# Dimensionality Reduction and Clustering Techniques

MEENOWA Sarvesh

17/11/2021

Import important libraries

```
library('car')
library("readr")
library("ggpubr")
library("corrplot")
library("FactoMineR")
library("vcd")
library("moments")
library("ggplot2")
library("FactoMineR")
library("factoextra")
library("tidyverse")
library("ClustOfVar")
library("ClusterR")
library("cluster")
library("RColorBrewer")
library("scales")
library("ggdendro")
library("NbClust")
library("igraph")
library("ggraph")
library("GGally")
library("plotly")
library("PerformanceAnalytics")
library("xtable")
```

```
#import dataset
df = read_csv("H:/Downloads/Datatsets/users.db.csv")
head(df)
```

```
## # A tibble: 6 x 16
##   userid date.crea  score n.matches n.updates.photo n.photos last.connex
##    <dbl> <date>     <dbl>     <dbl>           <dbl>    <dbl> <date>
## 1      1 2011-09-17  1.50        11               5        6 2011-10-07
## 2      2 2017-01-17  8.95        56               2        6 2017-01-31
## 3      3 2019-05-14  2.50        13               3        4 2019-06-17
## 4      4 2015-11-27  2.82        32               5        2 2016-01-15
## 5      5 2014-11-28  2.12        21               1        4 2015-01-15
## 6      6 2017-06-05  1.70        14               2        6 2017-07-03
## # ... with 9 more variables: last.up.photo <date>, last.pr.update <lgl>,
```

```
## #   gender <dbl>, sent.ana <dbl>, length.prof <dbl>, voyage <dbl>, laugh <dbl>,
## #   photo.keke <dbl>, photo.beach <dbl>
```

```
colnames(df)
```

```
##  [1] "userid"          "date.crea"       "score"           "n.matches"
##  [5] "n.updates.photo" "n.photos"        "last.connex"     "last.up.photo"
##  [9] "last.pr.update"  "gender"          "sent.ana"        "length.prof"
## [13] "voyage"          "laugh"           "photo.keke"      "photo.beach"
```

```
#calculate log of variables and add them to df
df$score_log <- log10(df$score)
df$n.matches_log <- log10(df$n.matches)
df$n.photos_log <- log10(df$n.photos)
```

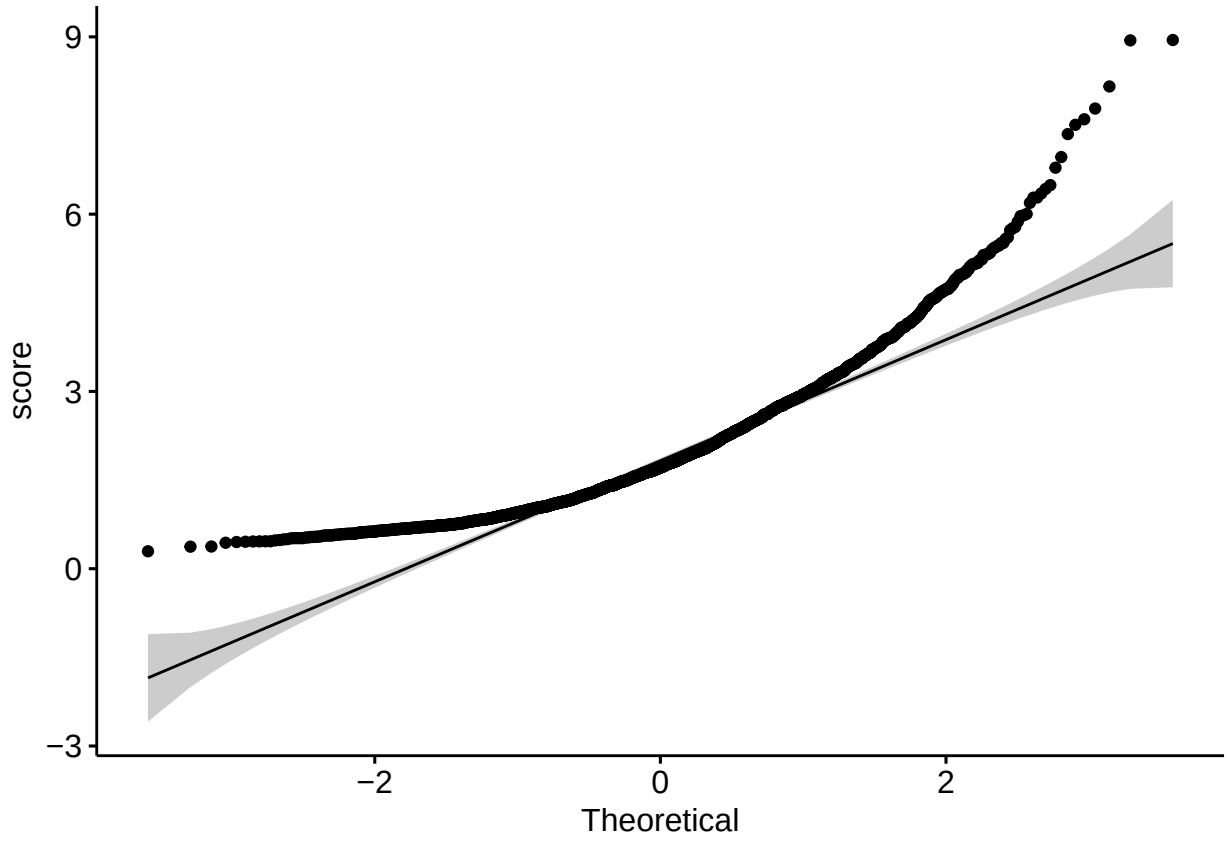
```
#create continuous dataset by subsetting continuous variables
df_cont <- df[c('score', 'n.matches', 'n.updates.photo', 'n.photos')]
head(df_cont)
```

```
## # A tibble: 6 x 4
##   score n.matches n.updates.photo n.photos
##   <dbl>     <dbl>           <dbl>    <dbl>
## 1  1.50        11               5        6
## 2  8.95        56               2        6
## 3  2.50        13               3        4
## 4  2.82        32               5        2
## 5  2.12        21               1        4
## 6  1.70        14               2        6
```

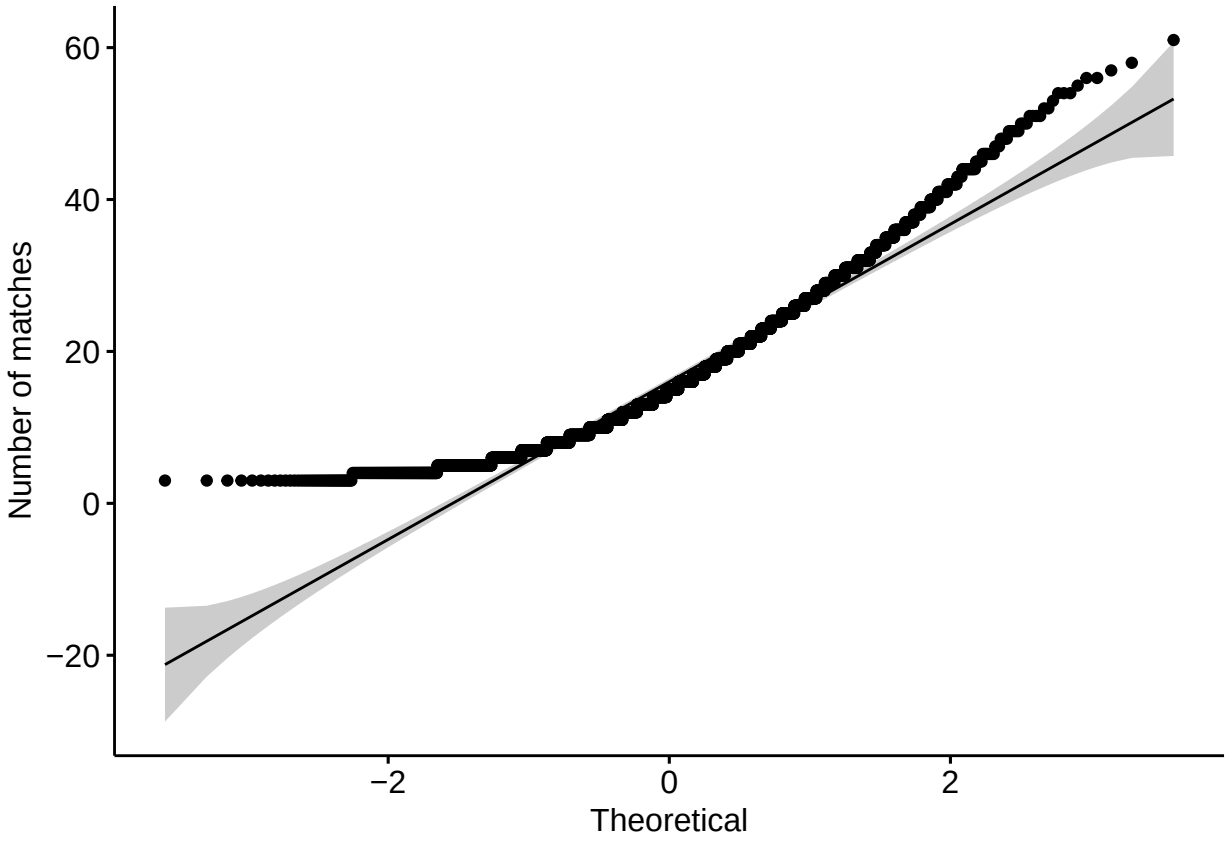
## Identifying correlations

Search for correlations among (cor.test). Use both parametric and non-parametric techniques (Pearson vs. Spearman for correlation between continuous variables). Design a couple of plots featuring correlated variables

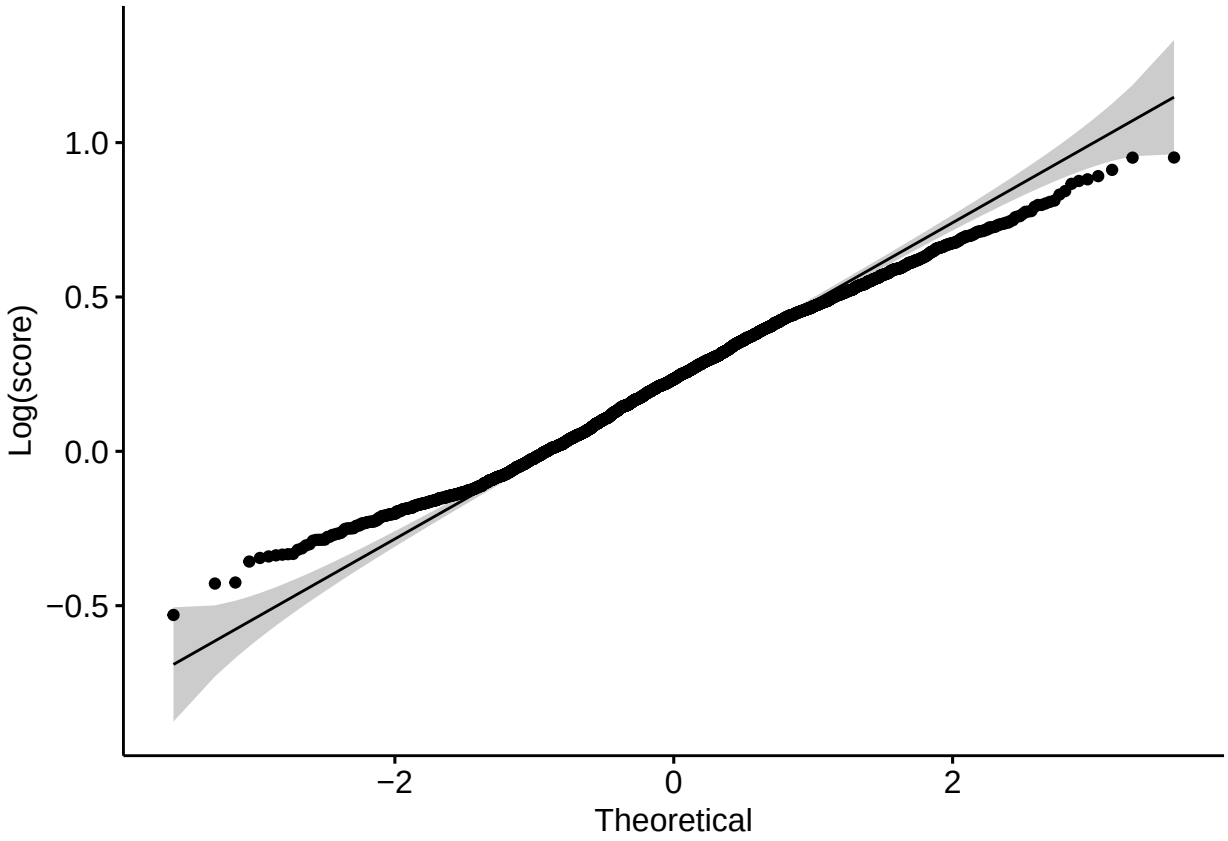
```
# check qqplot for score
ggqqplot(df$score, ylab = "score")
```



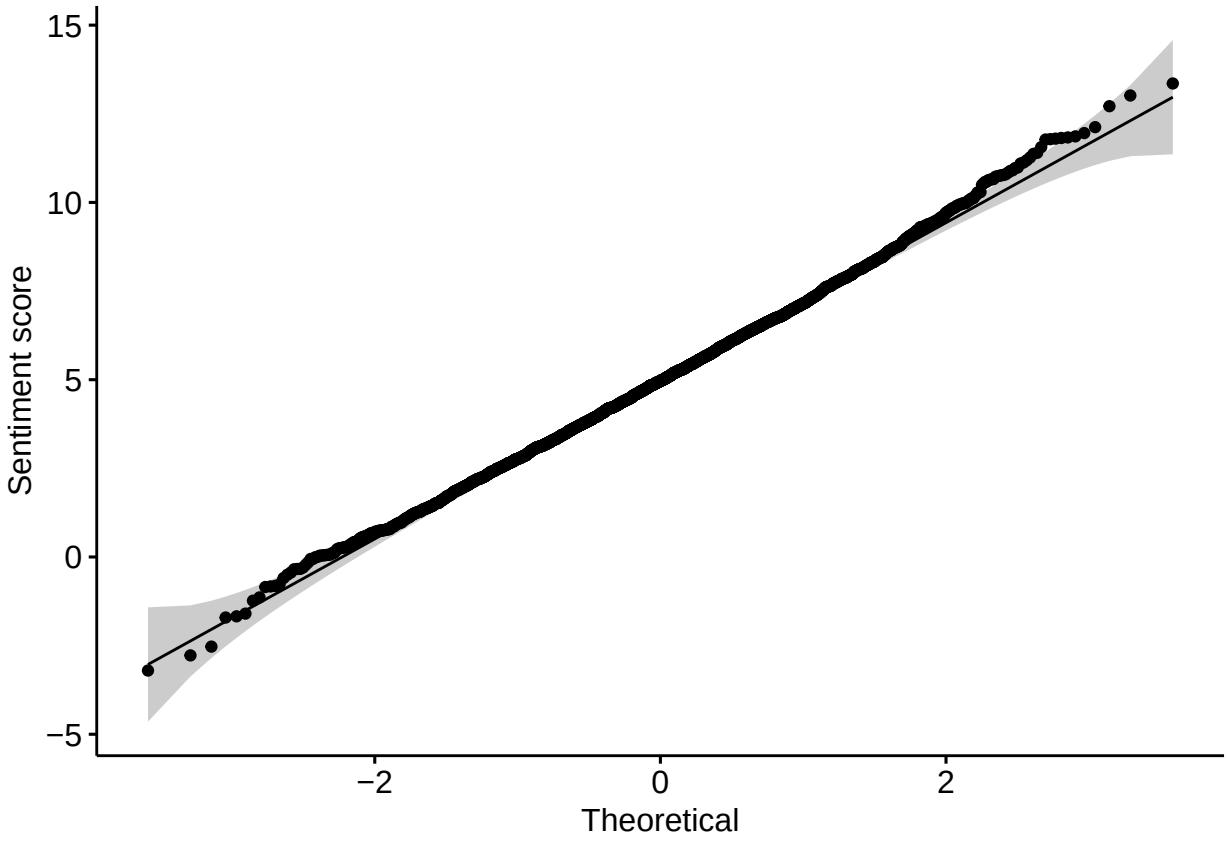

```
# check qqplot for n.matches
ggqqplot(df$n.matches, ylab = "Number of matches")
```




```
#check normal distribution of score
ggqqplot(df$score_log, ylab = "Log(score)")
```



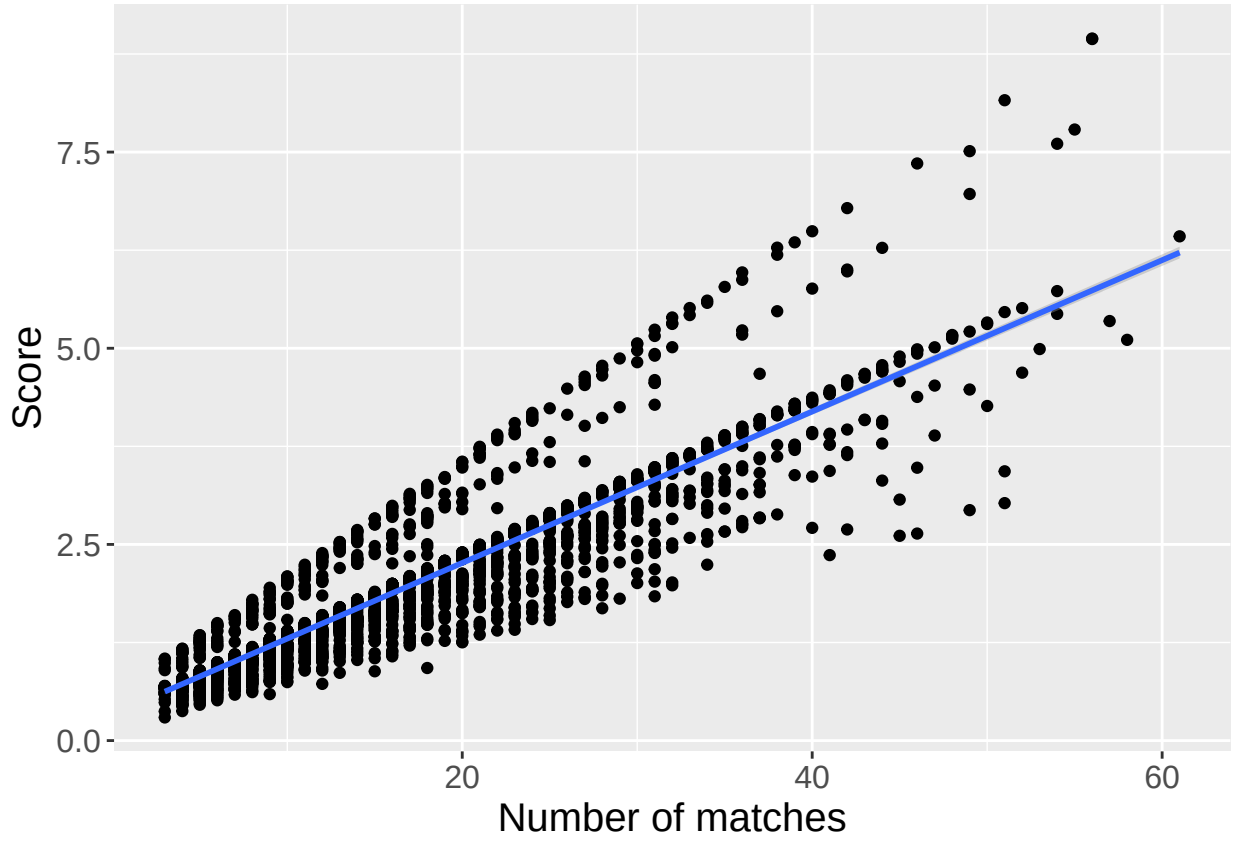

```
 #check normal distribution of sentiment score
ggqqplot(df$sent.ana, ylab = "Sentiment score")
```




```
#spearman correlation , non-parametric
res <- cor.test(df$score, df$n.matches, method = "spearman")
res
```

```
##
##  Spearman's rank correlation rho
##
## data:  df$score and df$n.matches
## S = 338158884, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.9248536
```

