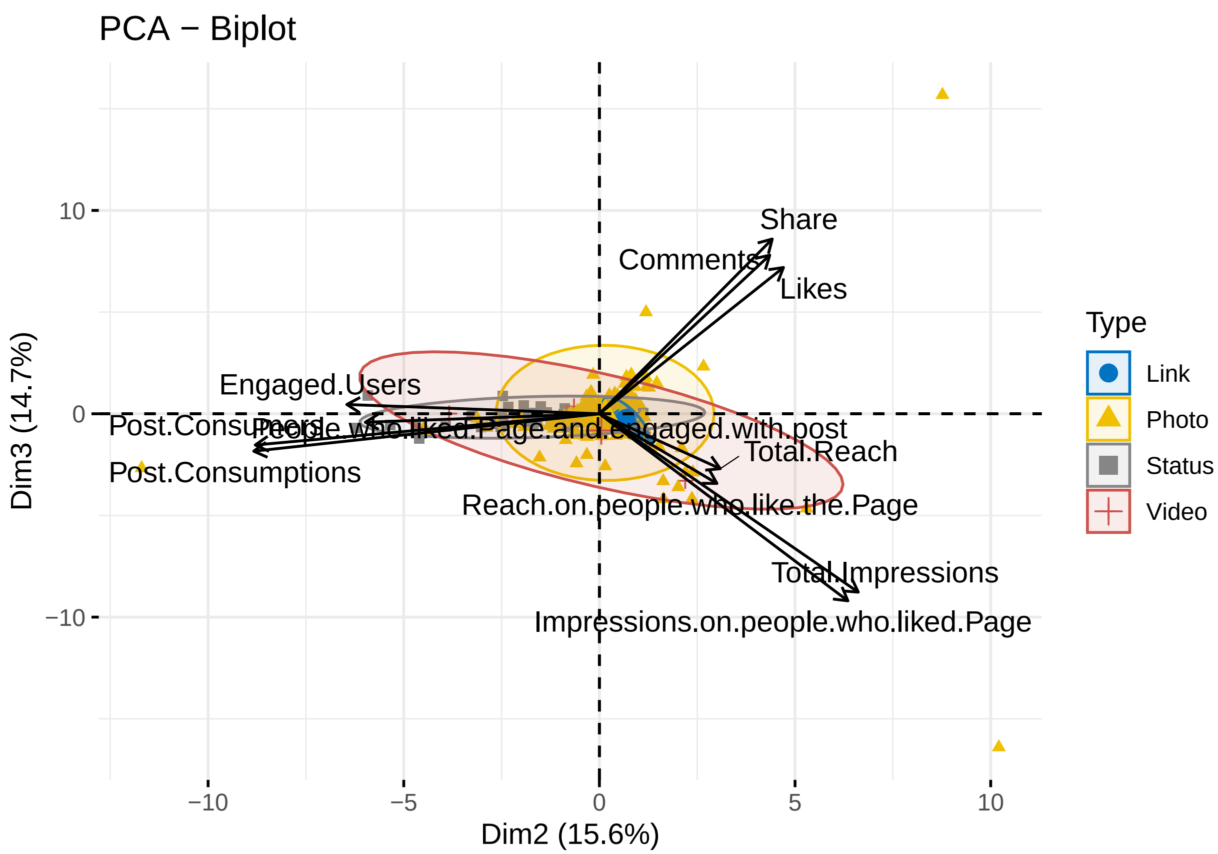

```
fviz_pca_biplot(FB_pca,
                col.ind = FB_metric$Type, palette = "jco",
                addEllipses = TRUE, label = "var",
                col.var = "black", repel = TRUE,
                legend.title = "Type")
```



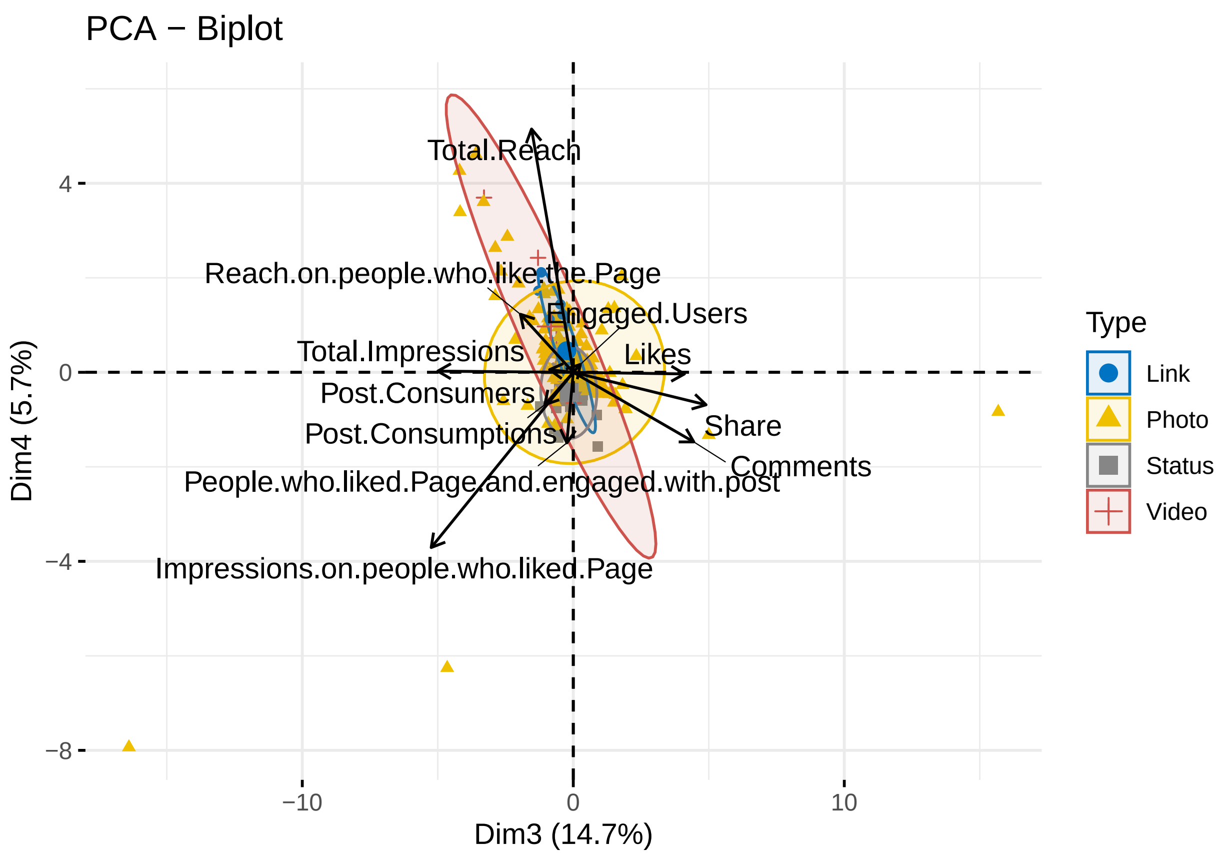

```r
#the data is classified and colored with regards to the Type of media
#link - blue, Photo - yellow, Status - grey, Video -red. The clusters can be
#seen to overlap on the two dimensions.
#This shows they either have close properties
#or the PCA does not do a good job with separation. This biplot is for the PC1
#and PC2

fviz_pca_biplot(FB_pca,
                col.ind = as.factor(FB_metric$Paid), palette = "jco",
                addEllipses = TRUE, label = "var",
                col.var = "black", repel = TRUE,
                legend.title = "Paid")
```