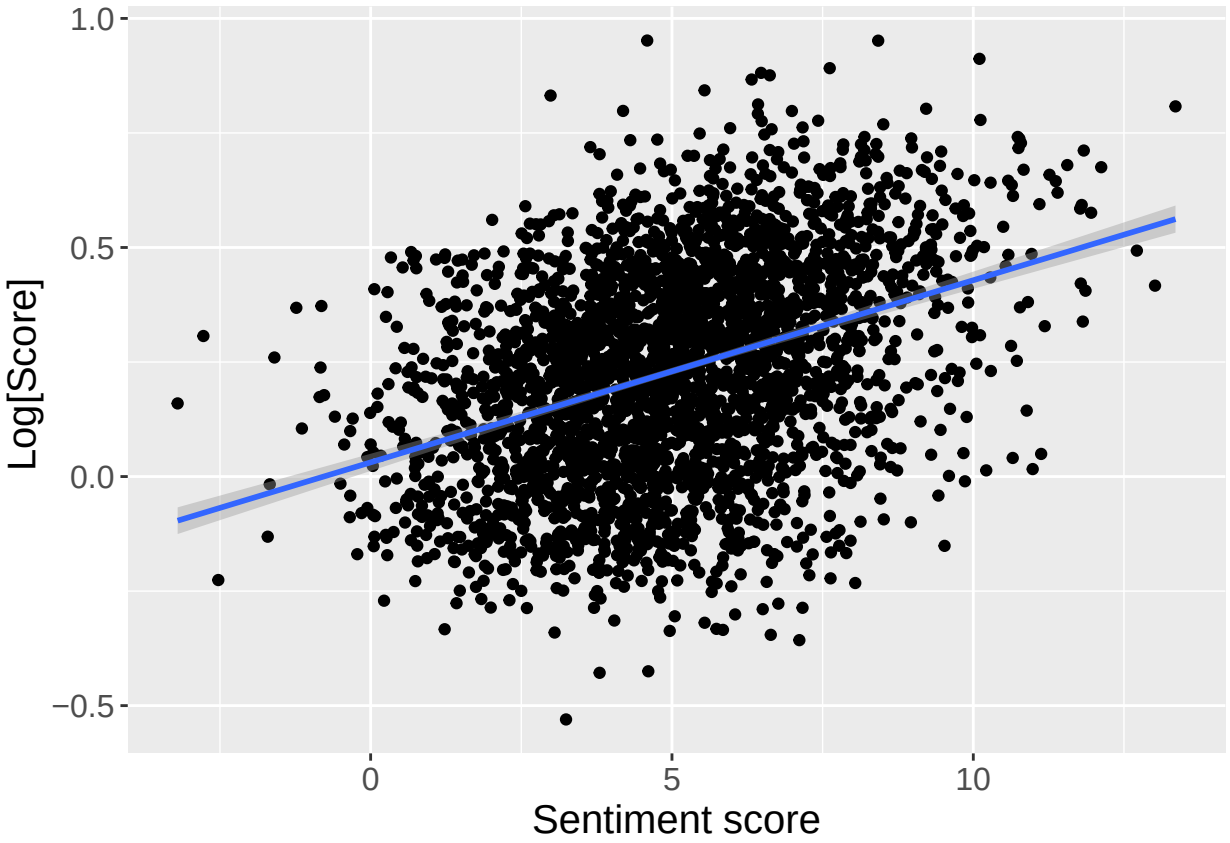
```
ggplot(df, aes(x=n.matches, y=score)) +
  geom_point()+
  geom_smooth(method=lm) + ylab("Score") + xlab("Number of matches") +theme(text = element_text(size =
```




```
#pearson correlation , parametric
res2 <- cor.test(df$score_log, df$sent.ana, method = "pearson")
res2
```

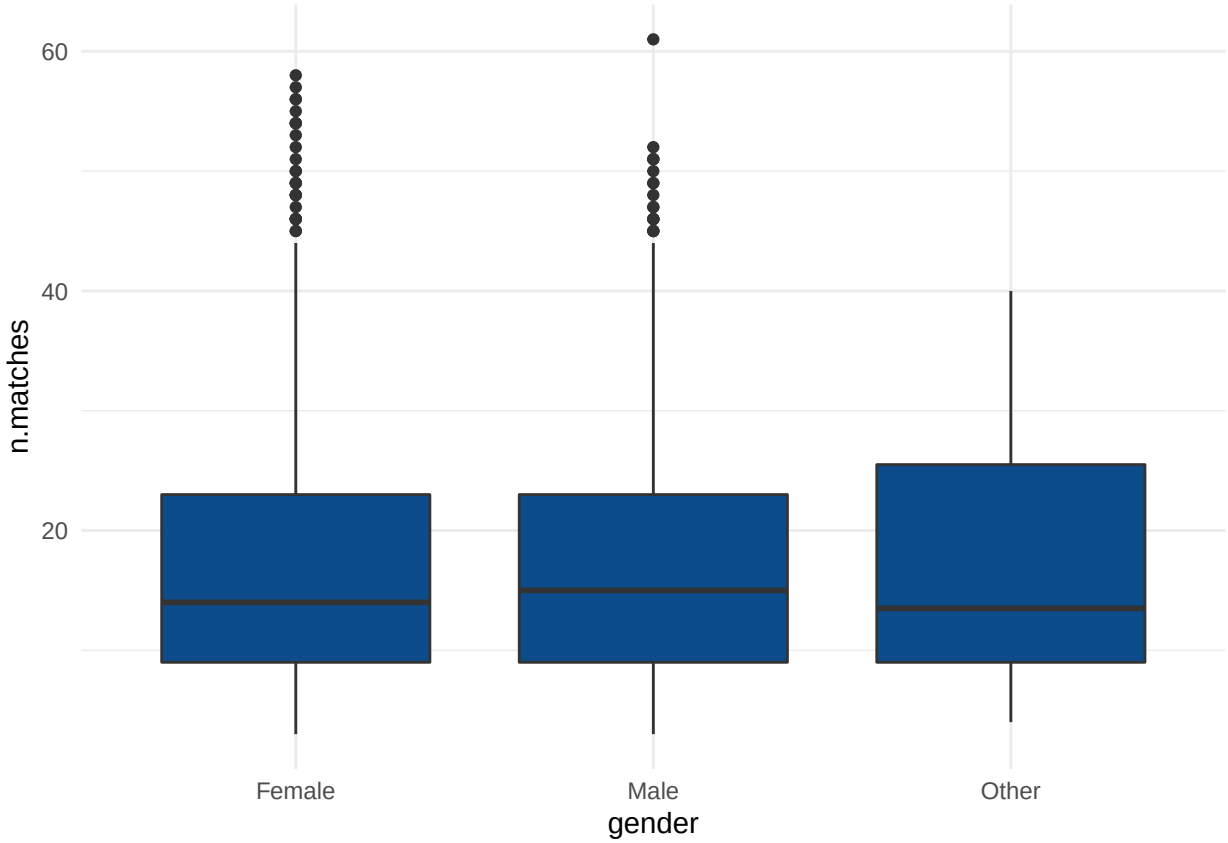
```
##
##  Pearson's product-moment correlation
##
## data:  df$score_log and df$sent.ana
## t = 22.748, df = 2998, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3527176 0.4137669
## sample estimates:
##       cor
## 0.3836613
```

```
ggplot(df, aes(x=sent.ana, y=score_log)) +
  geom_point()+
  geom_smooth(method=lm) + ylab(" Log[Score]") + xlab("Sentiment score") +theme(text = element_text(siz
```




```r
#recode gender from integers to categories
df$gender[df$gender == 0] <- "Male"
df$gender[df$gender == 1] <- "Female"
df$gender[df$gender == 2] <- "Other"
```

```r
#boxplot for gender and number of matches
ggplot(df) +
  aes(x = gender, y = n.matches) +
  geom_boxplot(fill = "#0c4c8a") +
  theme_minimal()
```




```
#check if it follows normal distribution or not for n.matches ,males
shapiro.test(subset(df, gender == "Male")$n.matches)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  subset(df, gender == "Male")$n.matches
## W = 0.93526, p-value < 2.2e-16
```

```
#check if it follows normal distribution or not for n.matches ,females
shapiro.test(subset(df, gender == "Female")$n.matches)
```

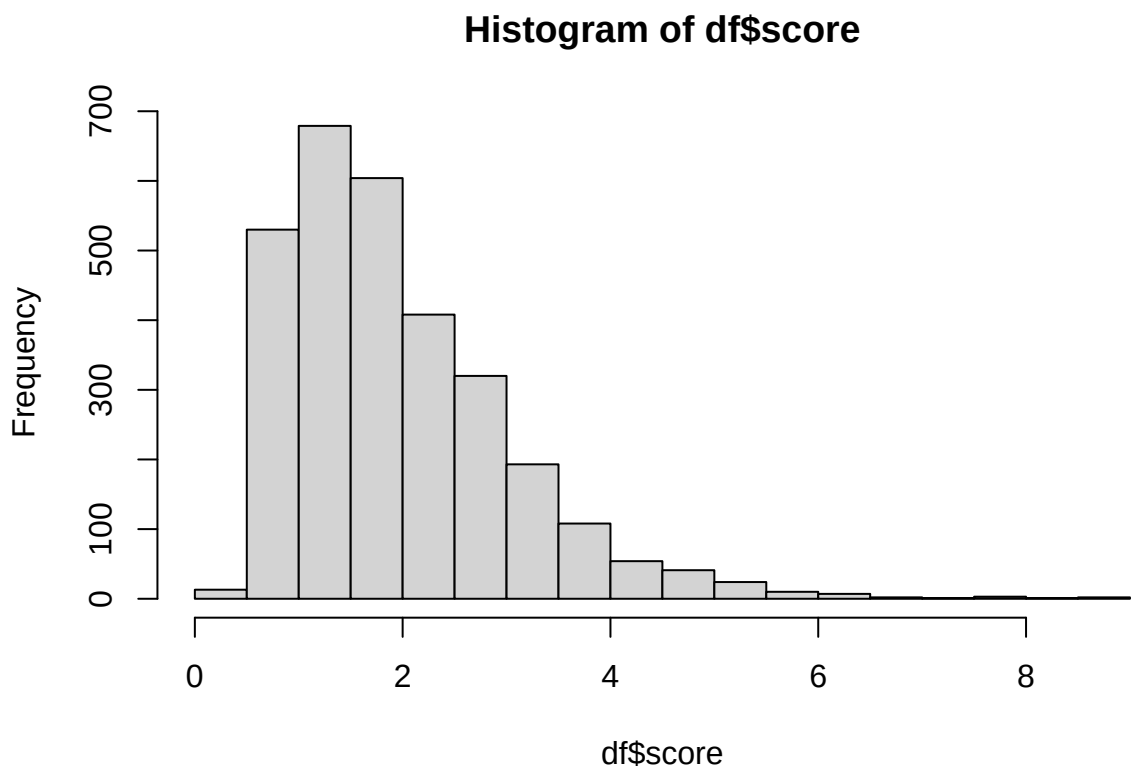
```
##
##  Shapiro-Wilk normality test
##
## data:  subset(df, gender == "Female")$n.matches
## W = 0.91965, p-value < 2.2e-16
```

```
#wilcoxon test - non parametric test
test <- wilcox.test(subset(df, gender != "Other")$n.matches ~ subset(df, gender != "Other")$gender)
print(test)
```

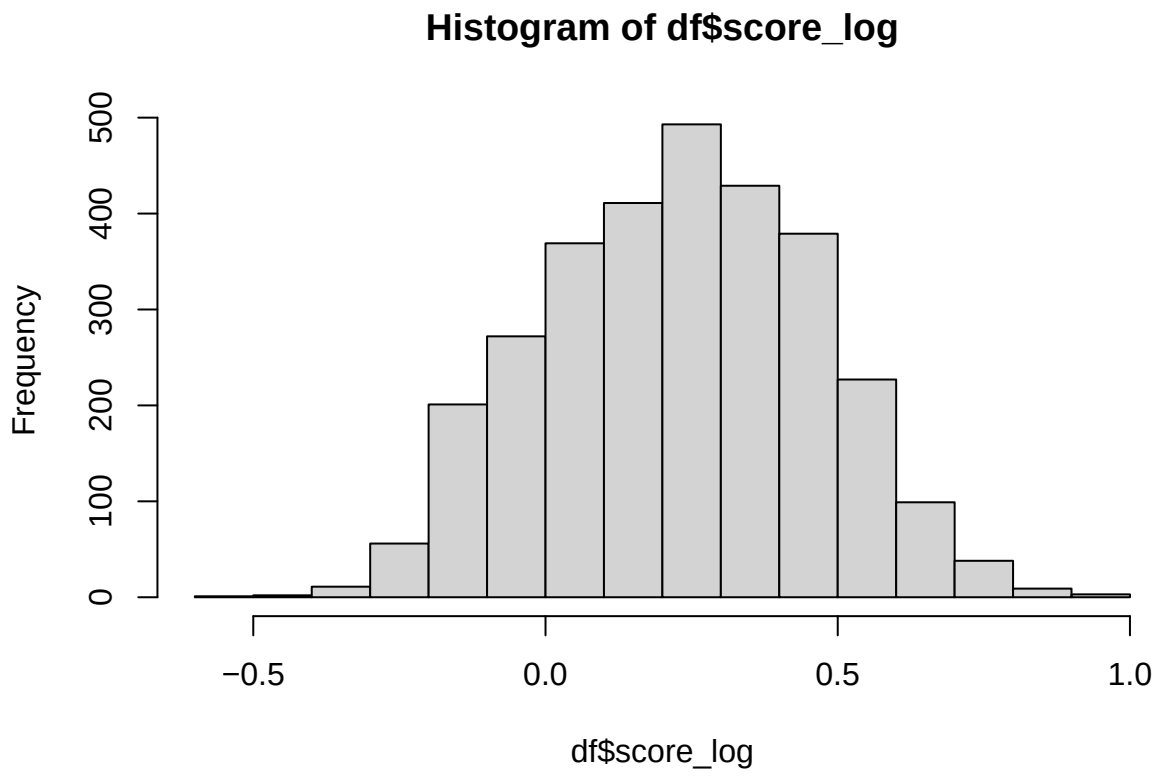
```
##
##  Wilcoxon rank sum test with continuity correction
```

```
##
## data:  subset(df, gender != "Other")$n.matches by subset(df, gender != "Other")$gender
## W = 1061836, p-value = 0.3993
## alternative hypothesis: true location shift is not equal to 0
```

```
#distribution of score
hist(df$score)
```




```
#distribution of log of score
hist(df$score_log)
```




```
#linear model : log score ~ gender, n.matches_log, n_photos_log
lm(score_log ~ gender+n.matches_log+n.photos_log, df)
```

```
##
## Call:
## lm(formula = score_log ~ gender + n.matches_log + n.photos_log,
##     data = df)
##
## Coefficients:
##   (Intercept)     genderMale    genderOther  n.matches_log   n.photos_log
##     -0.608137      -0.082629      -0.019904       0.767180       0.006908
```

```
#linear model : log score ~  n.matches_log, n_photos_log, gender*photo.keke , gender*photo_beach
lm(score_log ~ gender+n.matches_log+n.photos_log+gender*photo.keke+gender*photo.beach, df)
```

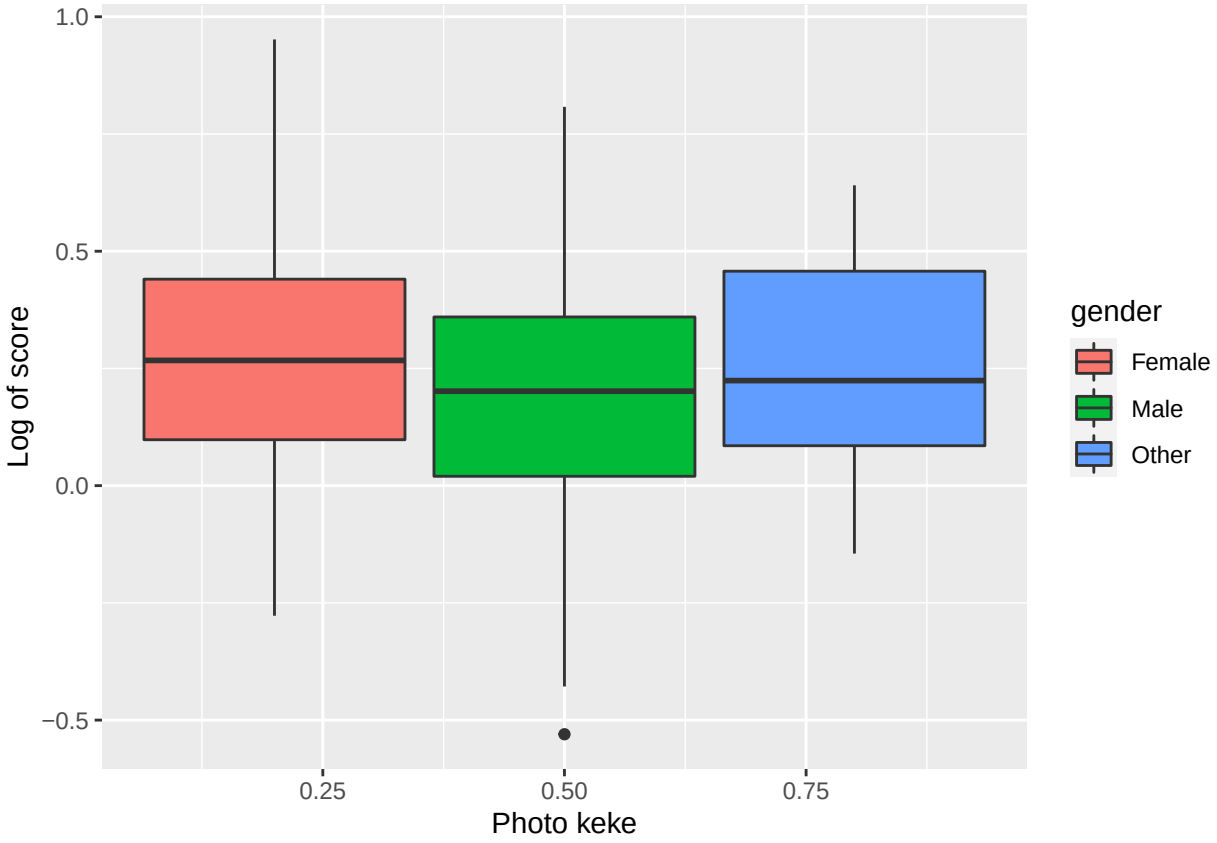
```
##
## Call:
## lm(formula = score_log ~ gender + n.matches_log + n.photos_log +
##     gender * photo.keke + gender * photo.beach, data = df)
##
## Coefficients:
##             (Intercept)               genderMale               genderOther
##                -0.651616                 0.001231                 0.022979
##           n.matches_log            n.photos_log              photo.keke
##                 0.771261                 0.001909                -0.019166
```

```
##             photo.beach    genderMale:photo.keke   genderOther:photo.keke
##                0.178552               -0.136739                0.023357
##  genderMale:photo.beach  genderOther:photo.beach
##               -0.275534               -0.188158
```

```
#linear model : log score ~  n.matches_log, n_photos_log, gender*photo.keke , gender*photo_beach
lm(score_log ~ gender*photo.keke+gender*photo.beach, df)
```

```
##
## Call:
## lm(formula = score_log ~ gender * photo.keke + gender * photo.beach,
##     data = df)
##
## Coefficients:
##             (Intercept)               genderMale               genderOther
##                0.229136                 0.004068                 0.038024
##              photo.keke              photo.beach    genderMale:photo.keke
##               -0.011413                 0.167943                -0.137434
##  genderOther:photo.keke   genderMale:photo.beach  genderOther:photo.beach
##                0.103481                -0.270049                -0.261643
```

```
ggplot(df, aes(x=photo.keke, y=score_log, fill=gender)) + geom_boxplot() + ylab("Log of score") +xlab("
```



```r
#anova with dependent as gender*photo.keke
anova(lm(score_log ~ gender*photo.keke, df))
```

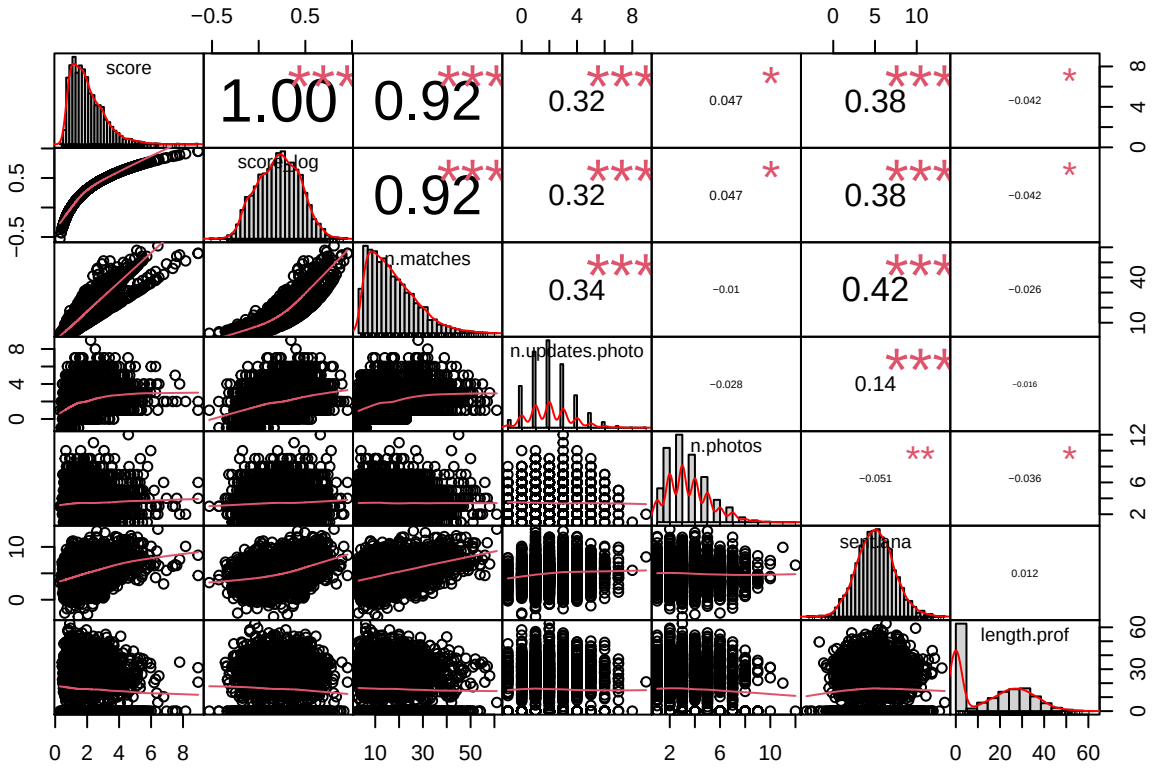
```
## Analysis of Variance Table
##
## Response: score_log
##                     Df  Sum Sq Mean Sq F value    Pr(>F)
## gender               2   4.475  2.2373  44.248 < 2.2e-16 ***
## photo.keke           1   3.860  3.8602  76.346 < 2.2e-16 ***
## gender:photo.keke    2   1.628  0.8138  16.096 1.114e-07 ***
## Residuals         2994 151.384  0.0506
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#summary table for linear model
summary(lm(score_log ~ gender*photo.keke, df))
```

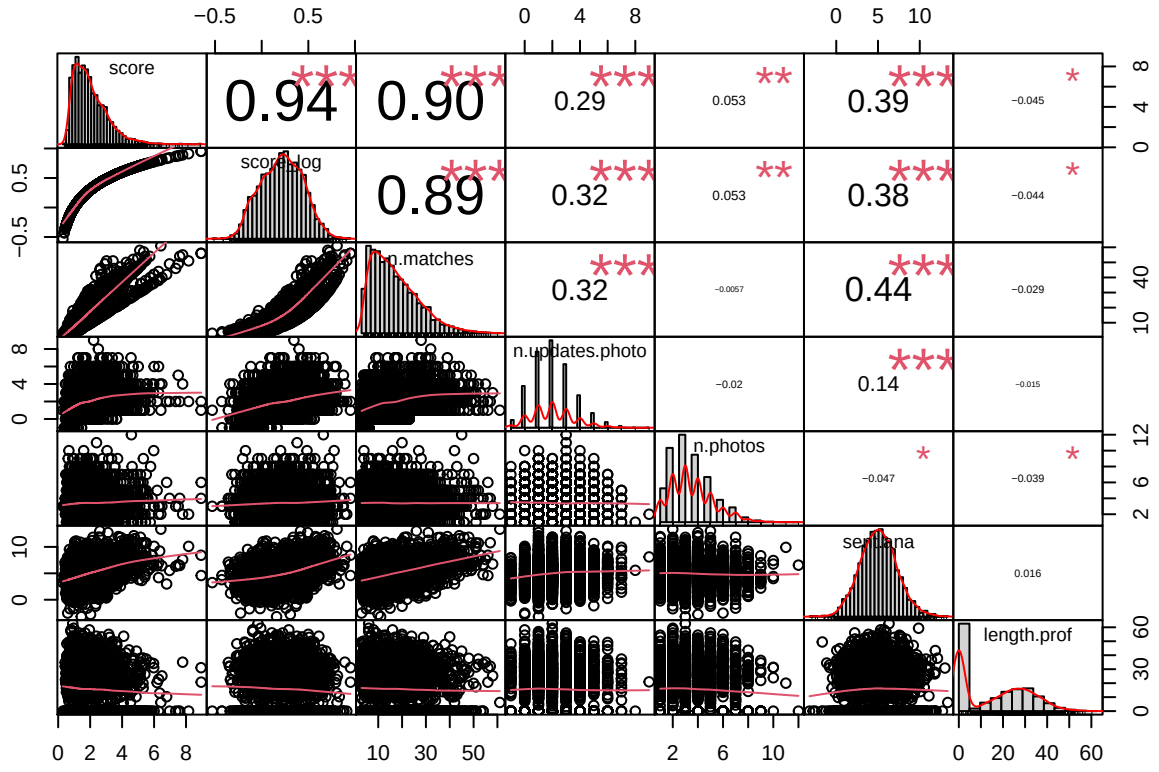
```
##
## Call:
## lm(formula = score_log ~ gender * photo.keke, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.60659 -0.17135  0.00359  0.16978  0.68068
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             0.270990   0.006285  43.117  < 2e-16 ***
## genderMale             -0.048548   0.009063  -5.357 9.11e-08 ***
## genderOther            -0.026813   0.031520  -0.851    0.395
## photo.keke             -0.018370   0.019179  -0.958    0.338
## genderMale:photo.keke  -0.127643   0.023822  -5.358 9.04e-08 ***
## genderOther:photo.keke  0.114682   0.106931   1.072    0.284
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2249 on 2994 degrees of freedom
## Multiple R-squared:  0.06175,    Adjusted R-squared:  0.06018
## F-statistic: 39.41 on 5 and 2994 DF,  p-value: < 2.2e-16
```

```r
#create df with numerical variables only
cont_df <- df[c('score', 'score_log', 'n.matches', 'n.updates.photo', 'n.photos', 'sent.ana', 'length.p
```

```r
chart.Correlation(cont_df, histogram=TRUE, pch=19,method="spearman")
```




```
chart.Correlation(cont_df, histogram=TRUE, pch=19,method="pearson")
```




## Perform a PCA on relevant numerical variables of the dataset

```
#perform pca , add scale = TRUE to scale data

cont_df.pca <- prcomp(cont_df, center = TRUE,scale. = TRUE)
summary(cont_df.pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4    PC5     PC6     PC7
## Standard deviation     1.7902 1.0286 0.9825 0.9354 0.8528 0.34105 0.23098
## Proportion of Variance 0.4578 0.1512 0.1379 0.1250 0.1039 0.01662 0.00762
## Cumulative Proportion  0.4578 0.6090 0.7469 0.8719 0.9758 0.99238 1.00000
```

## Table describing the loadings of each Principal Component.

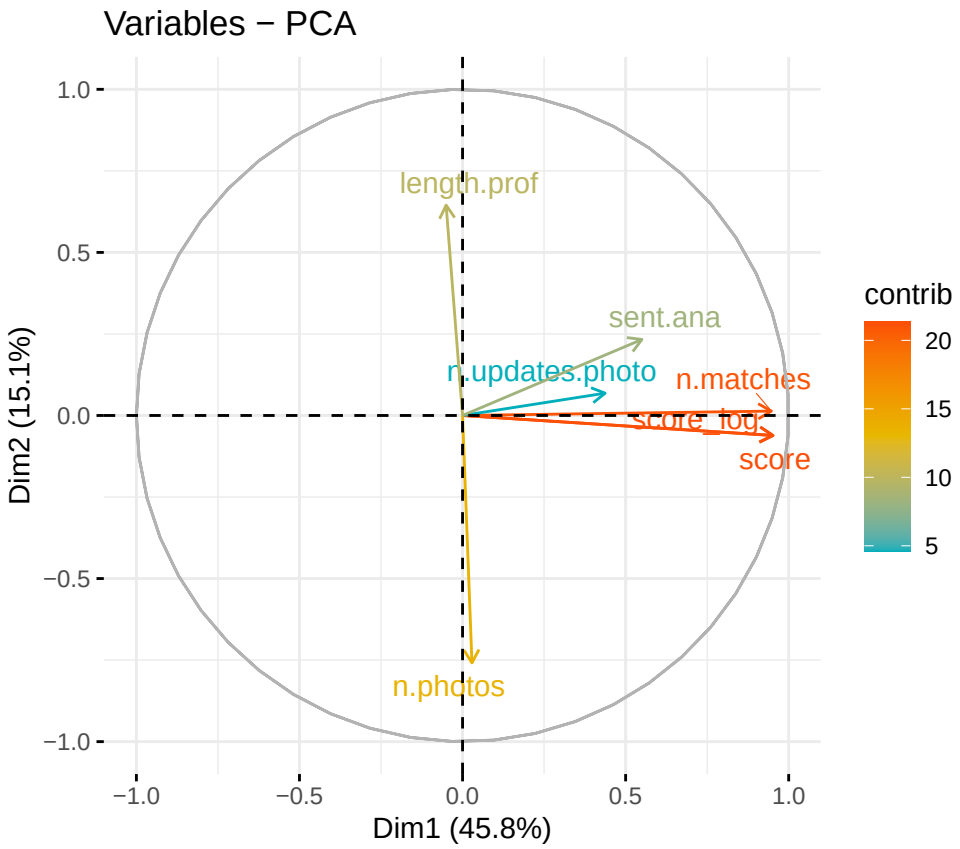
```
#get loadings
cont_df.pca$rotation
```

```
##                       PC1         PC2          PC3         PC4        PC5
## score           0.53191399 -0.05981388  0.046265887 -0.07288421  0.2423494
## score_log       0.53126396 -0.05923331  0.041761002 -0.03338005  0.2333011
```

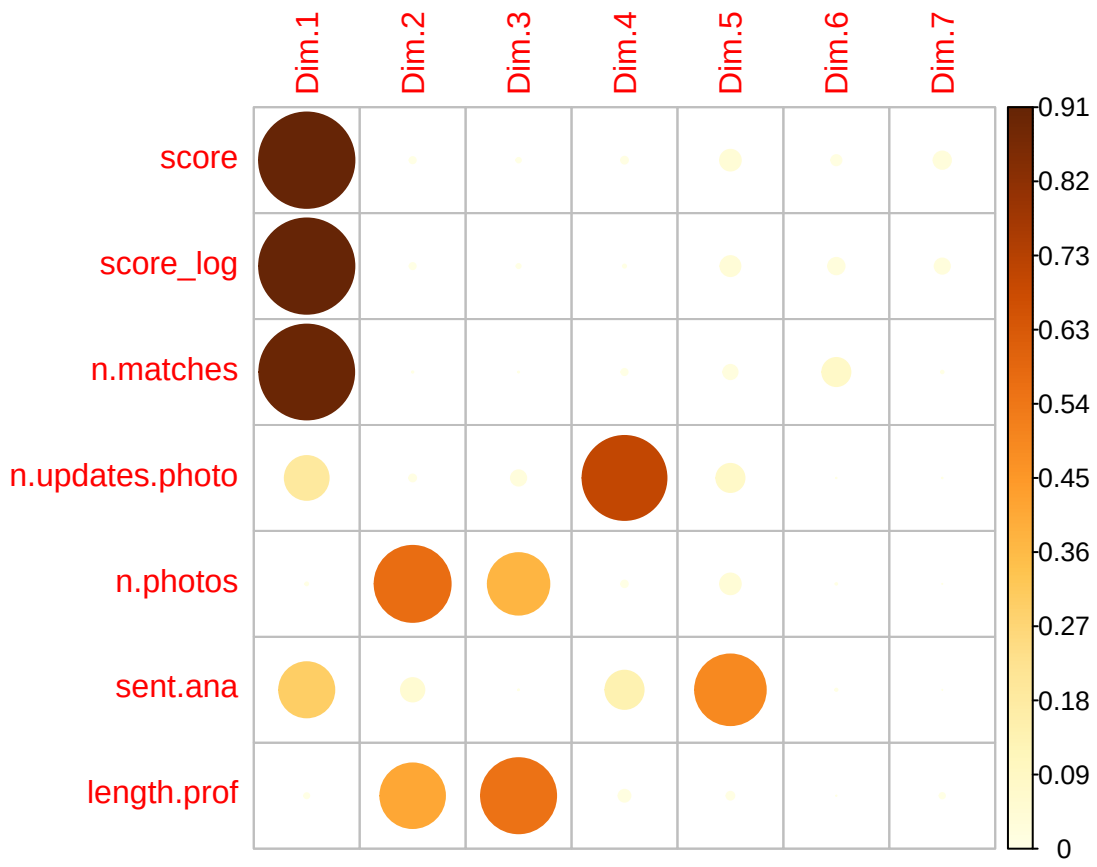
```
## n.matches         0.52891243  0.01308196  0.009993998 -0.06575461  0.1665249
## n.updates.photo   0.24418765  0.06647992 -0.154677673  0.89591953 -0.3290645
## n.photos          0.01607191 -0.73750887  0.624649818  0.07035009 -0.2416298
## sent.ana          0.30724643  0.22660163 -0.009410928 -0.40707989 -0.8271063
## length.prof      -0.02813308  0.62694128  0.762767633  0.12615476  0.0912881
##                          PC6          PC7
## score             0.295946091  0.748144264
## score_log         0.479452086 -0.653519764
## n.matches        -0.821977360 -0.110813538
## n.updates.photo   0.013975612  0.029615721
## n.photos         -0.047703560 -0.005023340
## sent.ana          0.065861455  0.002470454
## length.prof       0.009281706  0.002002602
```

### A variable circle of correlations

```
#supplementary variable in blue, and color indicator : "contribution"
fviz_pca_var(cont_df.pca, col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)
```

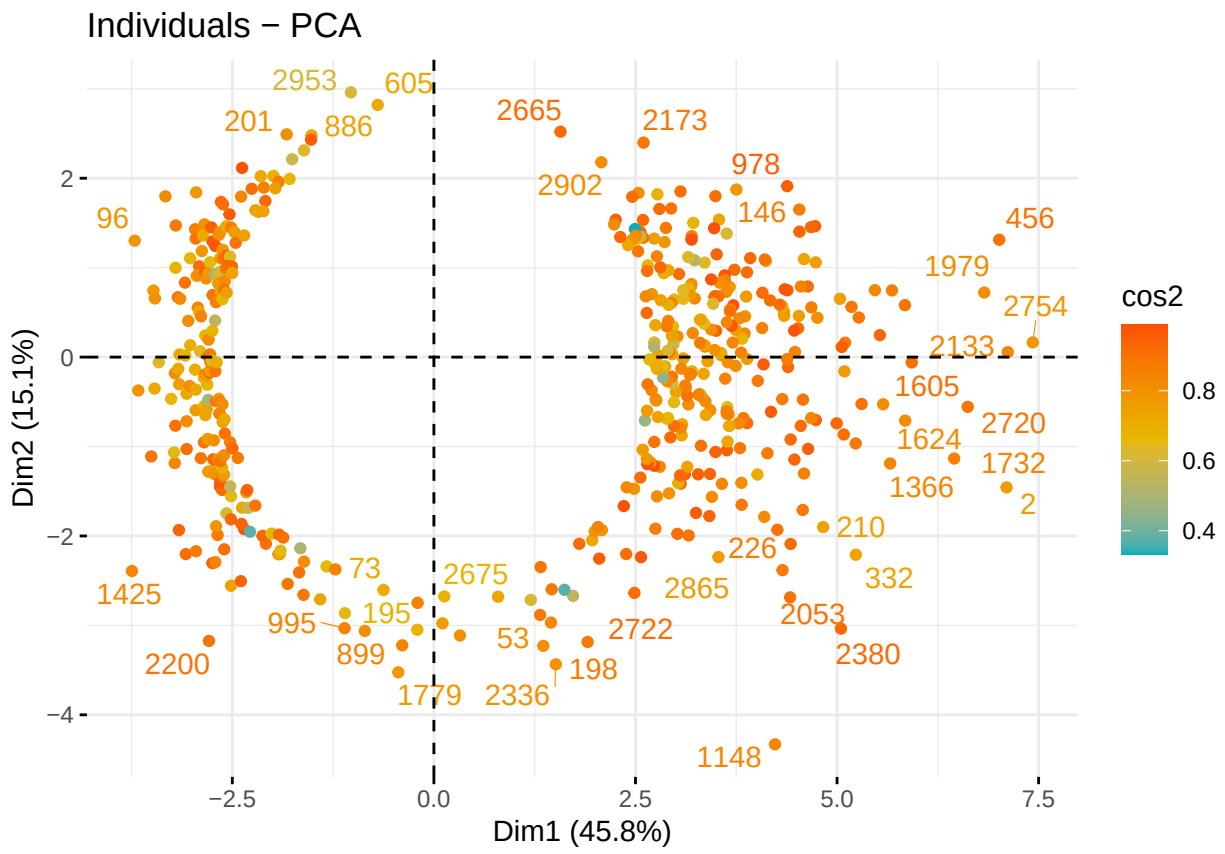



```
#get corrplot with cos2
var <- get_pca_var(cont_df.pca)
corrplot(var$cos2, is.corr = FALSE)
```