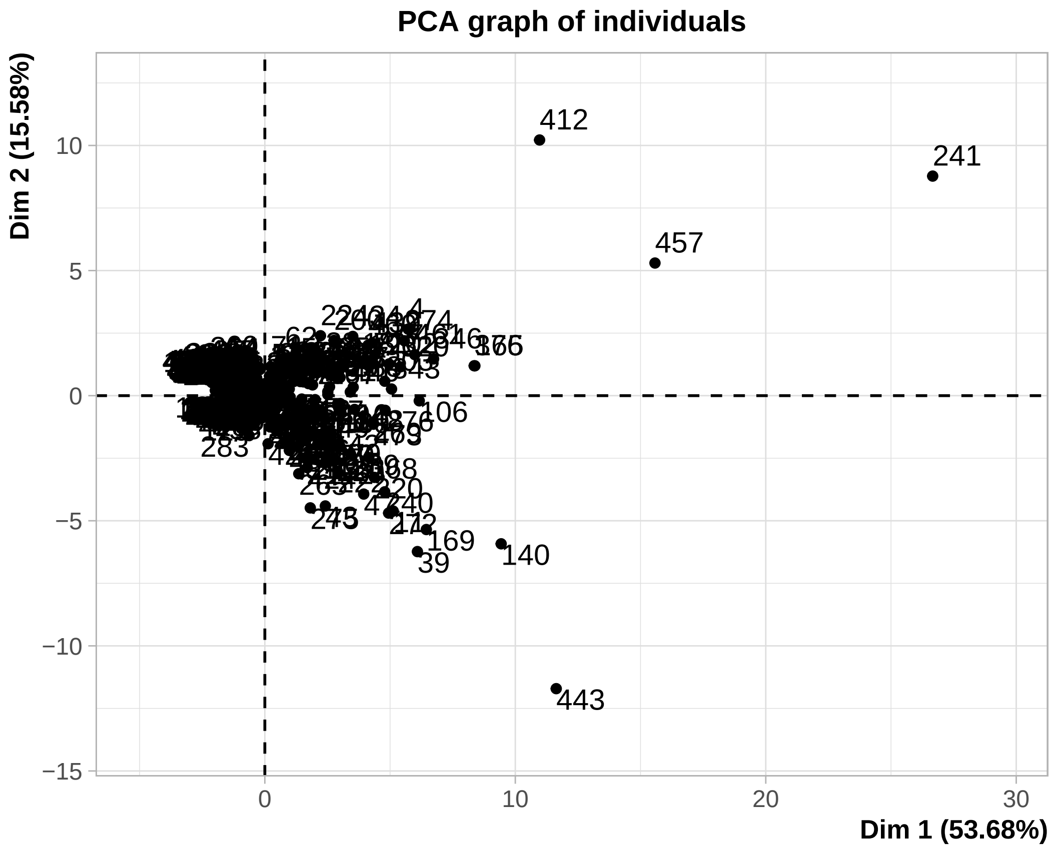

```r
#the data is classifed according to whether it was paid or unpaid. This shows
#the paid ad more largr reach and an increase in Likes, comments,
#total impressions,
#and other variables i.e arrows on the biplot. This biplot is for the PC1 and
#PC2
fviz_pca_biplot(FB_pca,
                col.ind = FB_metric$Type, palette = "jco",
                addEllipses = TRUE, label = "var",
                col.var = "black", repel = TRUE,
                legend.title = "Type", axes = c(2,3))
```

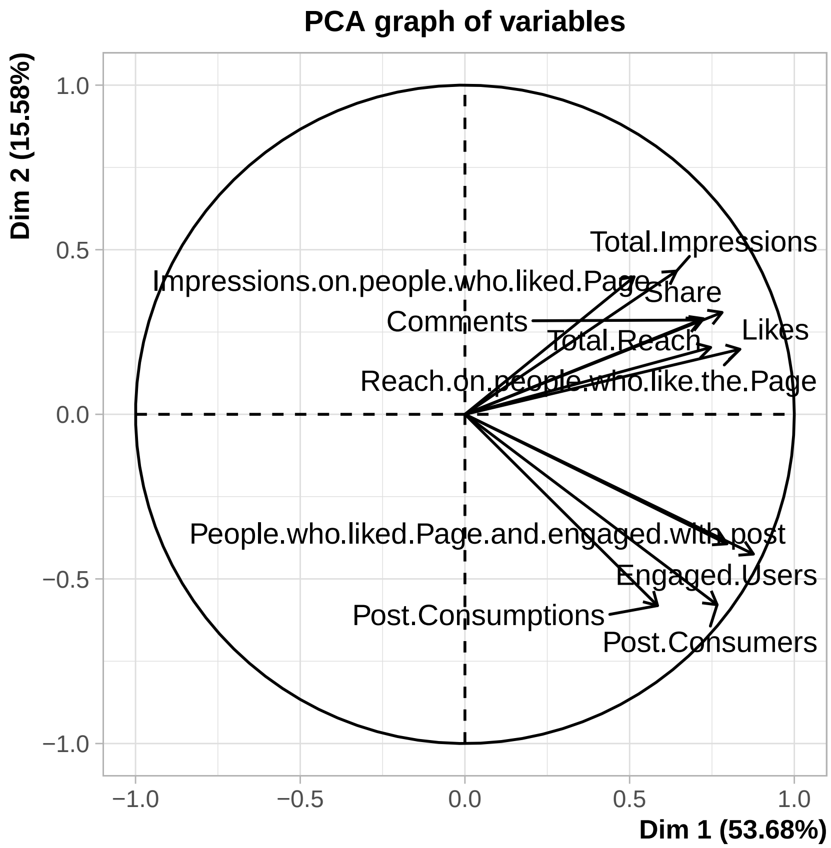



```r
#this plot is for PC 2 and PC 3. the separation in this section is less clear as
#this two dimensions have lower representation of the data.

fviz_pca_biplot(FB_pca,
                col.ind = FB_metric$Type, palette = "jco",
                addEllipses = TRUE, label = "var",
                col.var = "black", repel = TRUE,
                legend.title = "Type", axes = c(3,4))
```




```
#this plot is for PC 3 and PC 4. the separation in this section is less clear as
#this two dimensions have lower representation of the data.


FB_PCA = PCA(FB_metric_new)
```


PCA graph of individuals
Dim 2 (15.58%)
Dim 1 (53.68%)
10
5
0
−5
−10
−15
0
10
20
30
412
241
457
106
283
169
140
39
443




```
FB.desc <- dimdesc(FB_PCA)
# Description of dimension 1
FB.desc$Dim.1
```

```
## $quanti
##                                             correlation       p.value
## Engaged.Users                                 0.8760239 2.975835e-158
## Reach.on.people.who.like.the.Page             0.8347720 7.310765e-130
## People.who.liked.Page.and.engaged.with.post   0.7957610 1.885528e-109
## Likes                                         0.7808731 8.407496e-103
## Post.Consumers                                0.7656779  1.560501e-96
## Total.Reach                                   0.7461734  3.755787e-89
## Share                                         0.7240445  1.481687e-81
## Comments                                      0.7154004  8.704826e-79
## Total.Impressions                             0.6424995  5.635403e-59
## Post.Consumptions                             0.5847982  9.574201e-47
## Impressions.on.people.who.liked.Page          0.5137677  1.080732e-34
##
## attr(,"class")
## [1] "condes" "list "
```

```
#dimension description is used to identify the
#most significantly associated variables with a given principal component.
#the most associated in this case of Dim1  is Engaged user with weight of 0.876
FB.desc$Dim.2
```