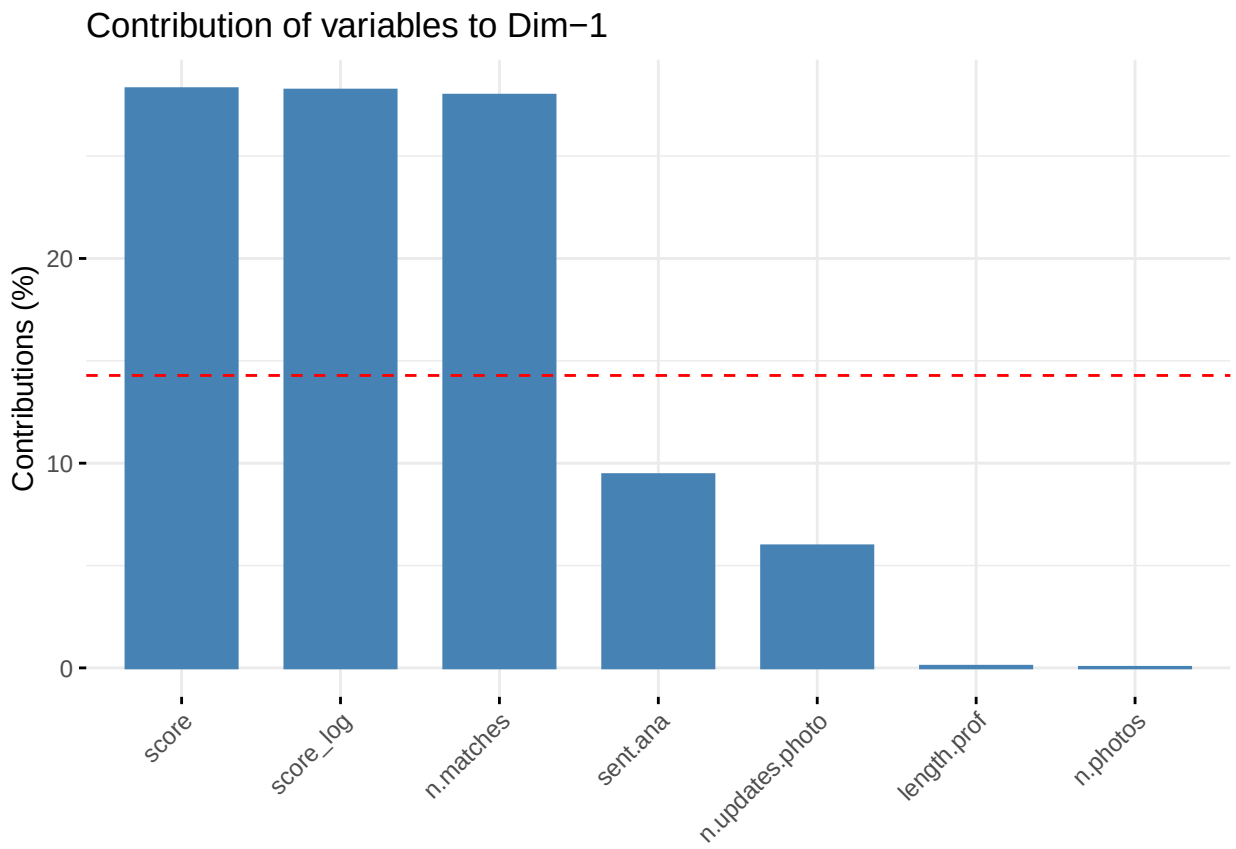



## An individual map with a sample of individuals.

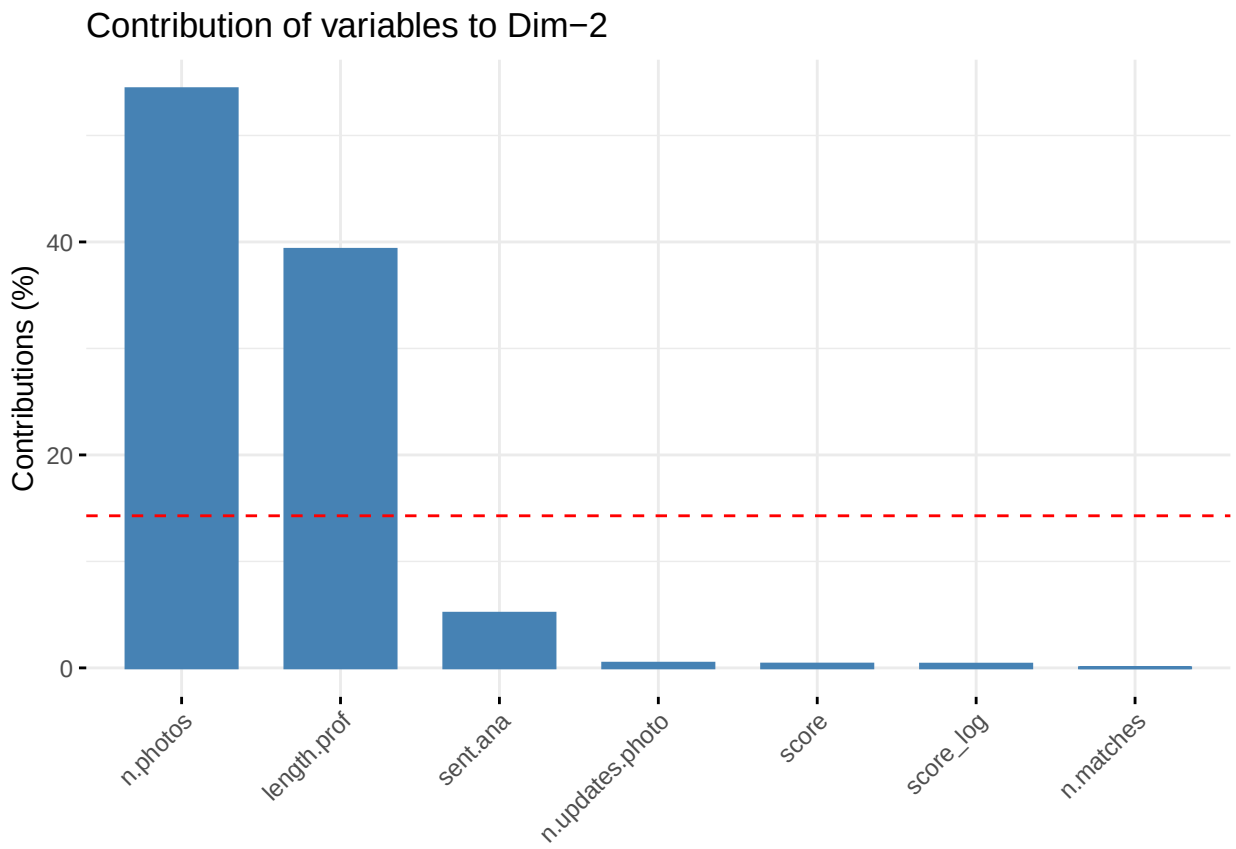
```
#individual graph with sample of 300 individuals
fviz_pca_ind(cont_df.pca, col.ind = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE ,# Avoid text overlapping (slow if many points),
             select.ind = list(contrib = 500)
)
```




```
# Highest contributing variables to dim 1
fviz_contrib(cont_df.pca, choice = "var", axes = 1, top = 10)
```

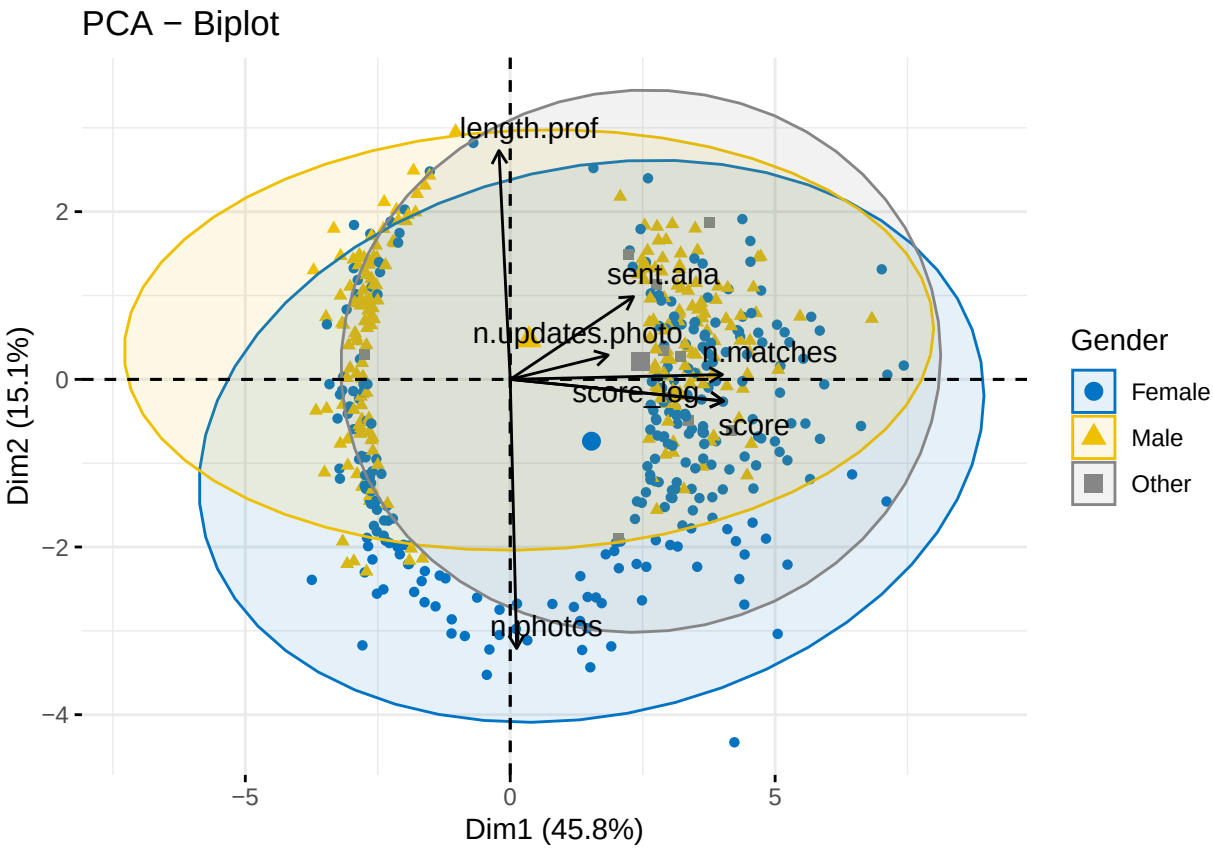



```
# Highest contributing variables to dim 2
fviz_contrib(cont_df.pca, choice = "var", axes = 2, top = 10)
```



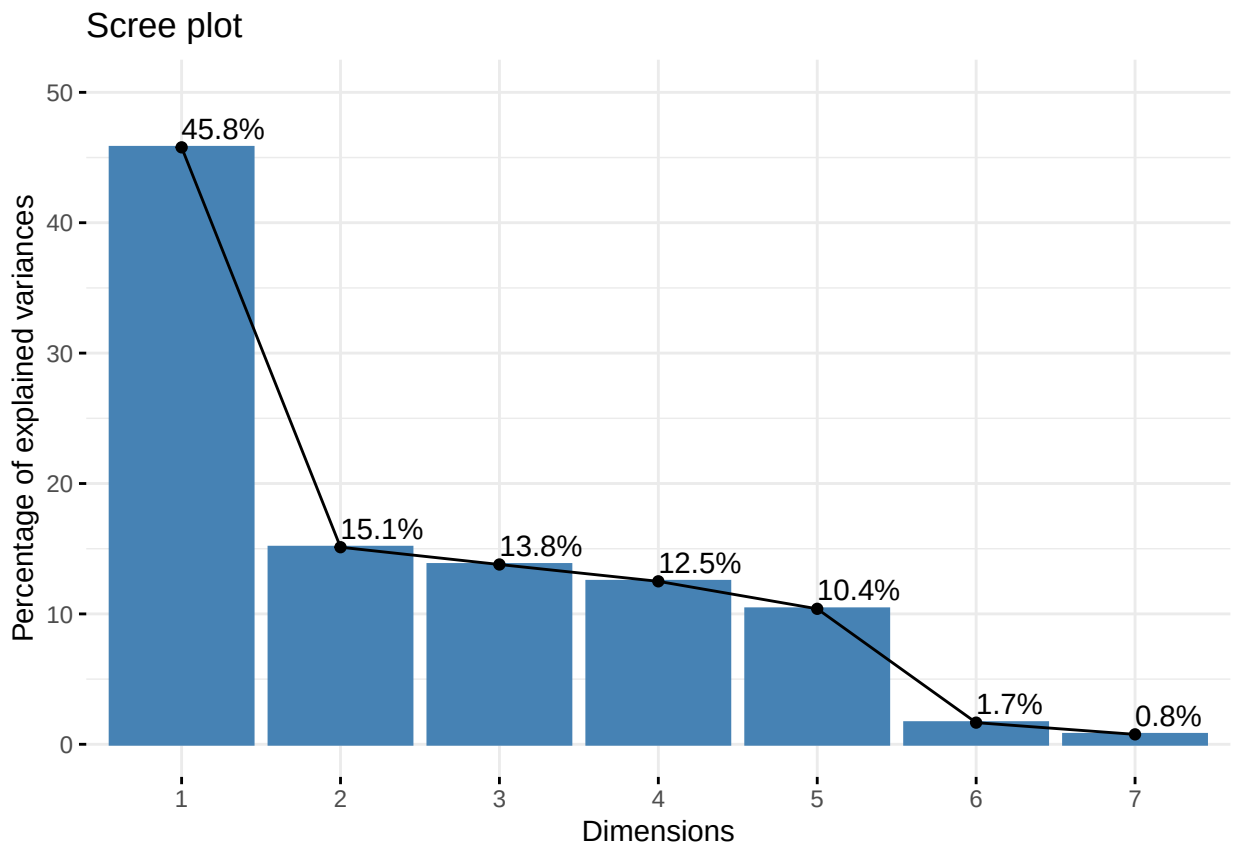

## Biplot with a limited number of individuals

```
# biplot with top 500 contributing individuals with elipses
fviz_pca_biplot(cont_df.pca,
                col.ind = df$gender, palette = "jco",
                addEllipses =T, label = "var",
                col.var = "black", repel = TRUE,
                legend.title = "Gender",
                select.ind = list(contrib = 500))
```

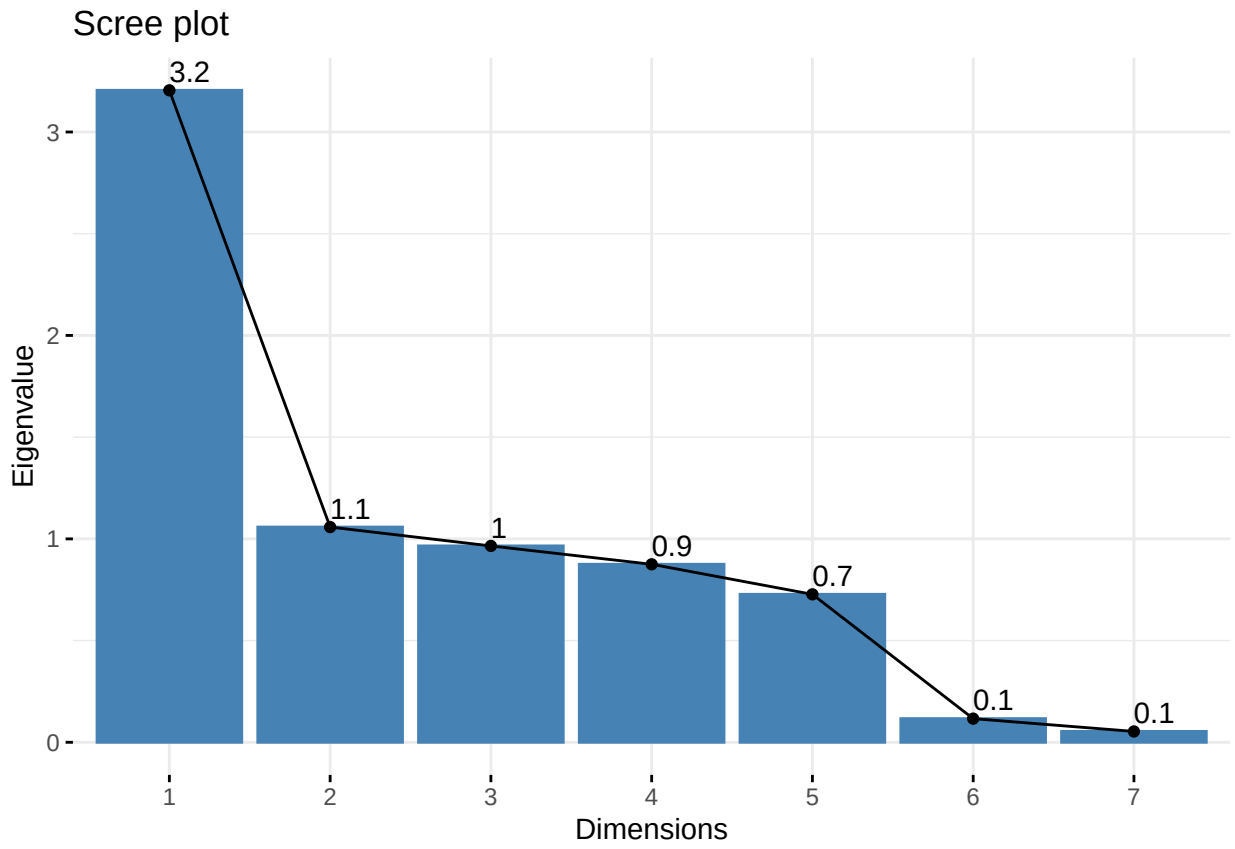



## Scree plots - Principal components selection

```
#scree plot with percentage variance explained
fviz_eig(cont_df.pca, addlabels = TRUE, ylim = c(0, 50))
```




```
#scree plot with eigenvalues
fviz_eig(cont_df.pca, choice = "eigenvalue",
         addlabels=TRUE)
```



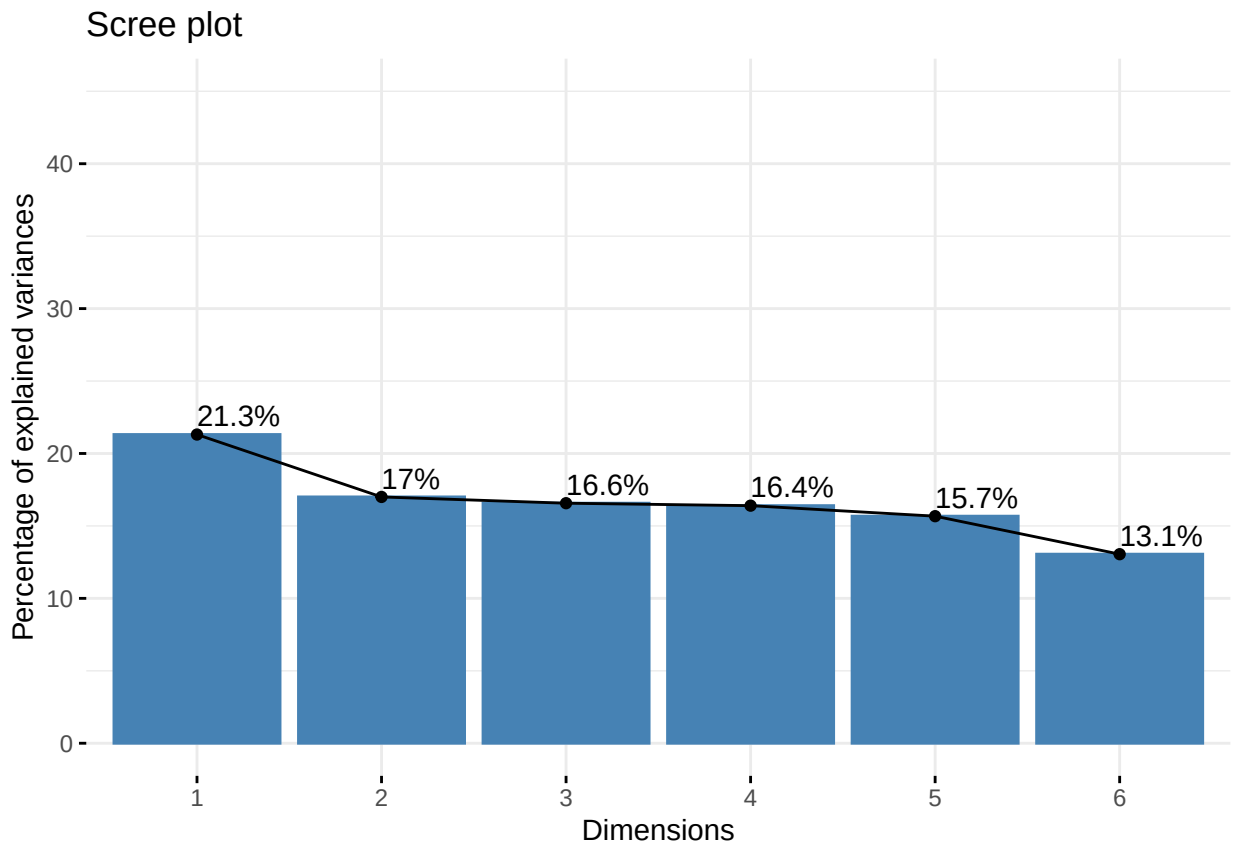

## MCA

```
#selecting all categorical data and converting them to factors

discrete_df <- df[c('gender', 'voyage', 'laugh', 'photo.keke', 'photo.beach')]

discrete_df$gender <- as.factor(discrete_df$gender)
discrete_df$voyage <- as.factor(discrete_df$voyage)
discrete_df$laugh <- as.factor(discrete_df$laugh)
discrete_df$photo.keke <- as.factor(discrete_df$photo.keke)
discrete_df$photo.beach <- as.factor(discrete_df$photo.beach)
head(discrete_df)
```