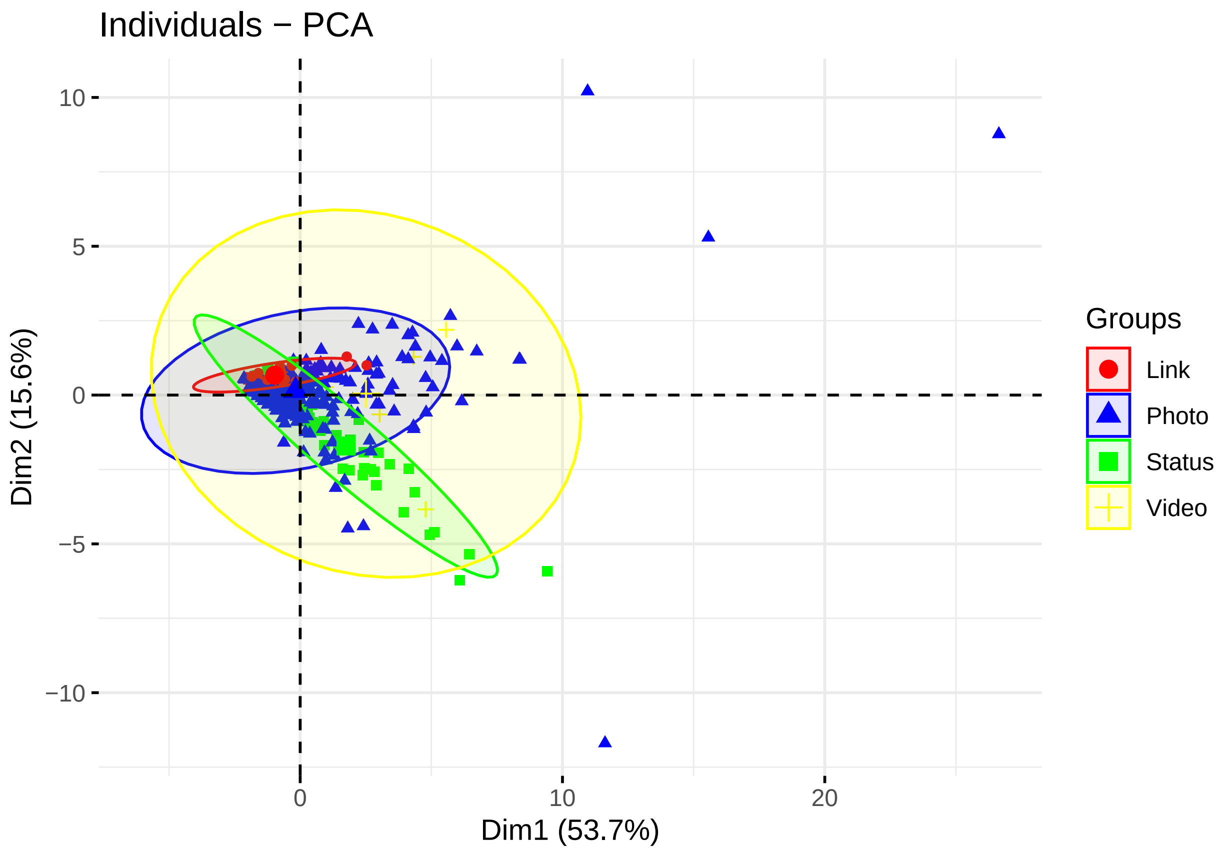
```
## $quanti
##                                           correlation      p.value
## Total.Impressions                          0.4352111 2.707651e-24
## Impressions.on.people.who.liked.Page       0.4177455 2.509697e-22
## Likes                                      0.3094808 1.898882e-12
## Share                                      0.2905969 4.345078e-11
## Comments                                   0.2863899 8.462583e-11
## Total.Reach                                0.2031995 5.186438e-06
## Reach.on.people.who.like.the.Page          0.1975940 9.475321e-06
## People.who.liked.Page.and.engaged.with.post -0.3934374 8.956885e-20
## Engaged.Users                             -0.4246145 4.361828e-23
## Post.Consumers                            -0.5784279 1.529802e-45
## Post.Consumptions                         -0.5810974 4.825099e-46
##
## attr(,"class")
## [1] "condes" "list "
```

```r
#the most associated in this case of Dim1  is total
#impressions with a weight of 0.435 with a positive correlation.
FB.desc$Dim.3
```

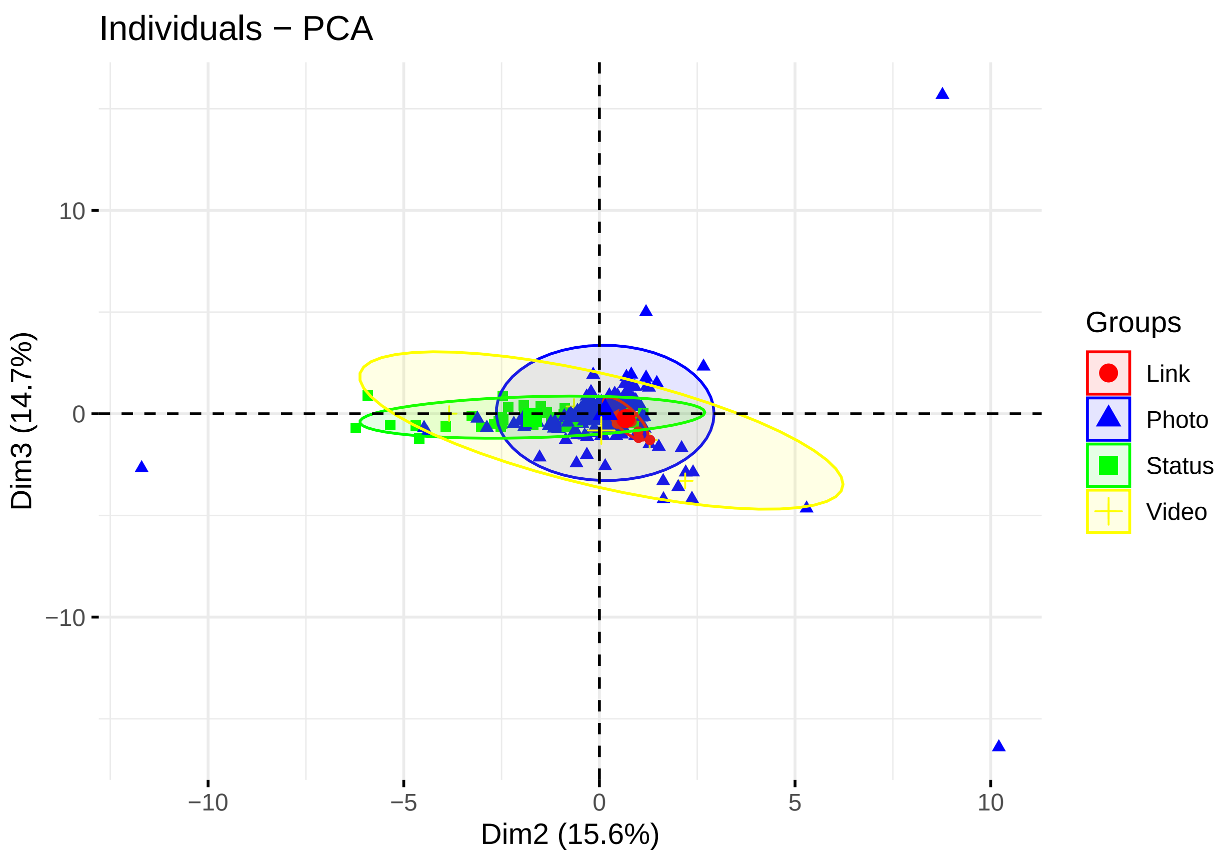
```
## $quanti
##                                     correlation      p.value
## Impressions.on.people.who.liked.Page   0.6050741 9.281529e-51
## Total.Impressions                      0.5769264 2.913931e-45
## Reach.on.people.who.like.the.Page      0.2253552 4.052445e-07
## Total.Reach                            0.1776483 7.064991e-05
## Post.Consumptions                      0.1209929 7.038346e-03
## Post.Consumers                         0.1010239 2.459468e-02
## Likes                                 -0.4732171 5.460178e-29
## Comments                              -0.5130969 1.362718e-34
## Share                                 -0.5652988 3.832155e-43
##
## attr(,"class")
## [1] "condes" "list "
```

```r
#pca for individuals
ind  = get_pca_ind(FB_pca)

# Coordinates of individuals
#head(ind$coord)
# Quality of individuals
#head(ind$cos2)
# Contributions of individuals
#head(ind$contrib)
#the individuals cannot be plotted as they are very many and would not have a good
#interpretation.

fviz_pca_ind(FB_pca,
             geom.ind = "point", # show points only (nbut not "text")
             col.ind = FB_metric$Type, # color by groups
             palette = c("red", "blue", "green", "yellow"),
             addEllipses = TRUE, # Concentration ellipses
```