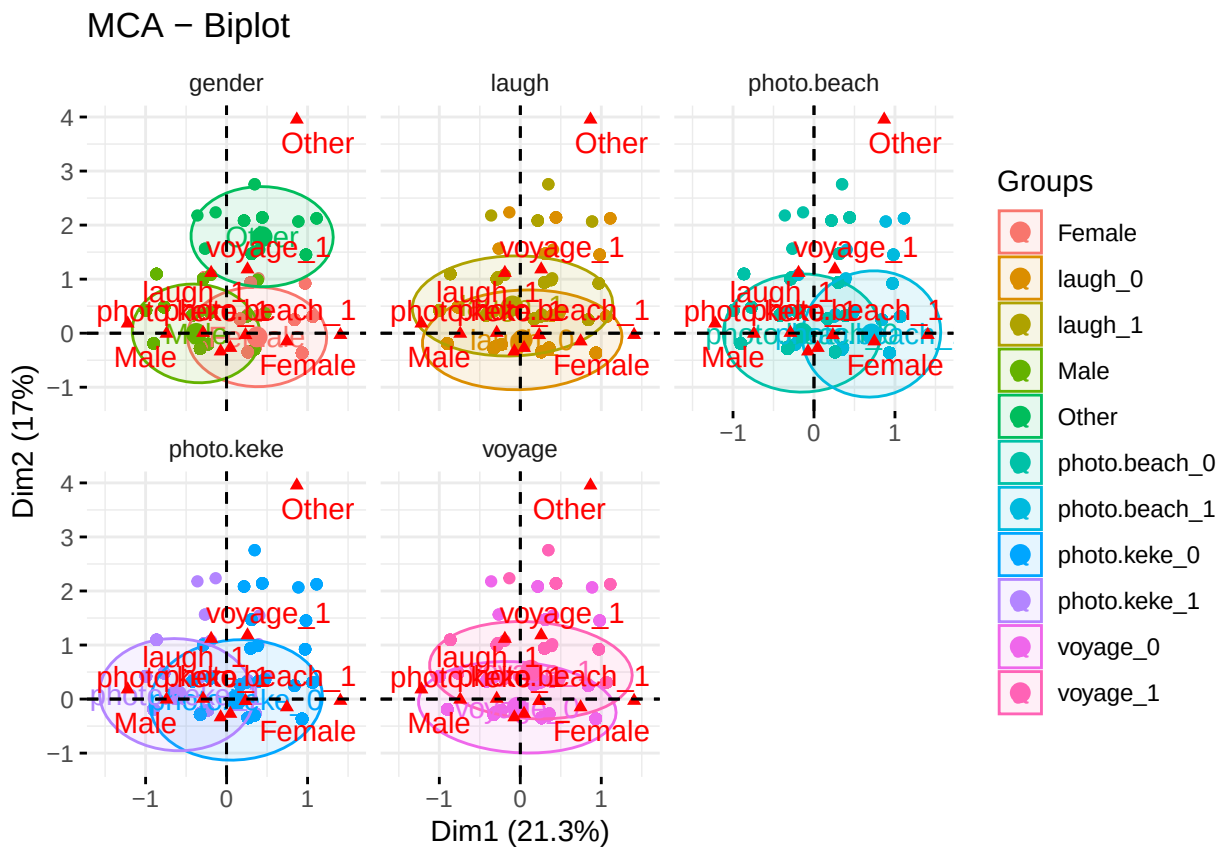
```
## # A tibble: 6 x 5
##   gender voyage laugh photo.keke photo.beach
##   <fct>  <fct>  <fct> <fct>      <fct>
## 1 Female 0      0     0          0
## 2 Female 0      0     0          1
## 3 Female 0      0     0          1
## 4 Male   0      0     0          1
## 5 Male   0      1     0          0
## 6 Female 0      0     0          0
```

```r
#apply MCA on categorical dataset
discrete_df.mca <- MCA(discrete_df, graph = FALSE)
```

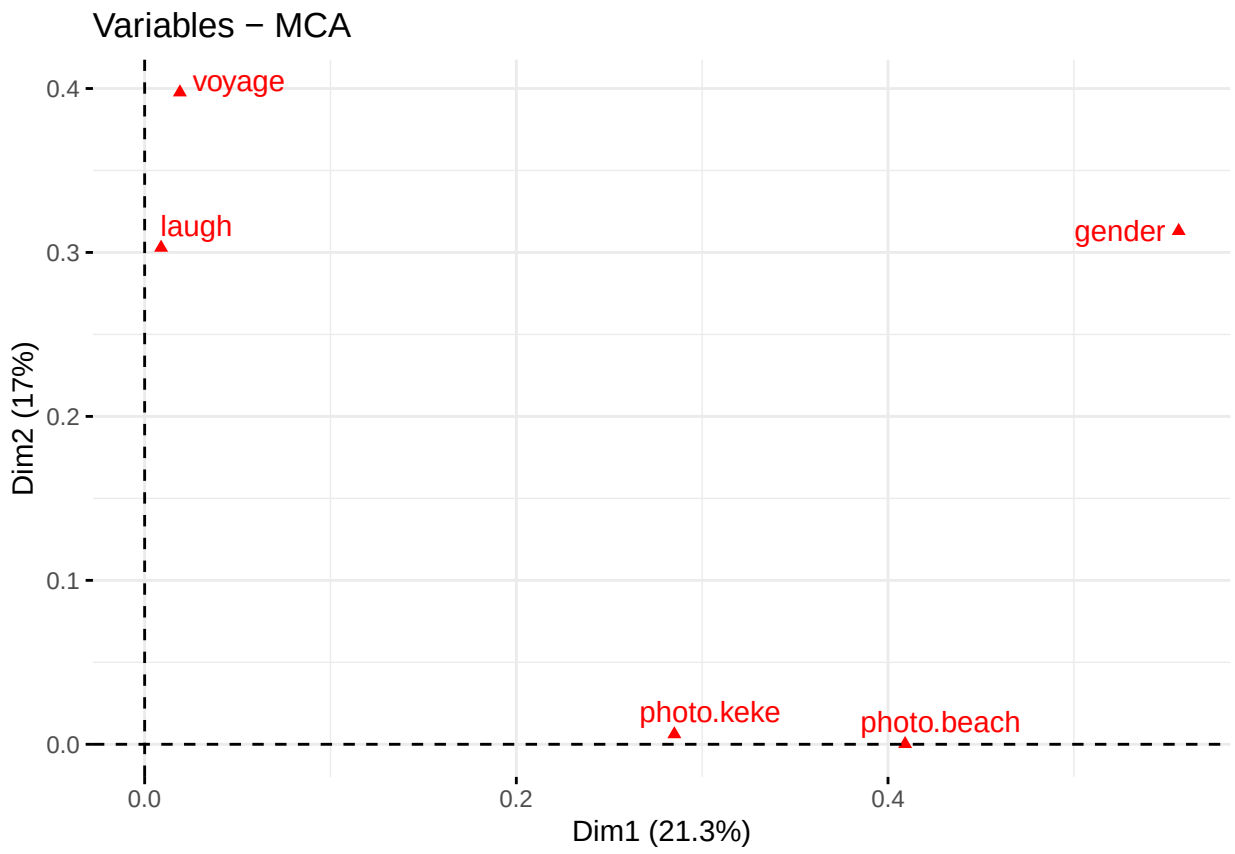
```r
#Scree plot for MCA in percentage
fviz_screeplot(discrete_df.mca, addlabels = TRUE, ylim = c(0, 45))
```



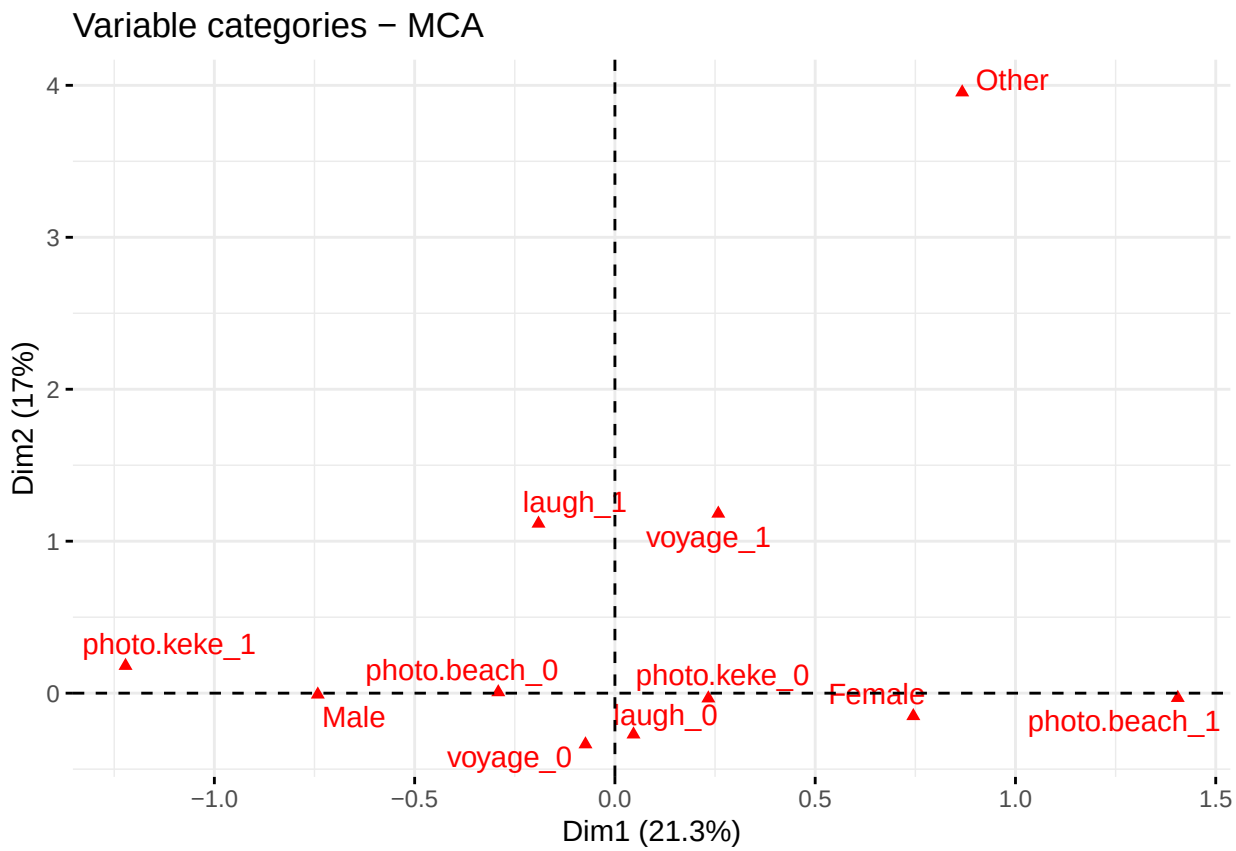

```r
#mca biplot split facetted by each group
fviz_mca_biplot(discrete_df.mca,
                repel = TRUE, # Avoid text overlapping (slow if many point)
                addEllipses = TRUE,
                habillage=colnames(discrete_df),

             ggtheme = theme_minimal())
```



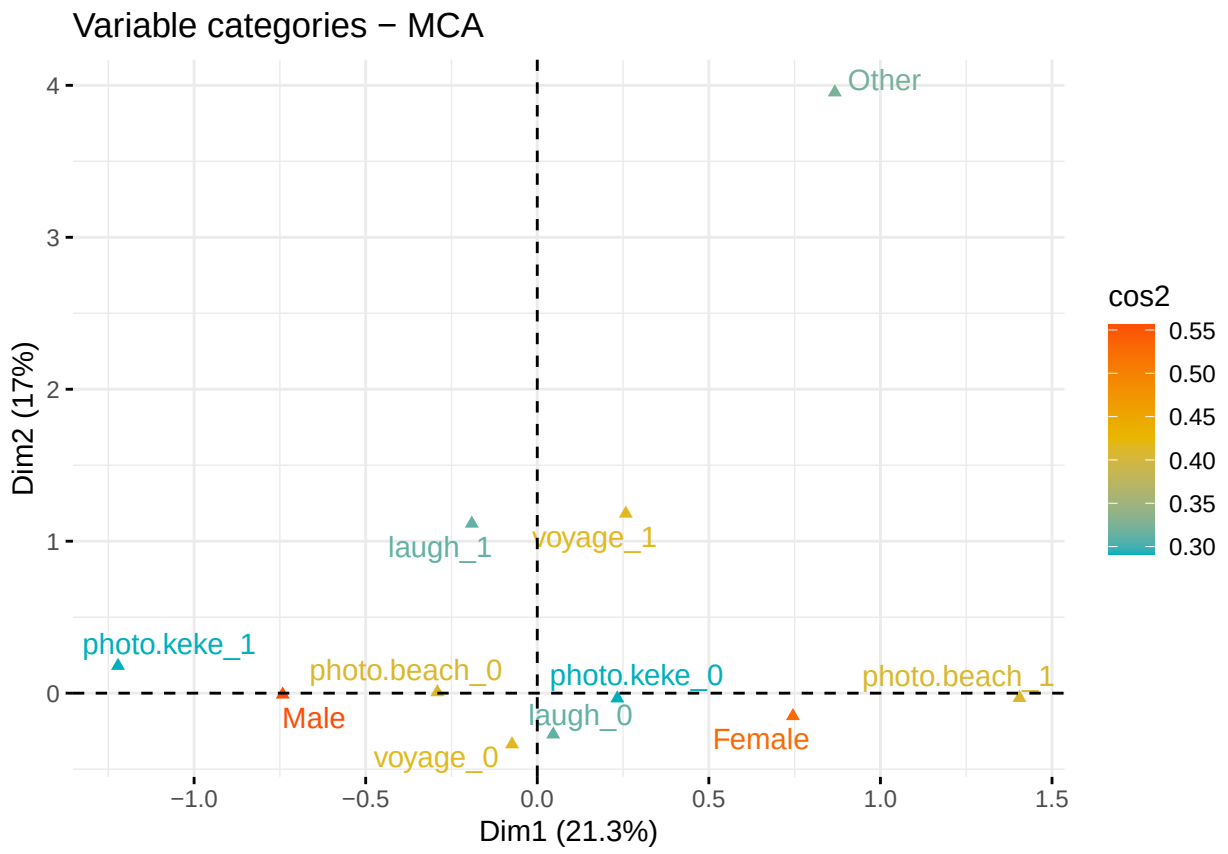

```
# Correlation between variables and principal dimensions - MCA
fviz_mca_var(discrete_df.mca, choice = "mca.cor",
             repel = TRUE, # Avoid text overlapping (slow)
             ggtheme = theme_minimal())
```



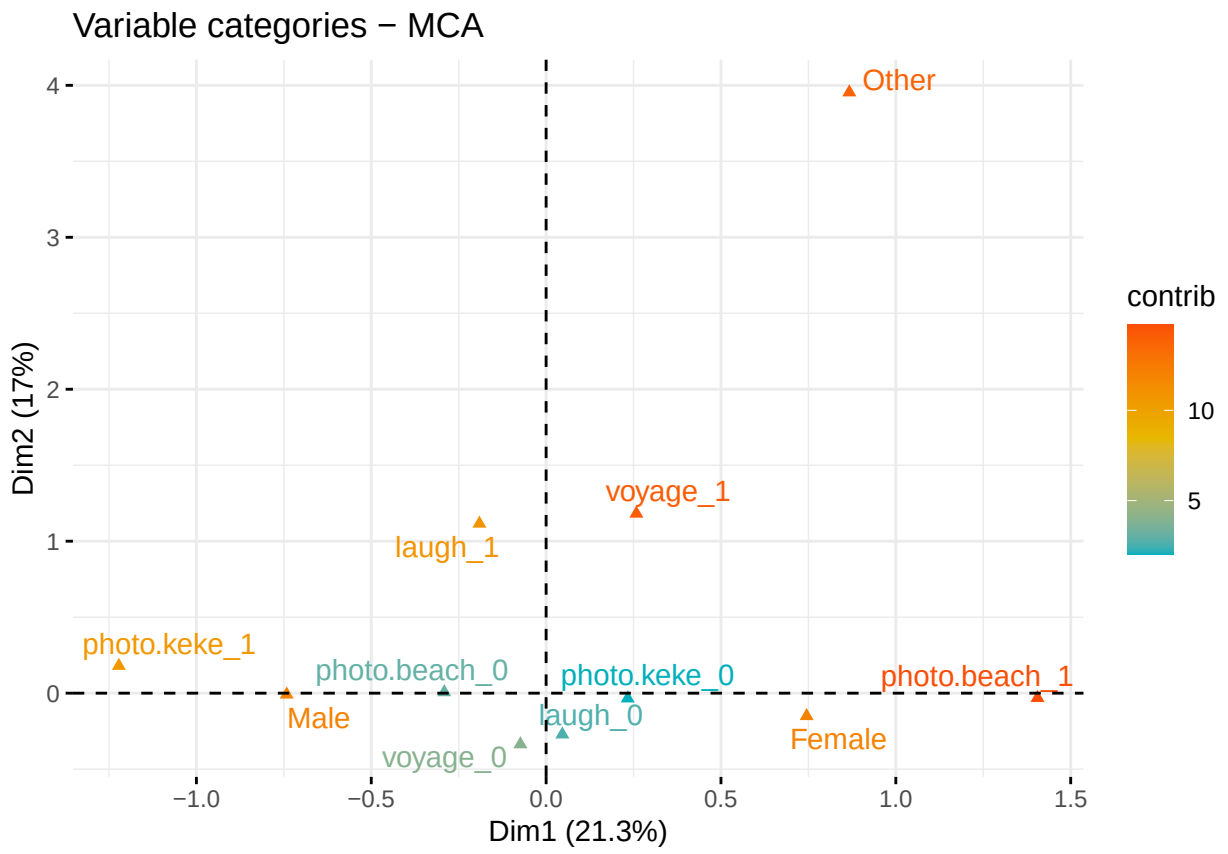

```
#coordinates of each variable categories in each dimension - MCA
fviz_mca_var(discrete_df.mca,
             repel = TRUE, # Avoid text overlapping (slow)
             ggtheme = theme_minimal())
```



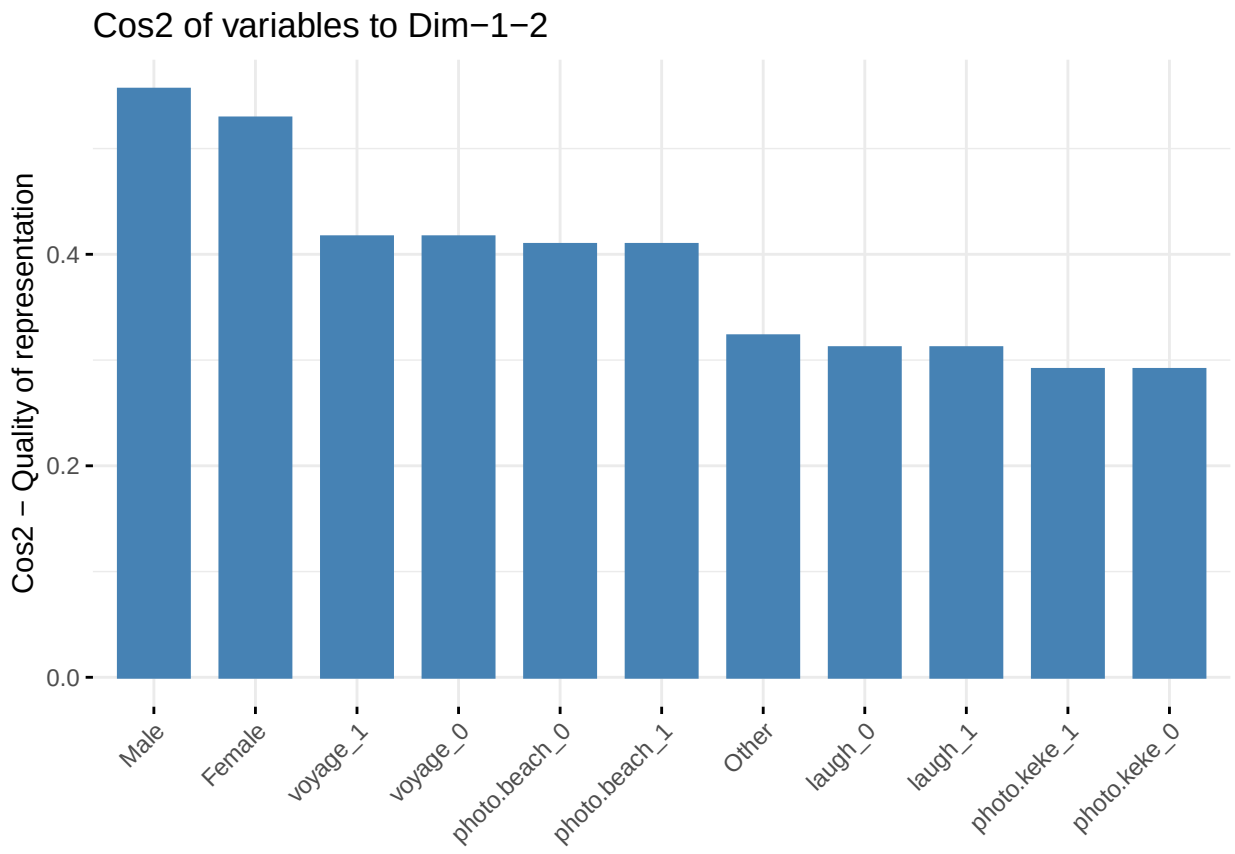

```
# coordinates of each variable categories in each dimension - MCA
# Color by cos2 values: quality on the factor map
fviz_mca_var(discrete_df.mca, col.var = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE, # Avoid text overlapping
             ggtheme = theme_minimal())
```



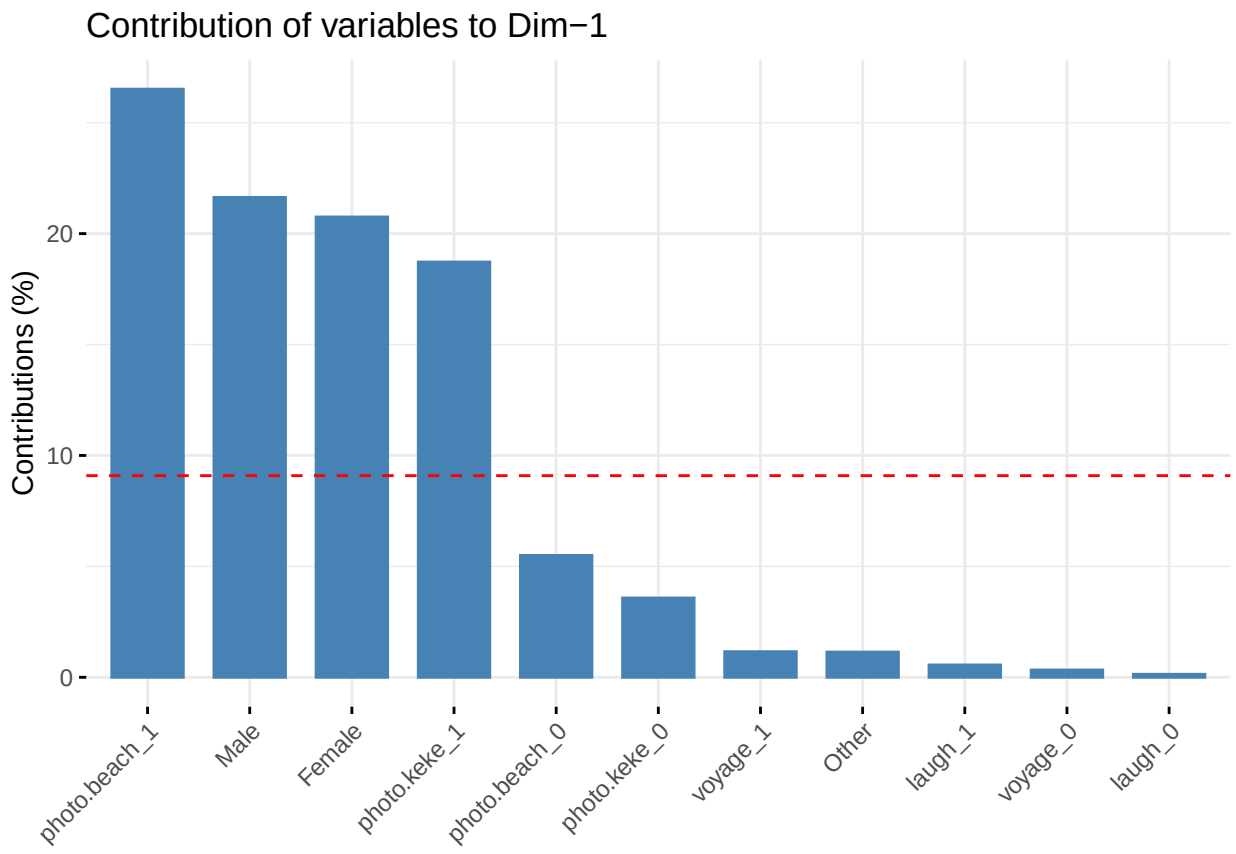

```
#coordinates of each variable categories in each dimension - MCA
# Color by contribution values

fviz_mca_var(discrete_df.mca, col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE, # Avoid text overlapping
             ggtheme = theme_minimal())
```



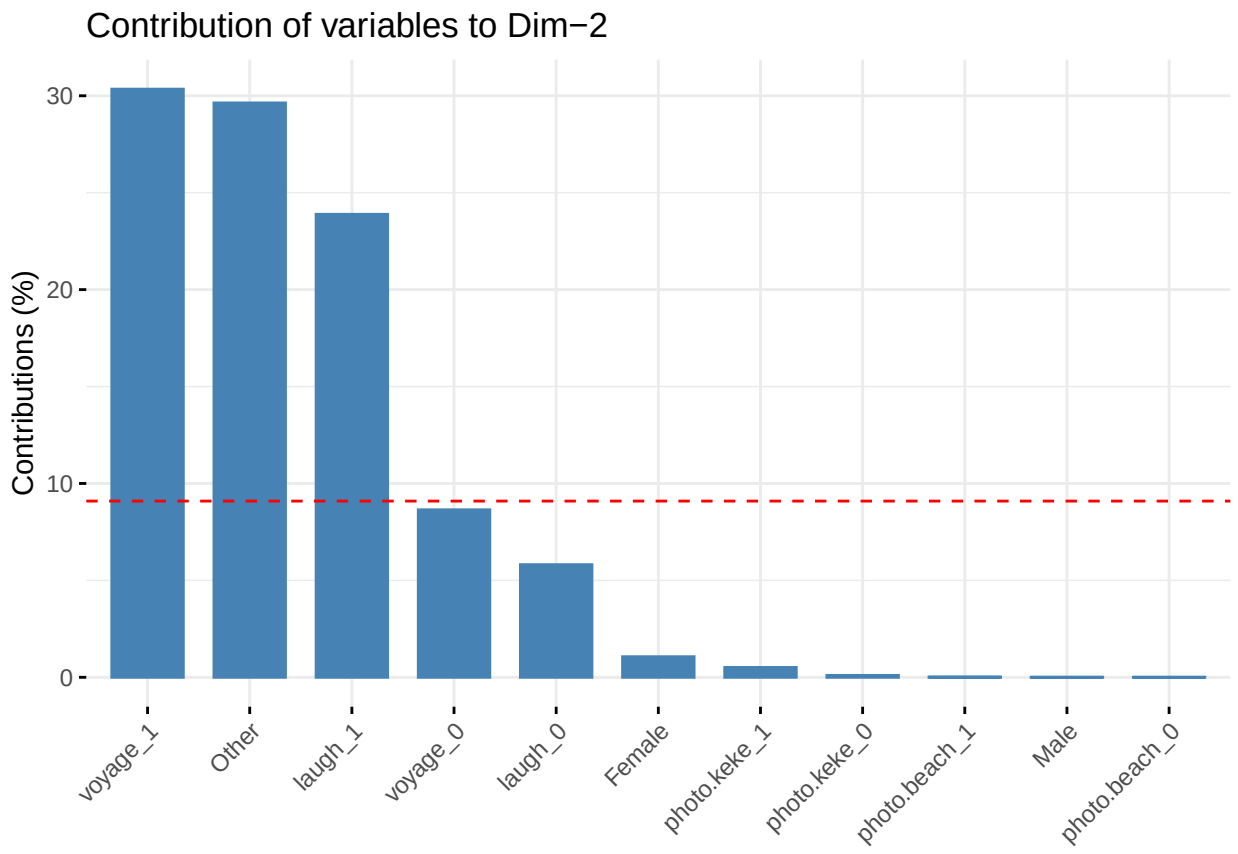

```
#a bar plot of variable cos2
# Cos2 of variable categories on Dim.1 and Dim.2
fviz_cos2(discrete_df.mca, choice = "var", axes = 1:2)
```



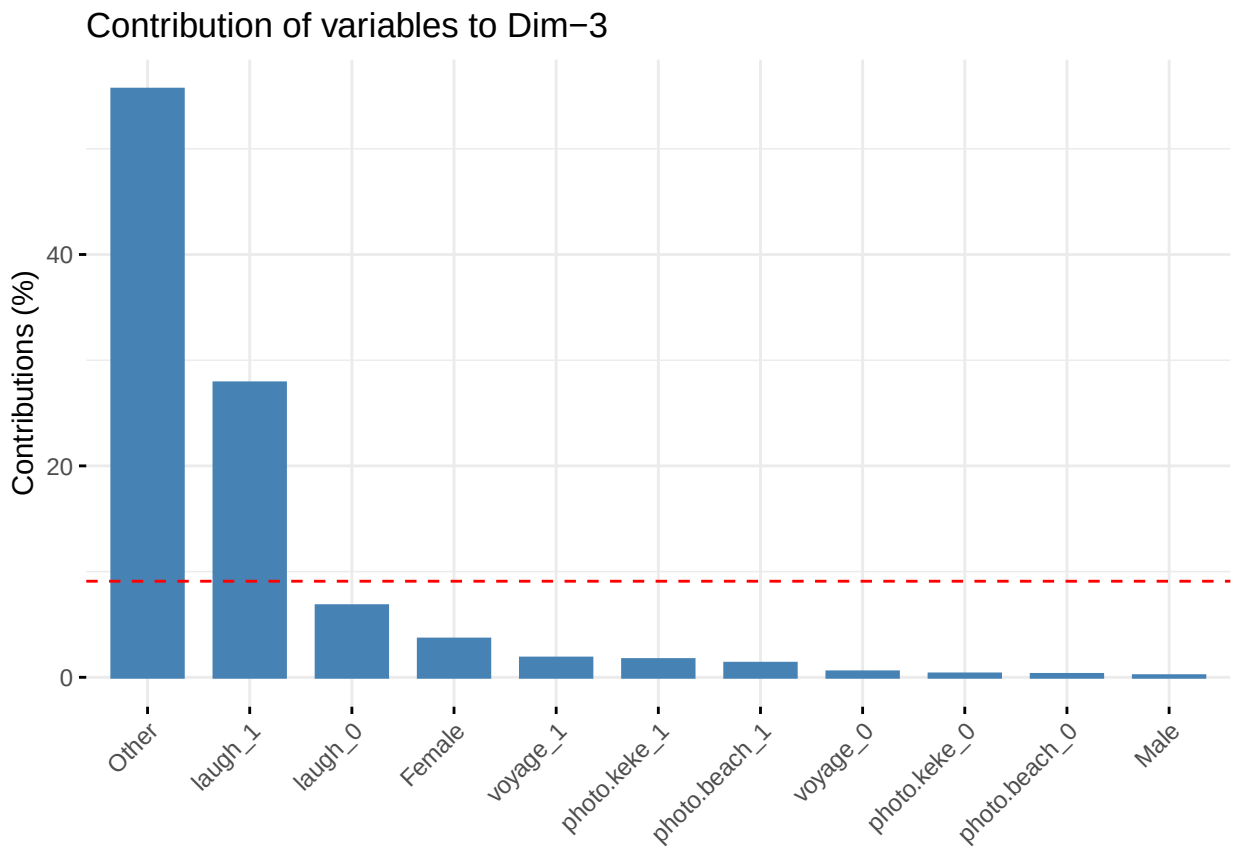

```
#contribution of variable categories on first PC - Barplot
fviz_contrib(discrete_df.mca, choice = "var", axes = 1)
```



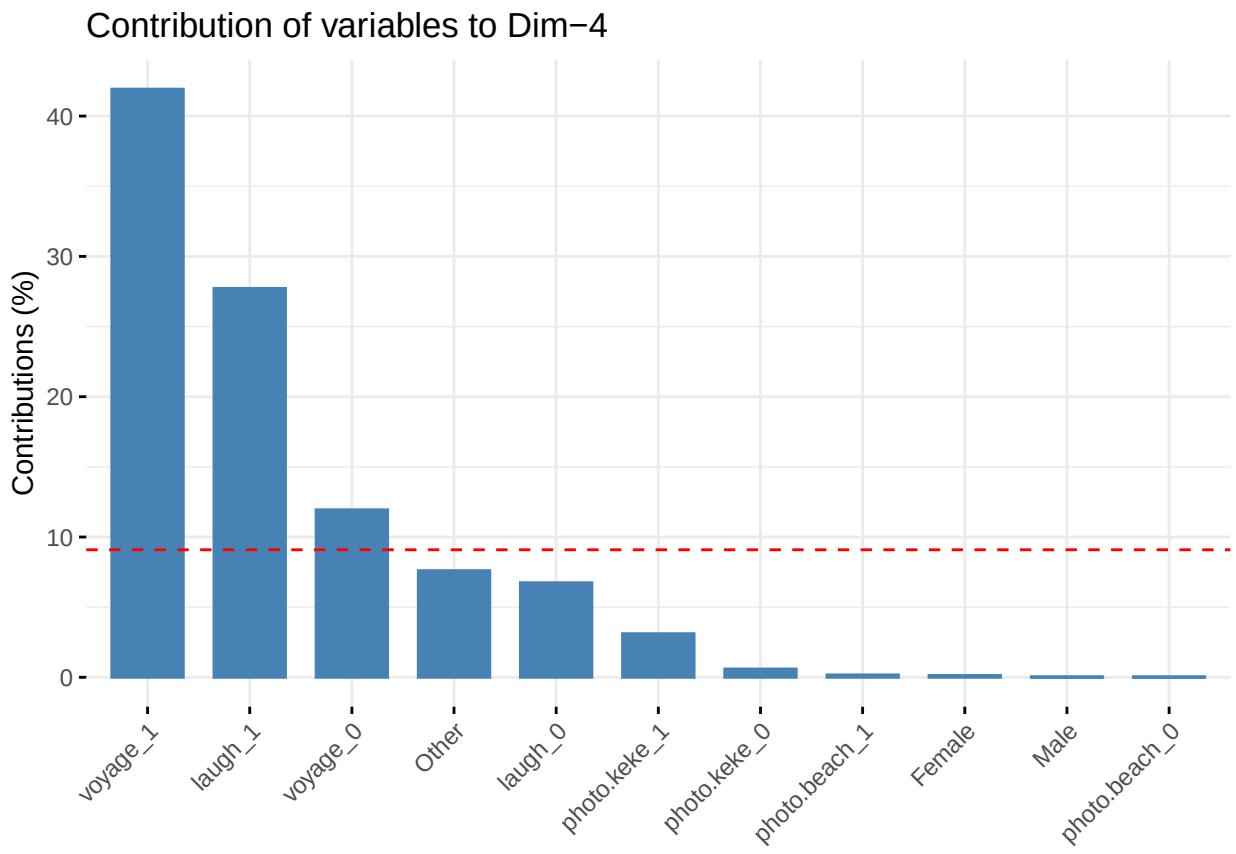

```
#contribution of variable categories on second Pc- Barplot
fviz_contrib(discrete_df.mca, choice = "var", axes = 2)
```



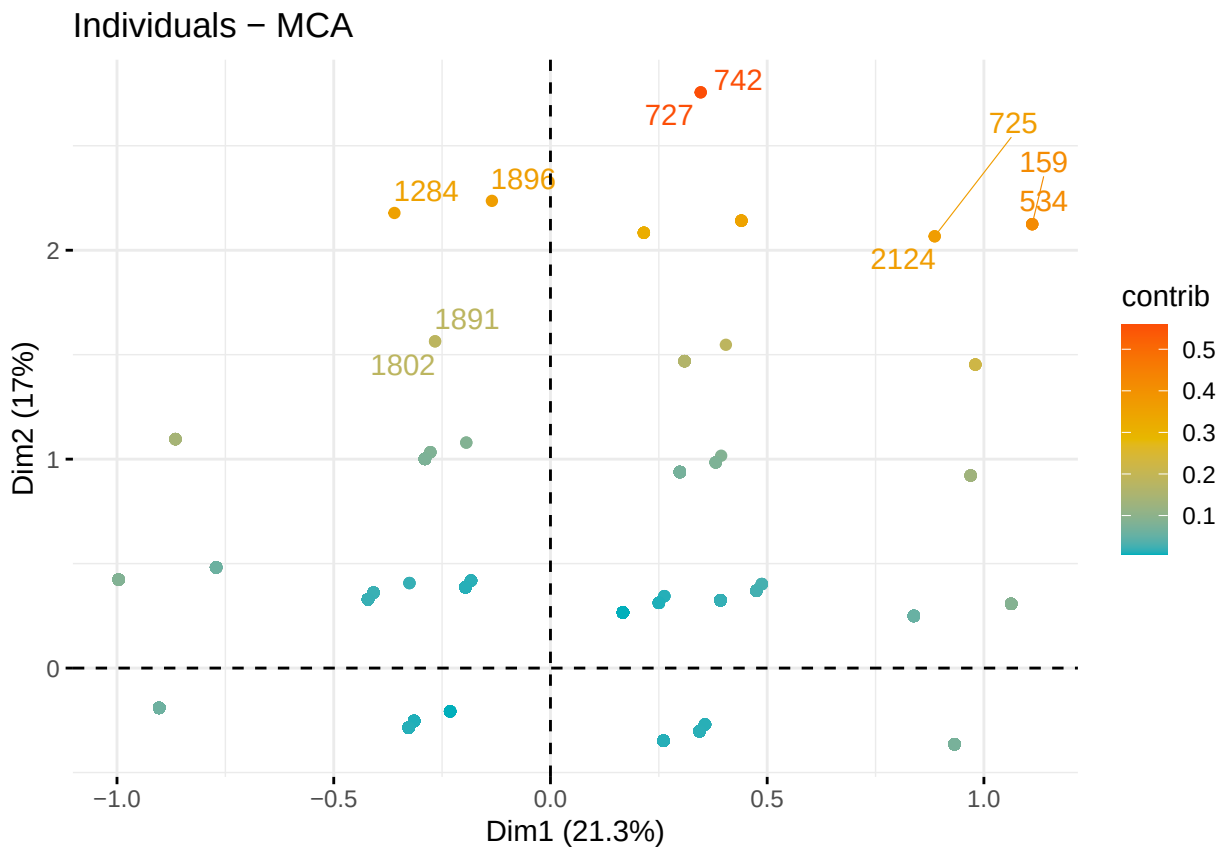

```
#contribution of variable categories on third Pc- Barplot
fviz_contrib(discrete_df.mca, choice = "var", axes = 3)
```



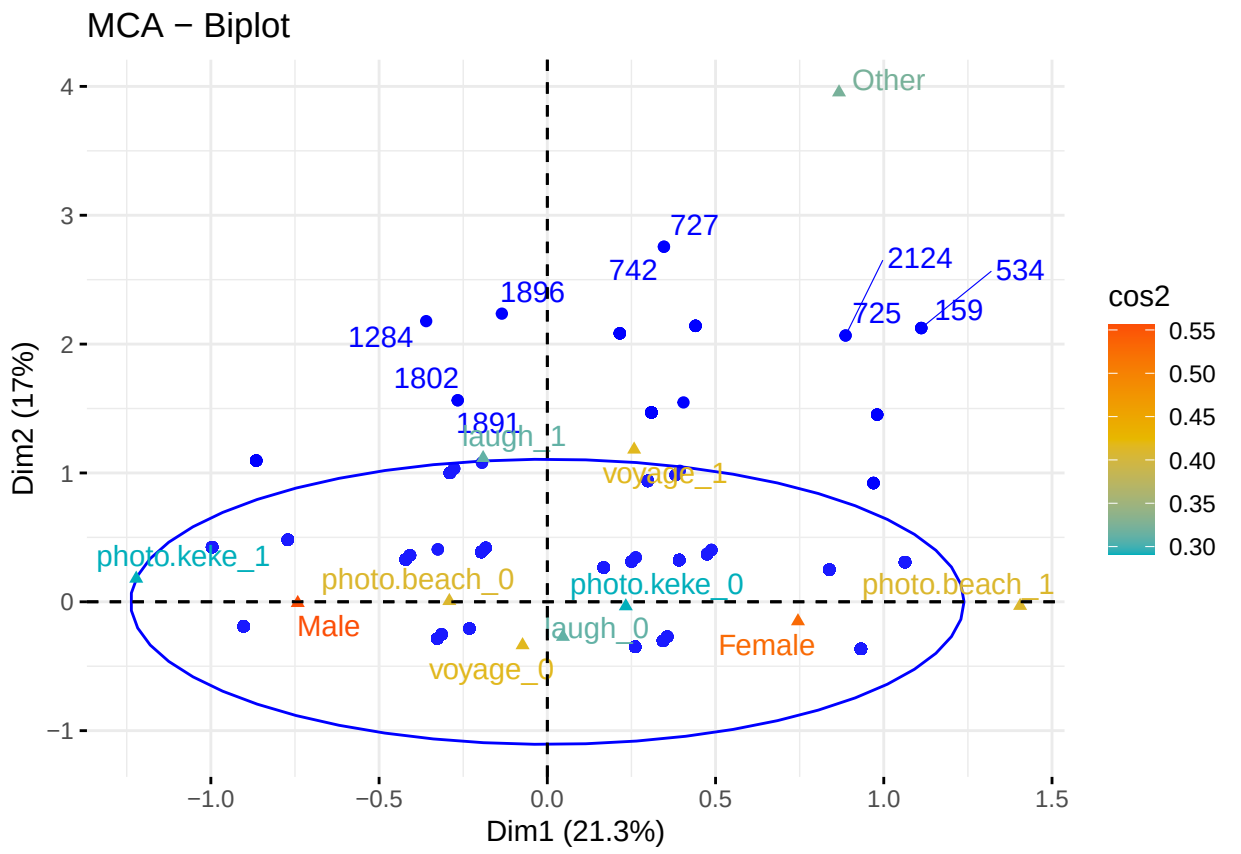

```
#contribution of variable categories on fourth Pc- Barplot
fviz_contrib(discrete_df.mca, choice = "var", axes = 4)
```