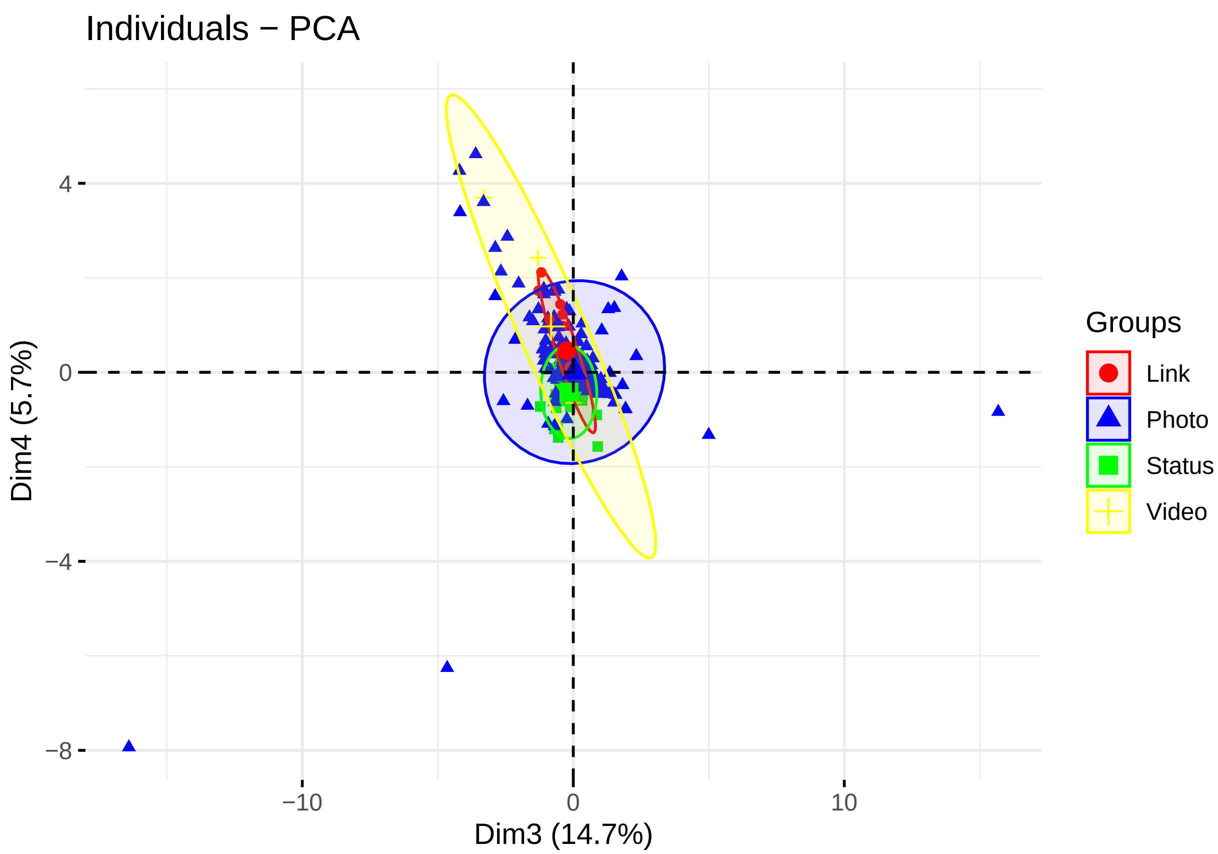
```
             legend.title = "Groups"
)
```



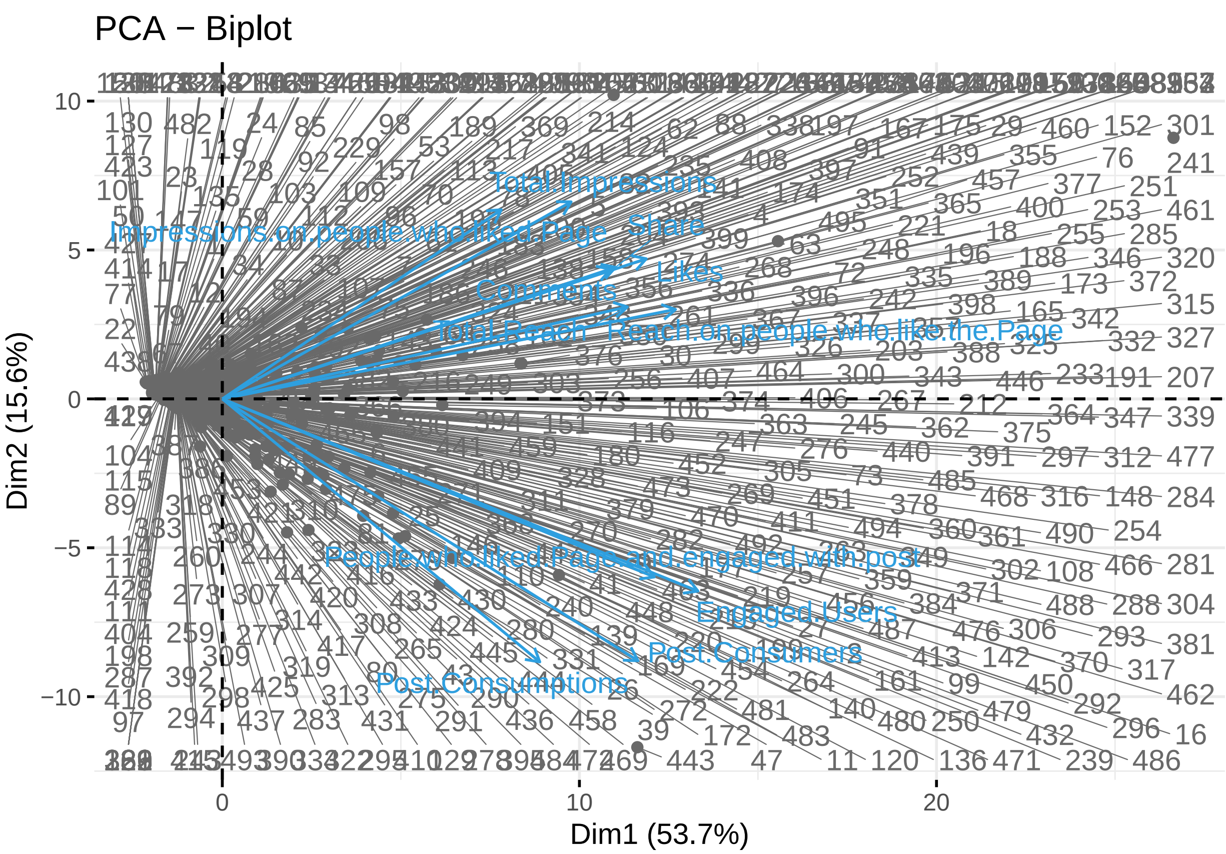

```
#these shows the various type of post and the colors. The types have a large
#representation because of the high representation of PC 1 and PC2 which are 53.7%
#and 15.6% respectively. A clear distinction cannot be seen as they are overlapping
#but PCA is not a very good clustering method but it is for unsupervised learning.
# a good dimension reduction is observed,
fviz_pca_ind(FB_pca,
             geom.ind = "point", # show points only (nbut not "text")
             col.ind = FB_metric$Type, # color by groups
             palette = c("red", "blue", "green", "yellow"),
             addEllipses = TRUE, # Concentration ellipses
             legend.title = "Groups",
             axes = c(2,3)
)
```