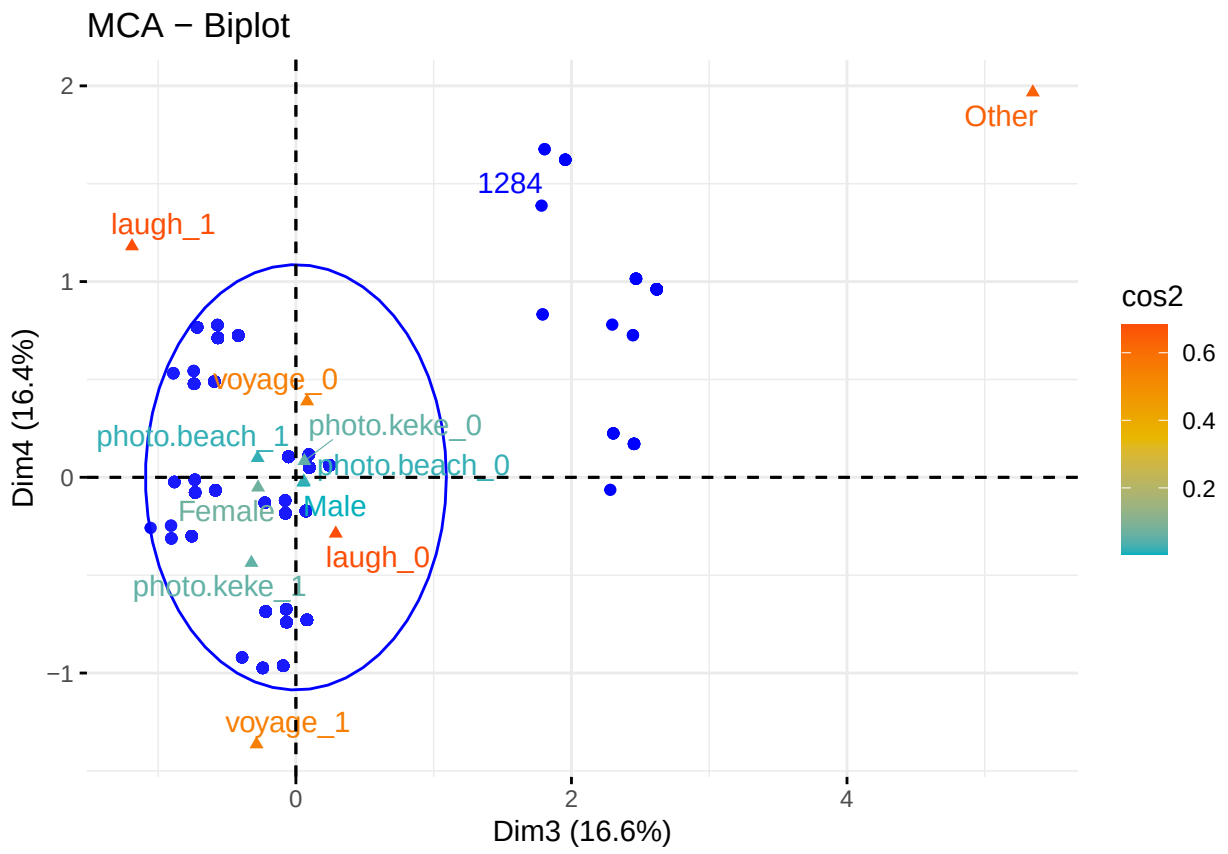

```
# Individuals plot for MCA

fviz_mca_ind(discrete_df.mca, col.ind = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE, # Avoid text overlapping (slow if many points)
             ggtheme = theme_minimal())
```




```
#mca biplot with variable categories + individuals - First and second PCs
#colored by cos2 values
fviz_mca_biplot(discrete_df.mca, col.var = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE, # Avoid text overlapping,
             addEllipses = T,
             ggtheme = theme_minimal())
```




```
#mca biplot with variable categories + individuals - third and fourth PCs
#colored by cos2 values
fviz_mca_biplot(discrete_df.mca, col.var = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE, # Avoid text overlapping,
             addEllipses = T,
             axes=3:4,
             ggtheme = theme_minimal())
```



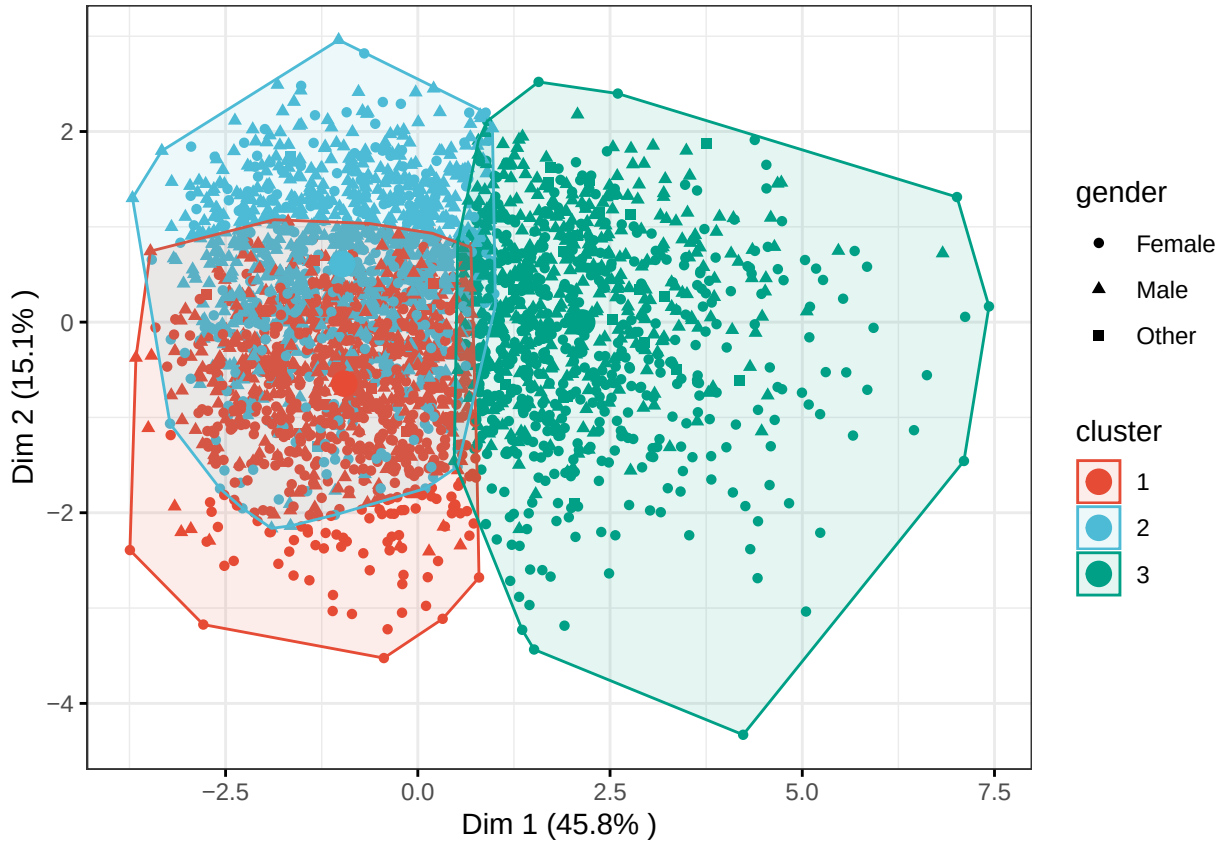

### K-means

```
#apply k-means starting with k = 3
set.seed(123)
res.km <- kmeans(scale(cont_df), 3, nstart = 25)

# Apply pca on continuous variables to reduce dimension
res.pca <- prcomp(cont_df,  scale = TRUE)
#coordinates of individuals
ind.coord <- as.data.frame(get_pca_ind(res.pca)$coord)
#Add clusters obtained by k-means
ind.coord$cluster <- factor(res.km$cluster)

#Add groups of gender obtained from initial dataframe
ind.coord$gender <- df$gender
# Inspect data
head(ind.coord)
```

```
##         Dim.1       Dim.2      Dim.3       Dim.4      Dim.5       Dim.6
## 1  0.07099475 -1.40395830 -0.2070684  1.49327900 -1.8800390  0.22372010
## 2  7.10266062 -1.45687764  1.6081774 -0.65724206  2.7825212  0.13690147
## 3  0.78217320  0.50888936  0.8637521  0.38288346 -0.4981963  0.85043506
## 4  2.26566329  0.09503862 -1.5396844  1.26614519  0.0228448 -0.47289288
## 5  0.39620049  0.66295233  1.3916272 -0.58232595  0.2796677 -0.09297676
```

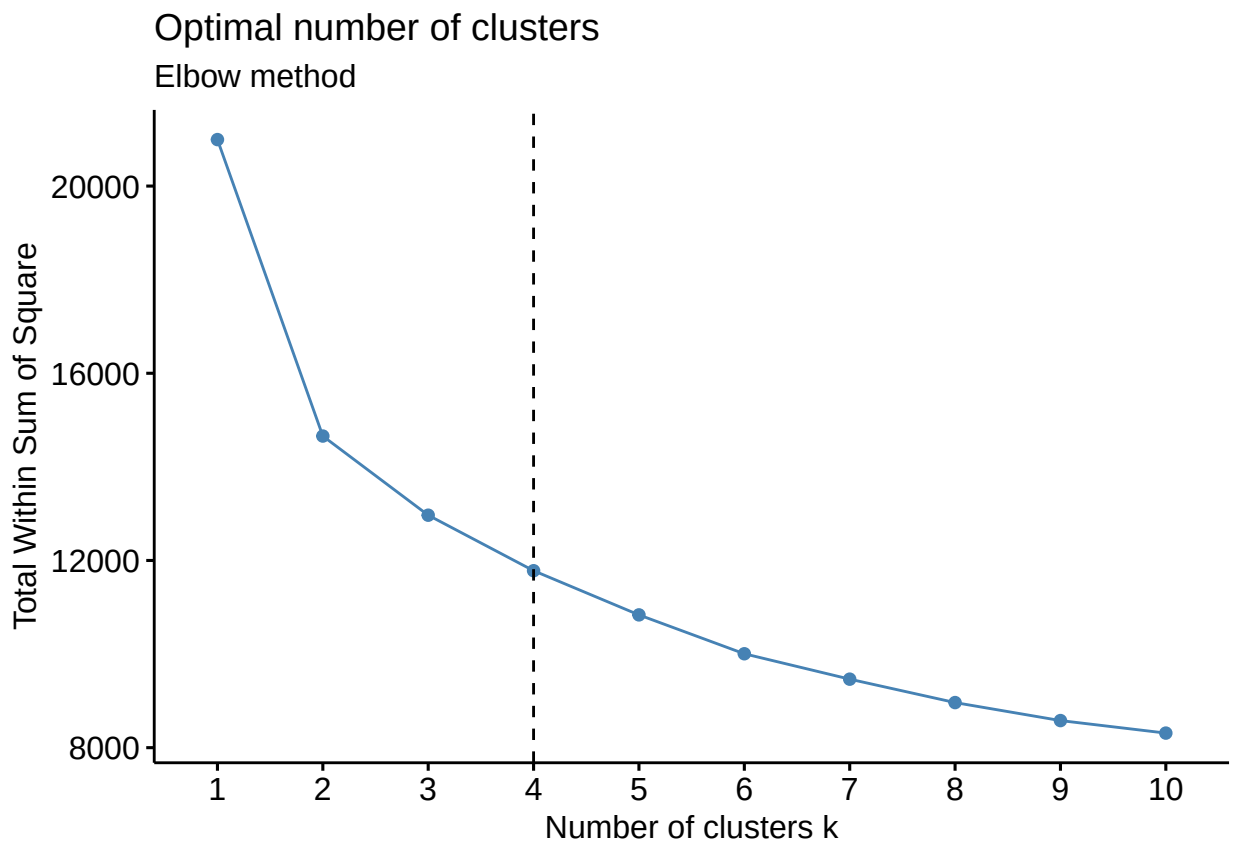
```
## 6 -0.20396414 -0.74217247  1.2379110  0.06917428 -0.5720430  0.10634366
##         Dim.7 cluster gender
## 1 -0.05050701       1 Female
## 2  2.37265399       3 Female
## 3 -0.03466737       3 Female
## 4 -0.12781163       3   Male
## 5 -0.22231619       2   Male
## 6 -0.15469684       2 Female
```

```r
#get eigenvalues
eigenvalue <- round(get_eigenvalue(cont_df.pca), 1)
#get variance percentage from eigenvalues

variance.percent <- eigenvalue$variance.percent
#plot k-means clusters with individual points
ggscatter(
  ind.coord, x = "Dim.1", y = "Dim.2",
  color = "cluster", palette = "npg", ellipse = TRUE, ellipse.type = "convex",
  shape = "gender", size = 1.5,  legend = "right", ggtheme = theme_bw(),
  xlab = paste0("Dim 1 (", variance.percent[1], "% )" ),
  ylab = paste0("Dim 2 (", variance.percent[2], "% )" )
) +
  stat_mean(aes(color = cluster), size = 4)
```

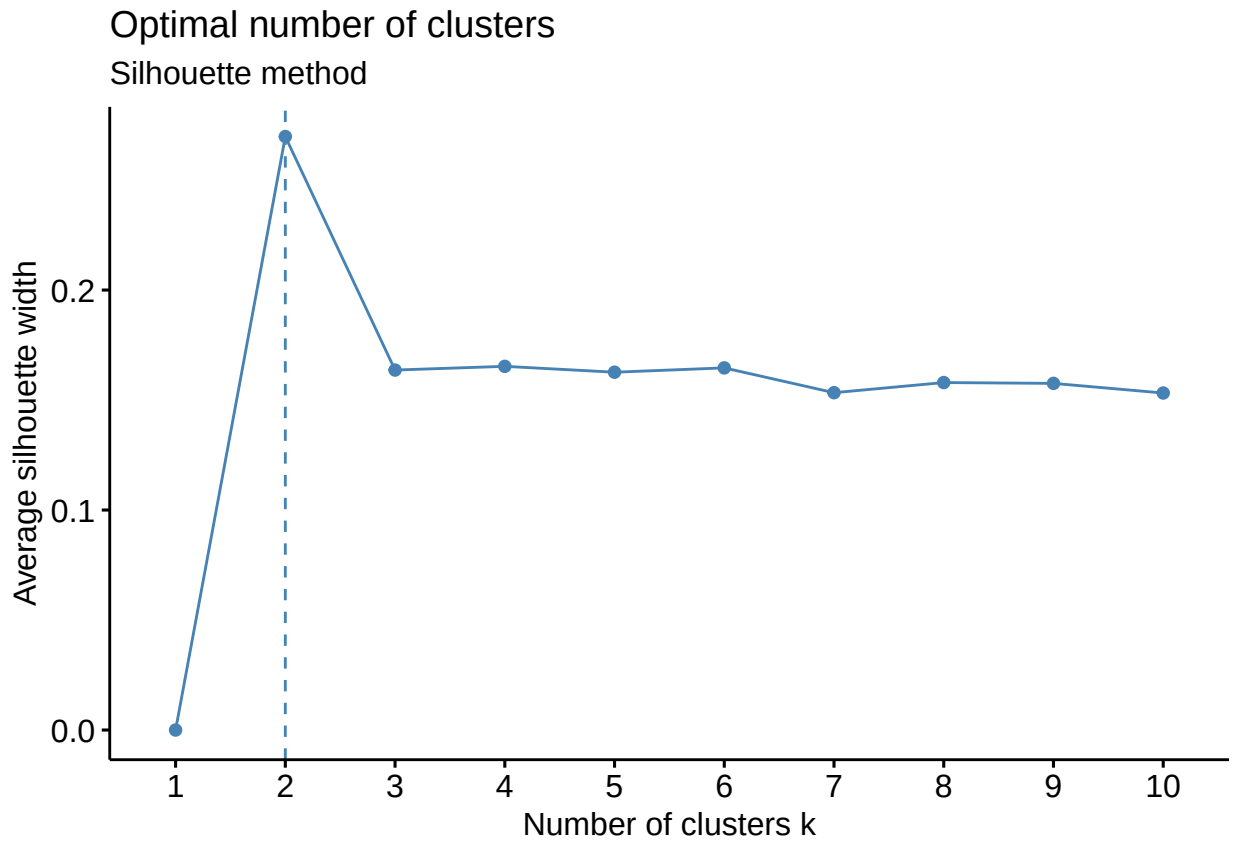


### Optimal number of clusters - k-means

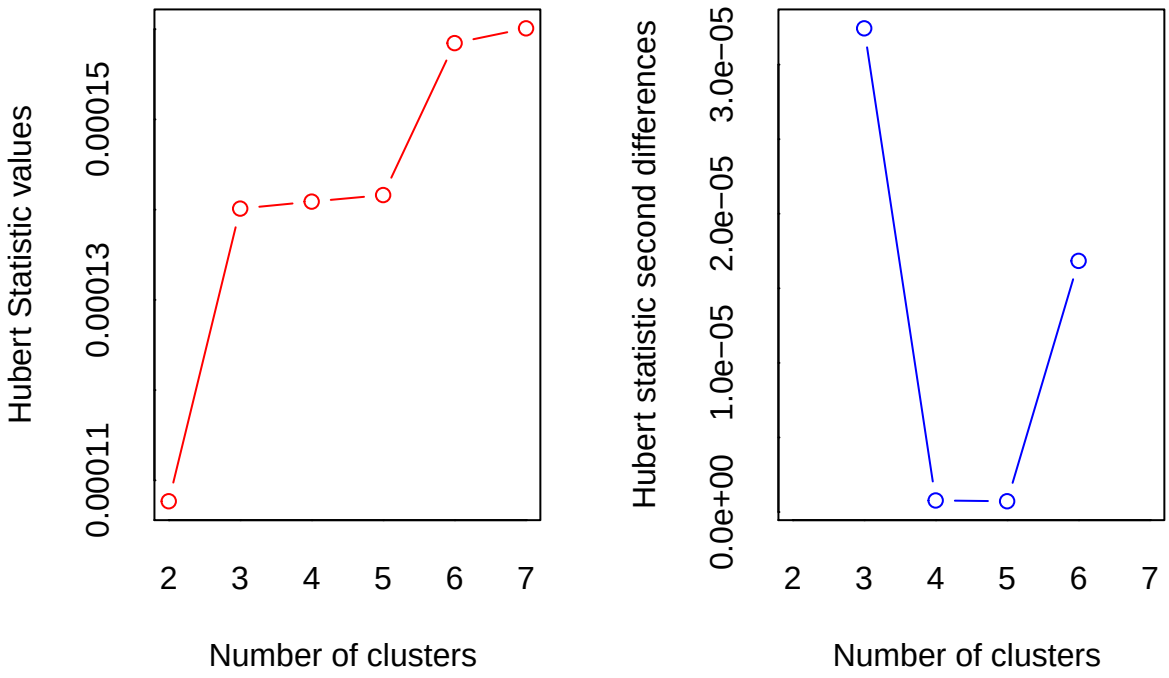
```
#scale df
scaled_contdf <- scale(cont_df)

# Elbow method
fviz_nbclust(scaled_contdf, kmeans, method = "wss") +
  geom_vline(xintercept = 4, linetype = 2)+
  labs(subtitle = "Elbow method")
```