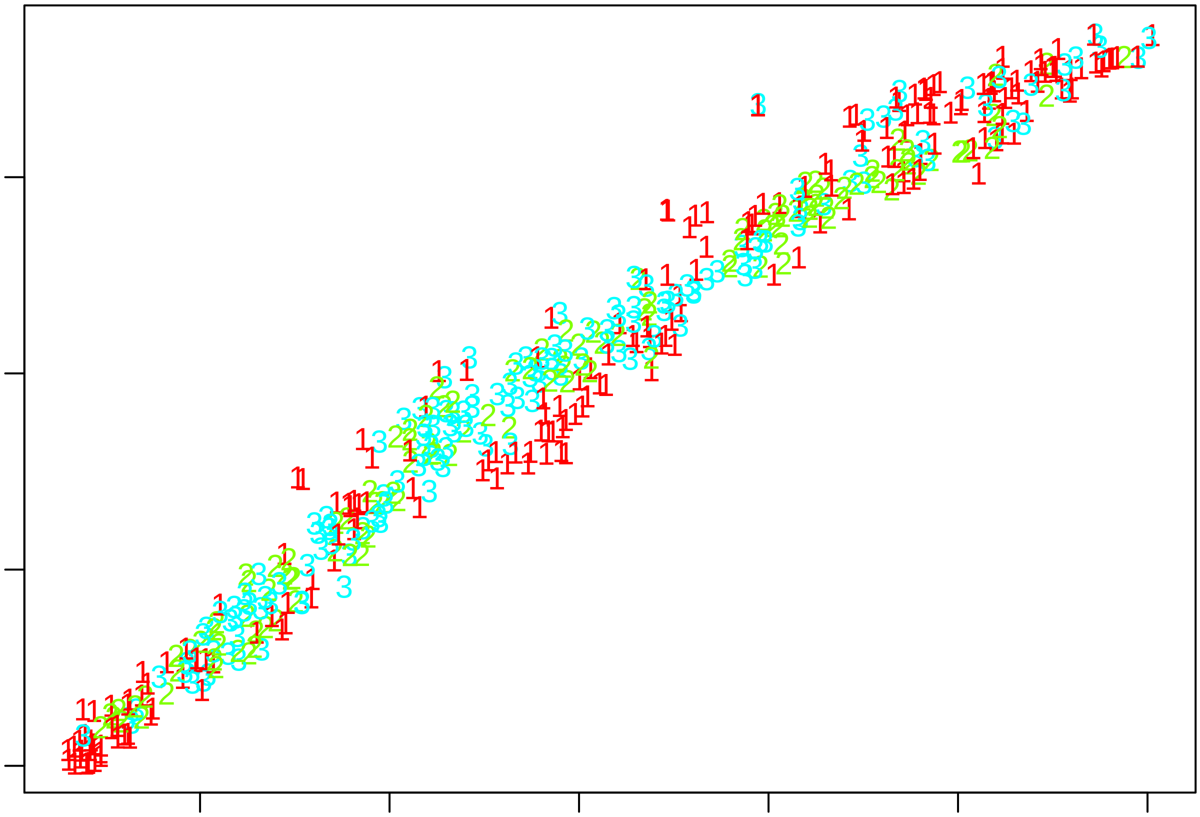

```r
#project on dimensions 2 and 3. The groups have smaller circles here which show
#smaller imapact of the post type on these PCs this is because they have smaller
#representations of 15.6% and 14.7%. They have less percentage of the total data.

fviz_pca_ind(FB_pca,
             geom.ind = "point", # show points only (nbut not "text")
             col.ind = FB_metric$Type, # color by groups
             palette = c("red", "blue", "green", "yellow"),
             addEllipses = TRUE, # Concentration ellipses
             legend.title = "Groups",
             axes = c(3,4)
)
```




```r
#project on dimensions 3 and 4. They have lesser percentage of the total columns
#as they only represent 14.7% and 5.7% repectively.

fviz_pca_biplot(FB_pca, repel = TRUE,
                col.var = "#2E9FDF", # Variables color
                col.ind = "#696969"  # Individuals color
)
```



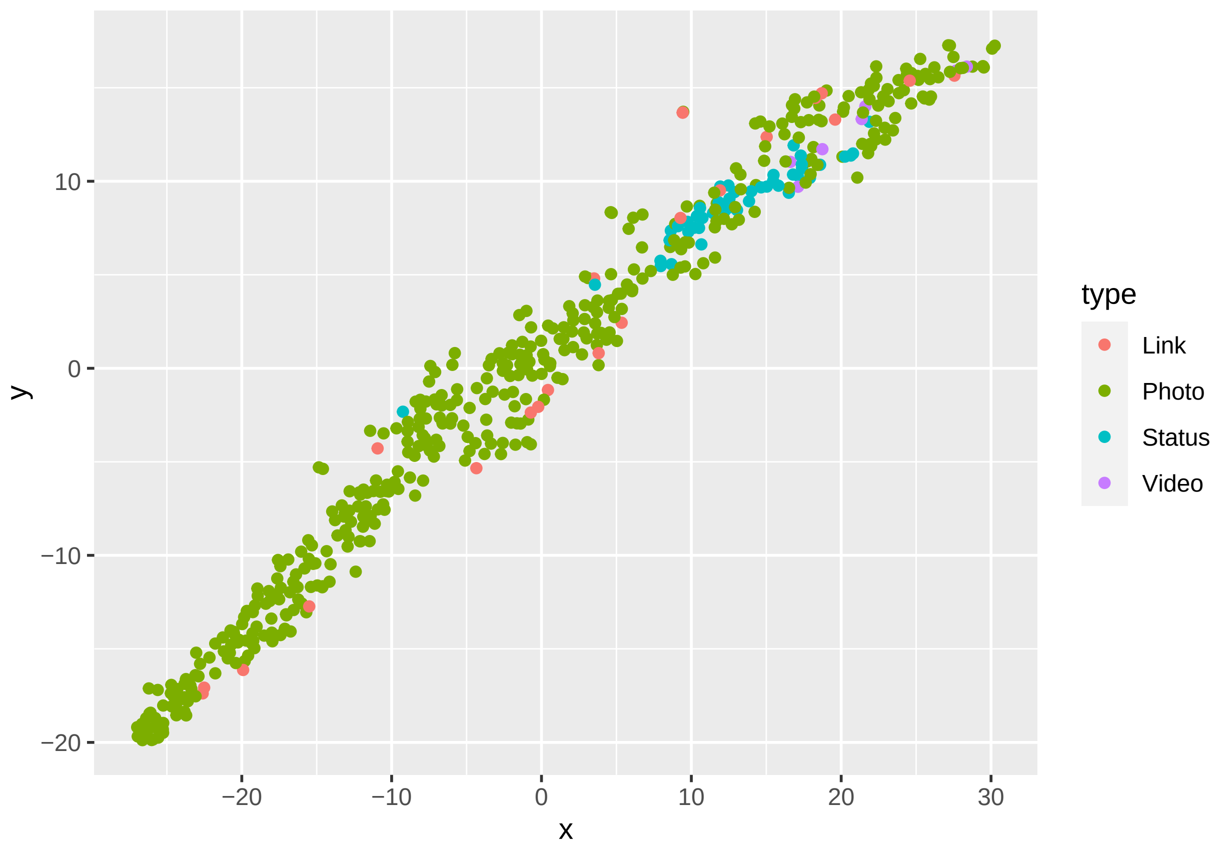

```
#THE BILPOT PLOT is unable to read due to the large number of varibales
#an individual that is on the same side of a
#given variable has a high value for this variable;
#an individual that is on the opposite side of a
#given variable has a low value for this variable.


#b
tsne_out = Rtsne(FB_metric_new[,1:11], perplexity=50,
                 theta = 0.0, max_iter = 5000, num_threads = 6)

#tsne_out
cols = rainbow(4)
plot(tsne_out$Y, t='n')
text(tsne_out$Y, labels=(FB_metric[,3]), col=cols[FB_metric[,3]])
```



```r
#the shows the category of the data colored according to the number of the
#category. The tsne do not show good clustering for the catergory type.


str(FB_metric)
```

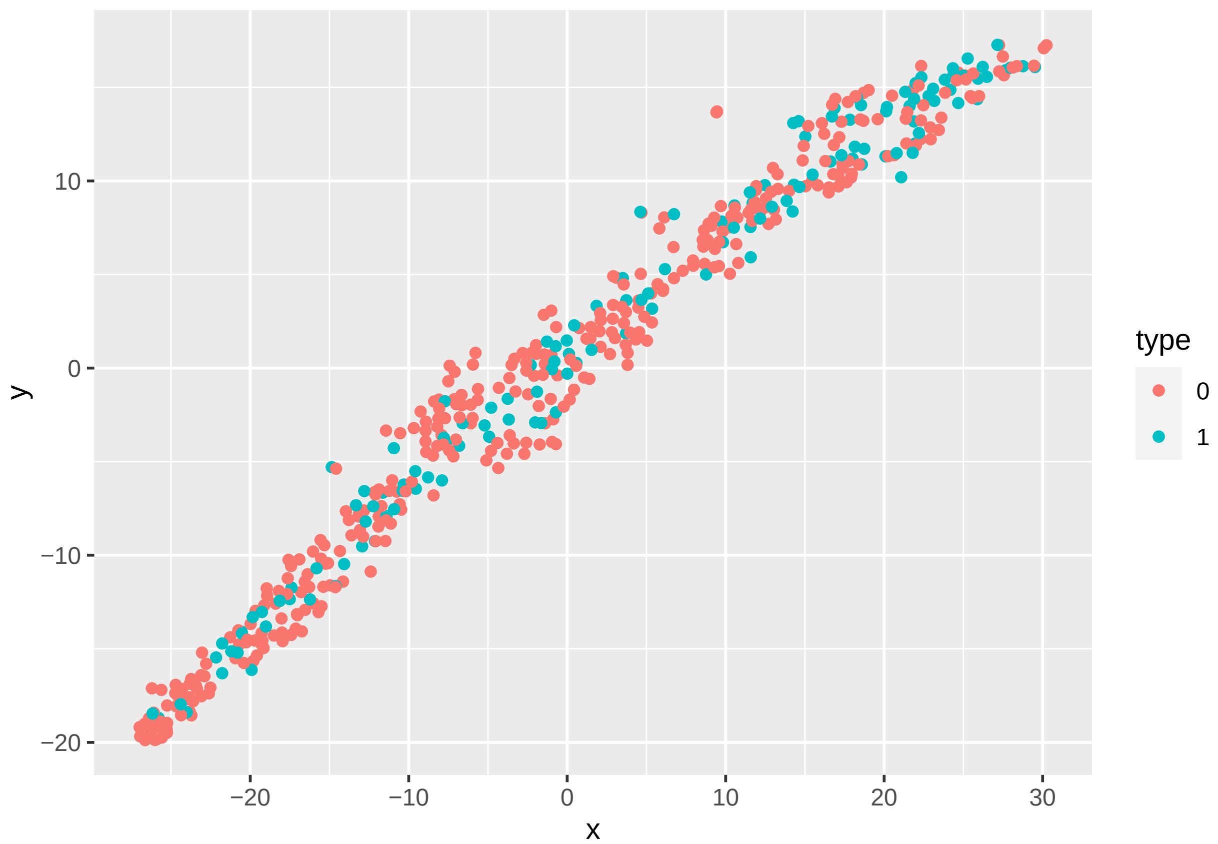
```
## 'data.frame':    495 obs. of  18 variables:
##  $ Page.total.likes                                   : int  139441 139441 139441 139441 139441 139441 13944
##  $ Type                                               : chr  "Photo" "Status" "Photo" "Photo" ...
##  $ Category                                           : int  2 2 3 2 2 2 3 3 2 3 ...
##  $ Post.Month                                         : int  12 12 12 12 12 12 12 12 12 12 ...
##  $ Post.Weekday                                       : int  4 3 3 2 2 1 1 7 7 6 ...
##  $ Post.Hour                                          : int  3 10 3 10 3 9 3 9 3 10 ...
##  $ Paid                                               : int  0 0 0 1 0 0 1 1 0 0 ...
##  $ Total.Reach                                        : int  2752 10460 2413 50128 7244 10472 11692 13720 11
##  $ Total.Impressions                                  : int  5091 19057 4373 87991 13594 20849 19479 24137 2
##  $ Engaged.Users                                      : int  178 1457 177 2211 671 1191 481 537 1530 280 ...
##  $ Post.Consumers                                     : int  109 1361 113 790 410 1073 265 232 1407 183 ...
##  $ Post.Consumptions                                  : int  159 1674 154 1119 580 1389 364 305 1692 250 ...
##  $ Impressions.on.people.who.liked.Page               : int  3078 11710 2812 61027 6228 16034 15432 19728 15
##  $ Reach.on.people.who.like.the.Page                  : int  1640 6112 1503 32048 3200 7852 9328 11056 7912 
##  $ People.who.liked.Page.and.engaged.with.post: int  119 1108 132 1386 396 1016 379 422 1250 199 ...
##  $ Comments                                           : int  4 5 0 58 19 1 3 0 0 3 ...
##  $ Likes                                              : int  79 130 66 1572 325 152 249 325 161 113 ...
##  $ Share                                              : int  17 29 14 147 49 33 27 14 31 26 ...
```

```r
#head(tsne_out)
df = data.frame(x=tsne_out$Y[,1], y= tsne_out$Y[,2], type = FB_metric[,2])
df_1 = data.frame(x=tsne_out$Y[,1], y= tsne_out$Y[,2],
                  type = as.factor(FB_metric[,3]))
#the category of the data
df_2 = data.frame(x=tsne_out$Y[,1], y= tsne_out$Y[,2],
                  type = as.factor(FB_metric[,7]))
#is the ad paid or unpaid?
df_3 = data.frame(x=tsne_out$Y[,1], y= tsne_out$Y[,2],
                  type = as.factor(FB_metric[,4]))
#the month of the post
ggplot(data=df, aes(x=x, y=y, group=type, color=type))+geom_point()
```




```r
#the type of the post was used to cluster the data.
#The T-SNE didnt do a good job
#with the clusters
ggplot(data=df_1, aes(x=x, y=y, group=type, color=type))+geom_point()
```


```r
##the category of the post was used to cluster the data.
#The T-SNE didnt do a good job
#with the clusters
ggplot(data=df_2, aes(x=x, y=y, group=type, color=type))+geom_point()
```



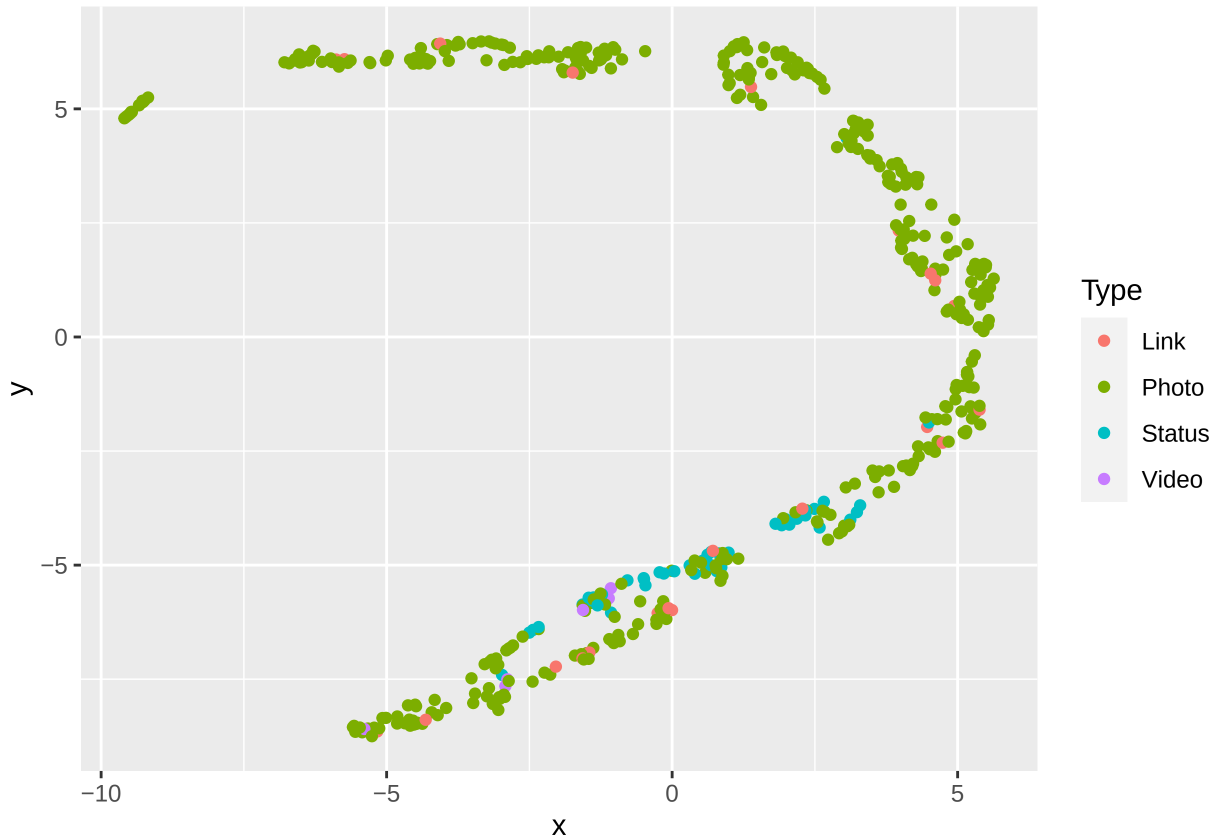

```
#the type of the paid or unpaid option was used to cluster the data.
#The T-SNE didnt do a good job with the clusters
ggplot(data=df_3, aes(x=x, y=y, group=type, color=type))+geom_point()
```


```
#the type of the month was used to cluster the data.
#The T-SNE didnt do a good job with the clusters

#install.packages("umap")

#reducing clustering for the purpose of clustering is
#where the difference between tsne and umap starts to show.
FB_umap = umap(FB_metric_new, n_components = 2,
               n_neighbors = 50, min_dist = 0.01)

head(FB_umap$layout)
```

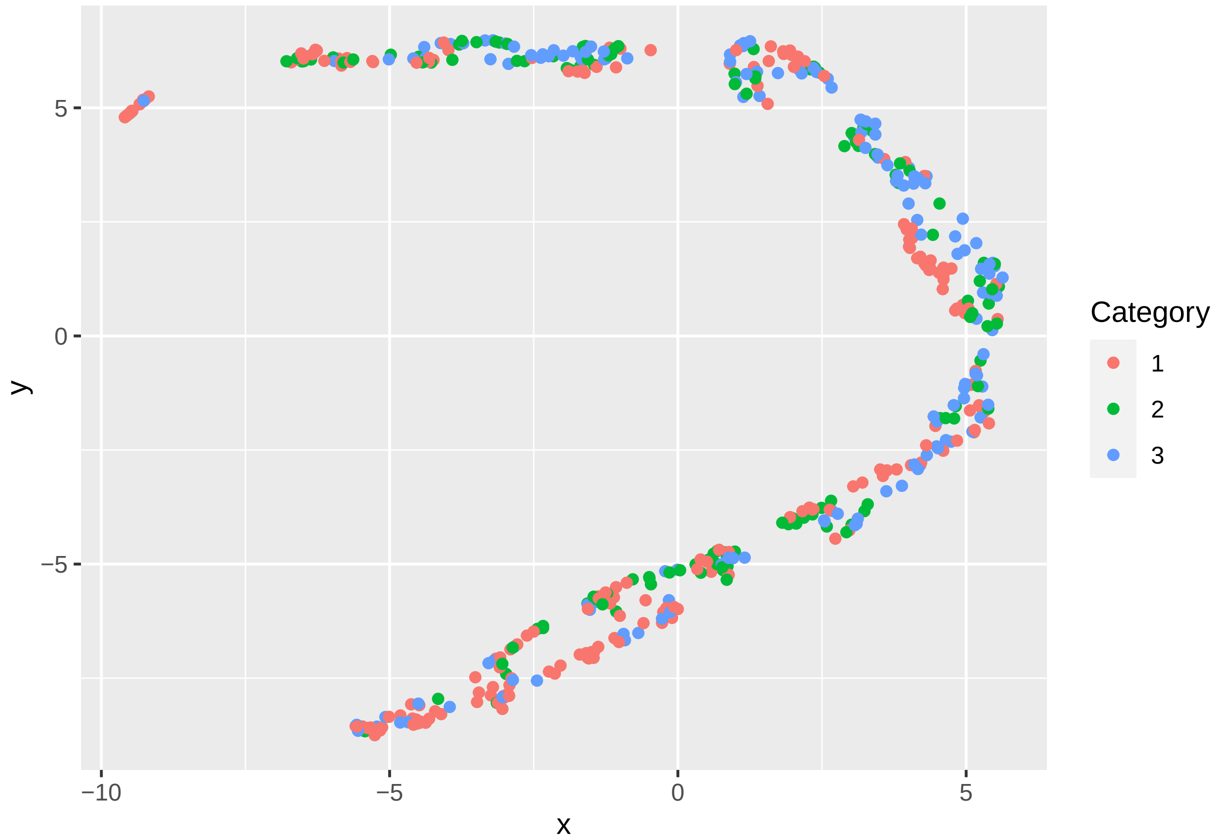
```
##             [,1]      [,2]
## [1,] -1.4951800 -6.657059
## [2,]  1.1502086  4.093892
## [3,] -2.3293904 -6.984326
## [4,] -2.4276861  8.135807
## [5,]  2.2804605  1.570793
## [6,]  0.7312988  4.981033
```

```
fb.umap = umap(FB_metric[,8:18])
fb_umap_df = data.frame(x=fb.umap$layout[,1], y = fb.umap$layout[,2],
                        Type = FB_metric$Type,
                        Category = as.factor(FB_metric$Category),
                        Post.Month = as.factor(FB_metric$Post.Month),
```

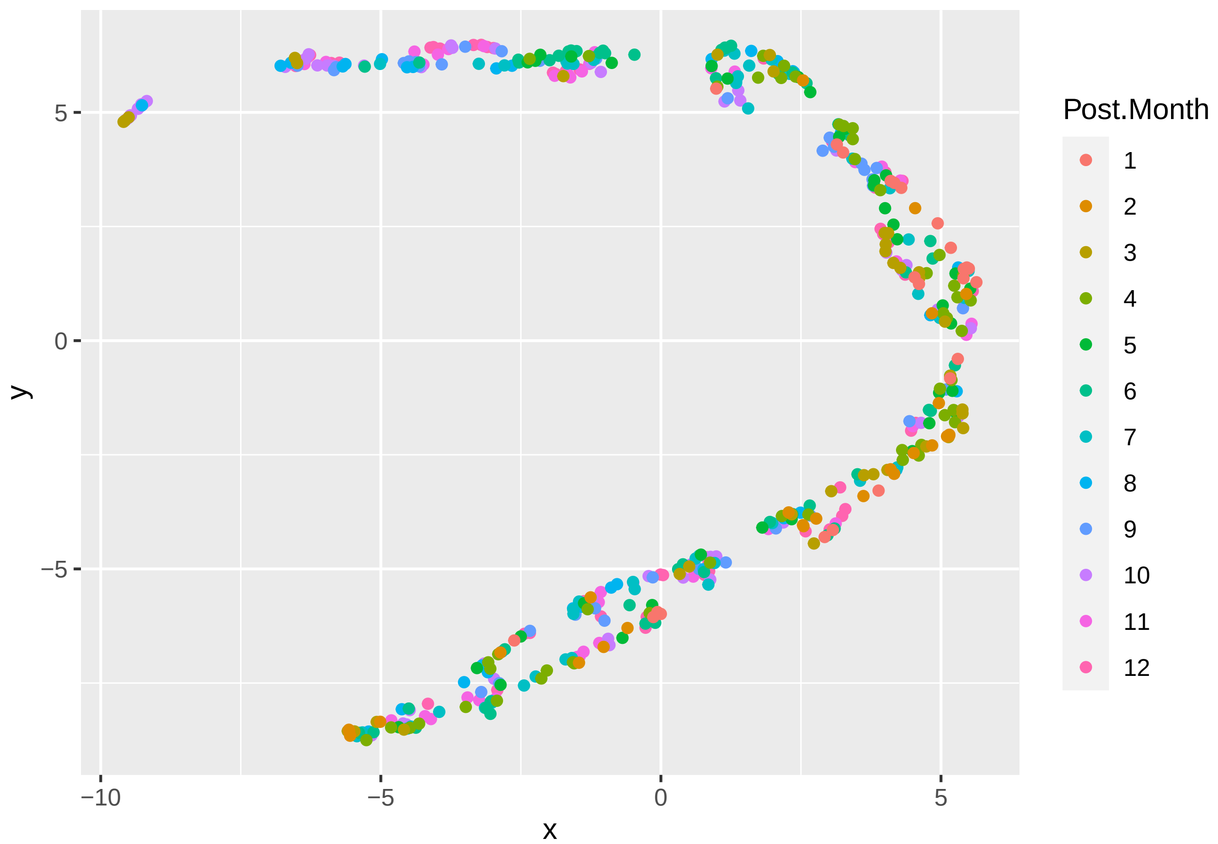
```r
                              Paid = as.factor(FB_metric$Paid))

ggplot(data = fb_umap_df, aes(x=x, y = y, color = Type))+geom_point()
```



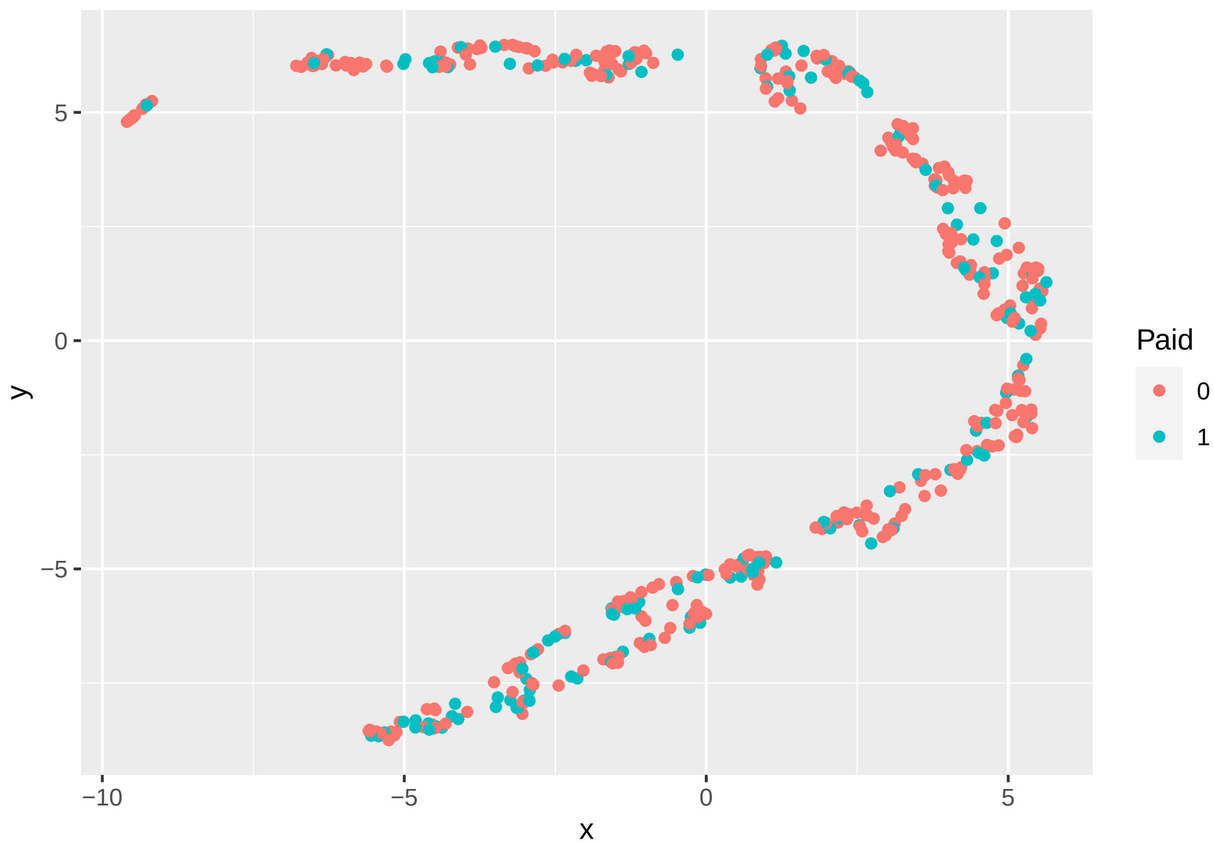

```r
#the UMAP tries to cluster using the type of post. It however does not show good
#clusters for the data. The photo post is shown to dominate the clusters as the
#biggest group.
ggplot(data = fb_umap_df, aes(x=x, y = y, color = Category))+geom_point()
```




```r
#UMAP tries to cluster the data with the post category. There is no clear cluster
#separation and no district clusters are seen.
ggplot(data = fb_umap_df, aes(x=x, y = y, color = Post.Month))+geom_point()
```




```r
#UMAP tries to cluster the data with the post month There is no clear cluster
#separation and no district clusters are seen.
ggplot(data = fb_umap_df, aes(x=x, y = y, color = Paid))+geom_point()
```




```
#UMAP tries to cluster the data with whether they are paid or not.  There is no clear cluster
#separation and no district clusters are seen.

#the UMAP parameter optimization as done to achieve better results. There was
#however not very obvious results.
#the UMAP shows a better representation of clusters than Tsne. Even though the
#clusters were not very clear, the structure looked better with the UMAP.
#Parameter optimization was done for TSNE and UMAP, but the results still dont
#show very good trends and results.
```