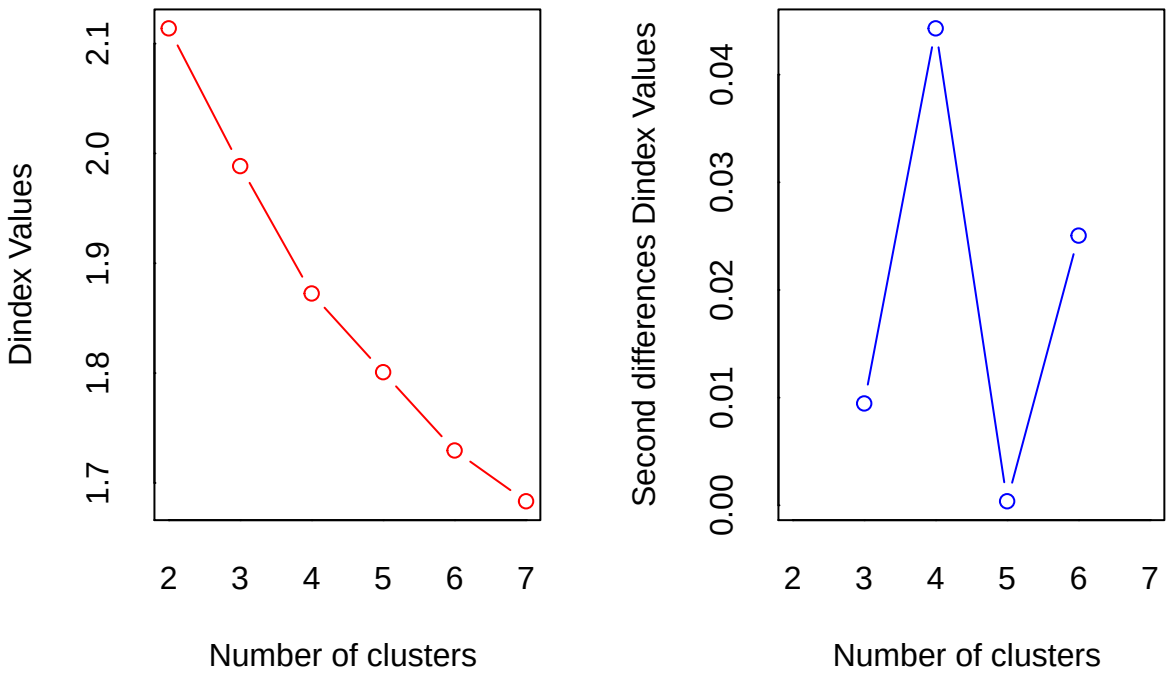

```
# Silhouette method
fviz_nbclust(scaled_contdf, kmeans, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```




```
#use majority rule of 30 indices to get optimal number of clusters
nb <- NbClust(scaled_contdf, distance = "euclidean", min.nc = 2,
              max.nc = 7, method = "kmeans")
```




```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```

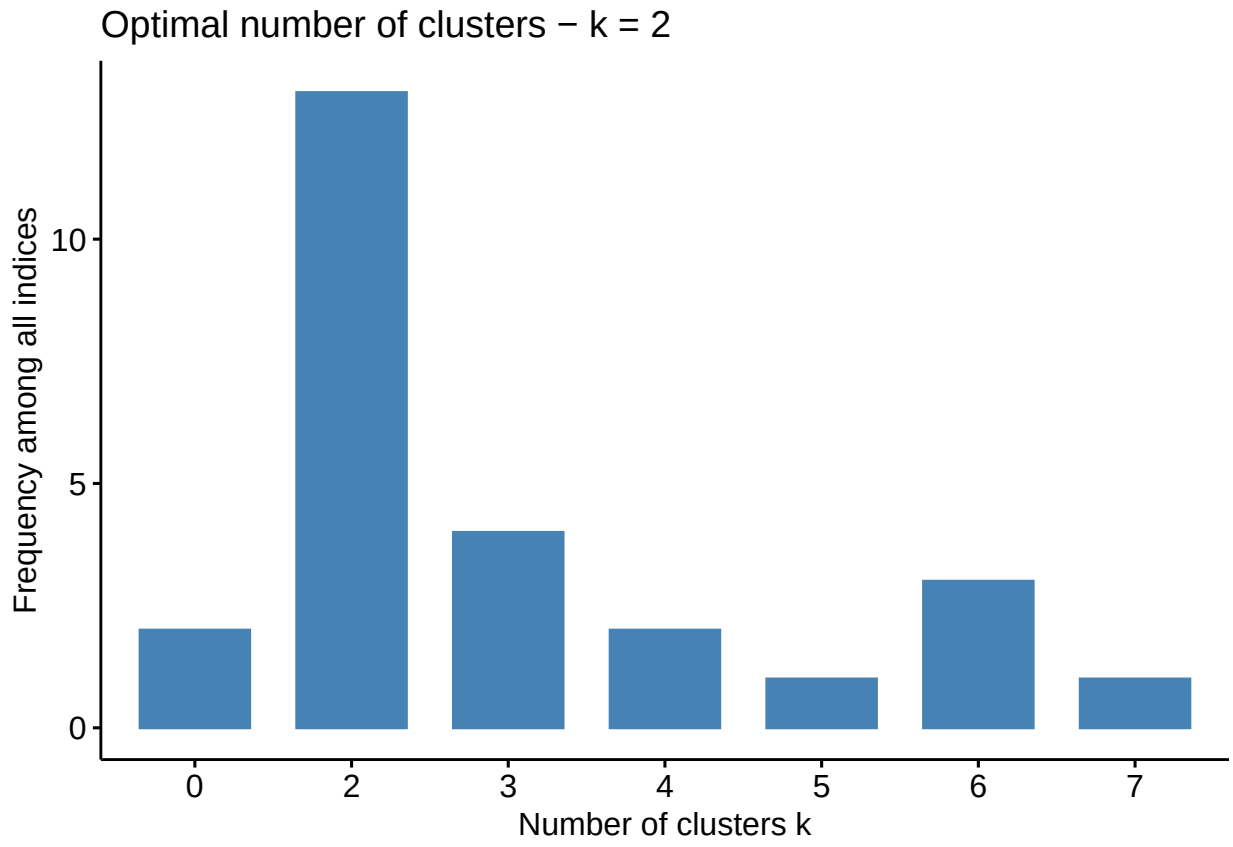



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 13 proposed 2 as the best number of clusters
## * 4 proposed 3 as the best number of clusters
## * 2 proposed 4 as the best number of clusters
## * 1 proposed 5 as the best number of clusters
## * 3 proposed 6 as the best number of clusters
## * 1 proposed 7 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

```
fviz_nbclust(nb)
```

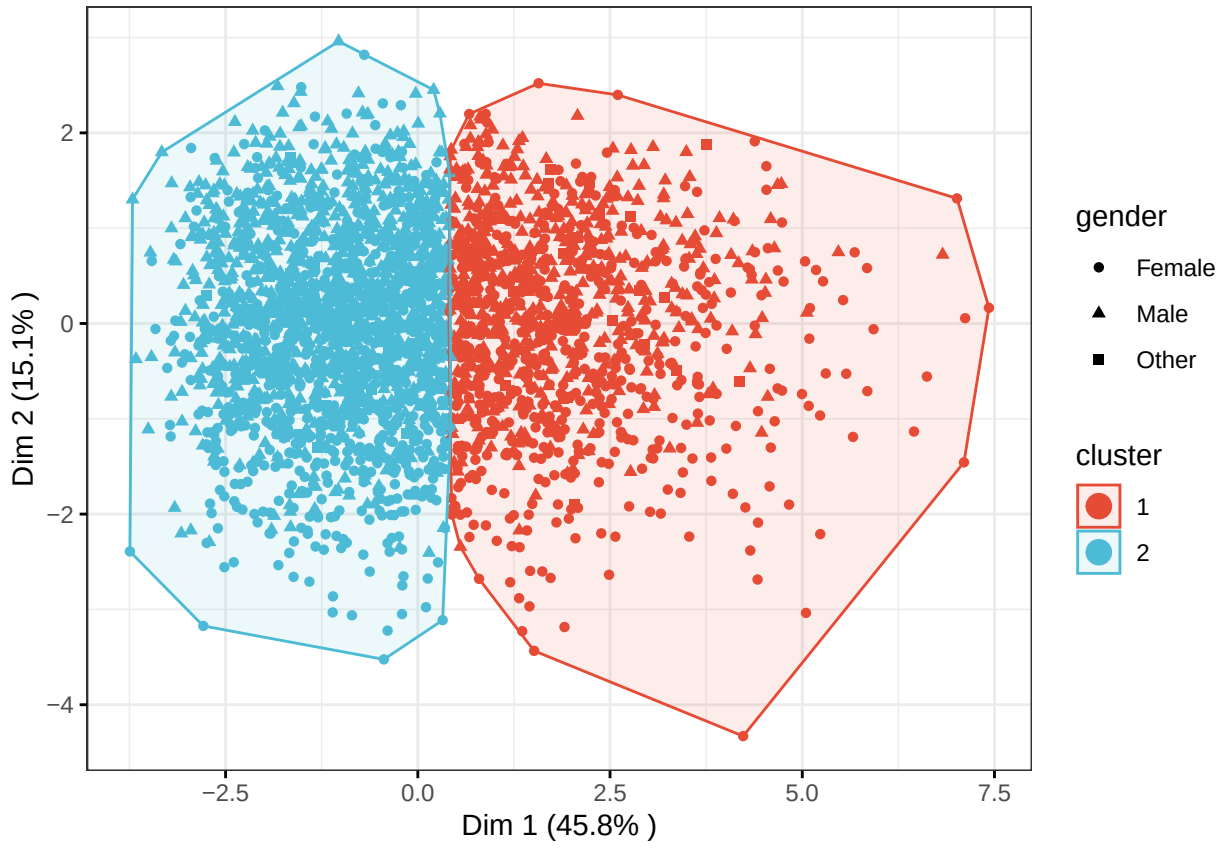
```
## Among all indices:
## ===================
```

```
## * 2 proposed  0 as the best number of clusters
## * 13 proposed  2 as the best number of clusters
## * 4 proposed  3 as the best number of clusters
## * 2 proposed  4 as the best number of clusters
## * 1 proposed  5 as the best number of clusters
## * 3 proposed  6 as the best number of clusters
## * 1 proposed  7 as the best number of clusters
##
## Conclusion
## =========================
## * According to the majority rule, the best number of clusters is  2 .
```




```
# optimal number of clusters, k = 2
# re-plot clusters with k = 2
set.seed(123)
res.km2 <- kmeans(scale(cont_df), 2, nstart = 25)


#coordinates of individuals
ind.coord2 <- as.data.frame(get_pca_ind(res.pca)$coord)
#Add clusters obtained by k-means
ind.coord2$cluster <- factor(res.km2$cluster)

#Add groups of gender obtained from initial dataframe
ind.coord2$gender <- df$gender
```

```
ggscatter(
  ind.coord2, x = "Dim.1", y = "Dim.2",
  color = "cluster", palette = "npg", ellipse = TRUE, ellipse.type = "convex",
  shape = "gender", size = 1.5,  legend = "right", ggtheme = theme_bw(),
  xlab = paste0("Dim 1 (", variance.percent[1], "% )" ),
  ylab = paste0("Dim 2 (", variance.percent[2], "% )" )
) +
  stat_mean(aes(color = cluster), size = 4)
```




```
#display cluster centers for each variable
res.km2$centers
```

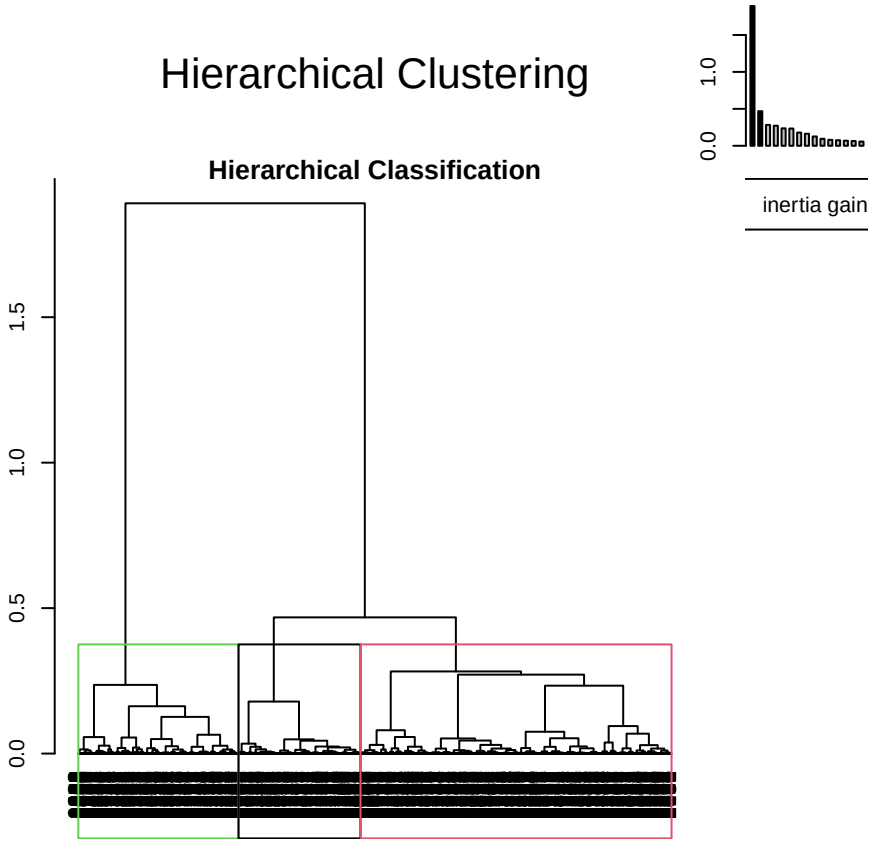
```
##        score  score_log  n.matches n.updates.photo     n.photos   sent.ana
## 1  1.0112455  1.0147384  1.0338898       0.4737756  0.004807421  0.5696334
## 2 -0.5812671 -0.5832748 -0.5942831      -0.2723277 -0.002763321 -0.3274271
##   length.prof
## 1 -0.05857819
## 2  0.03367093
```

## HCPC

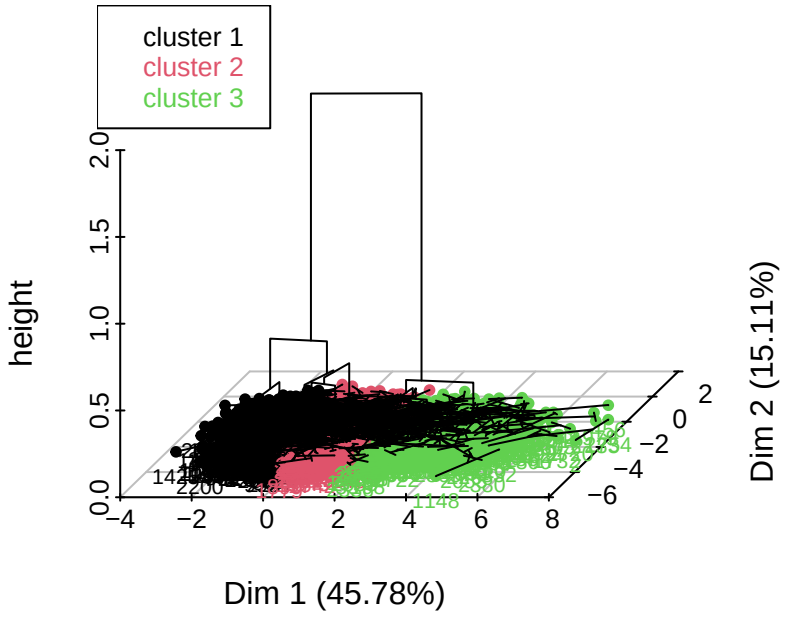
```
# Compute PCA with ncp = 3(principal components)
res.pca <- PCA(scaled_contdf, ncp = 3, graph = FALSE)
```

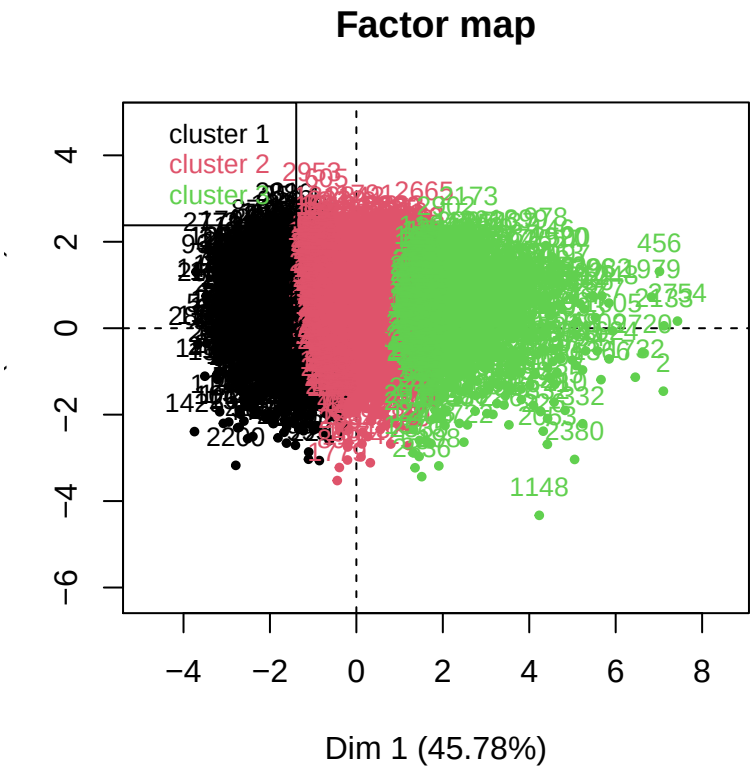
```
# Compute hierarchical clustering on principal components
res.hcpc <- HCPC(res.pca, graph = FALSE)
#dendogram
#fviz_dend(res.hcpc,
#          cex = 0.5, # Label size
#          palette = "jco", # Color palette see ?ggpubr::ggpar
#          rect = TRUE, rect_fill = TRUE, # Add rectangle around groups
#          rect_border = "jco", # Rectangle color
#          labels_track_height = 0.8 # Augment the room for labels
#)"
```

```
#applying PCA to continuous variables
res.pca <- PCA(cont_df, graph = FALSE,scale=TRUE)
#applying HCPC
hcpc <- HCPC(res.pca, min = 3, max = NULL, nb.clust = -1)
```

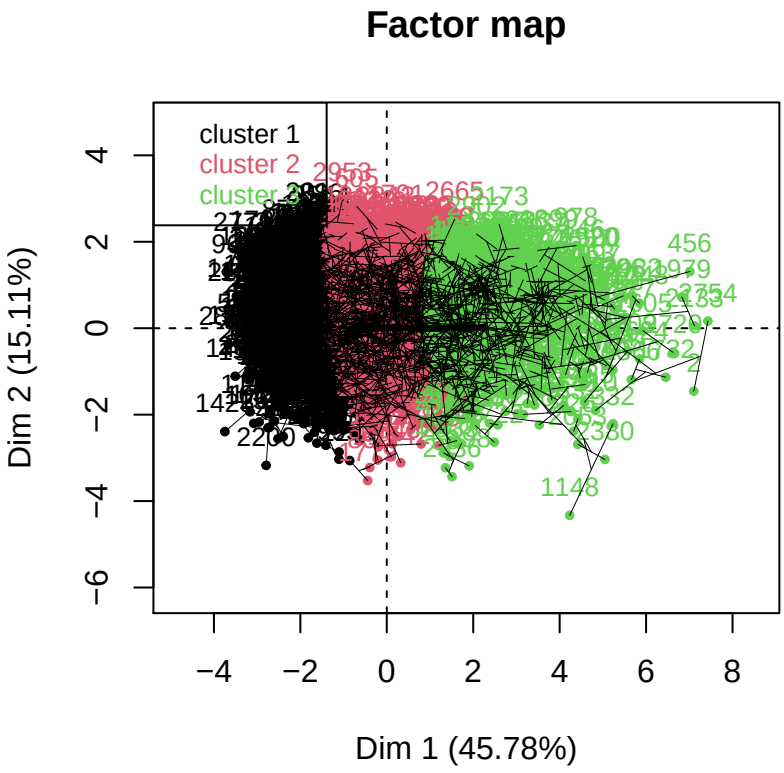


## Hierarchical clustering on the factor map

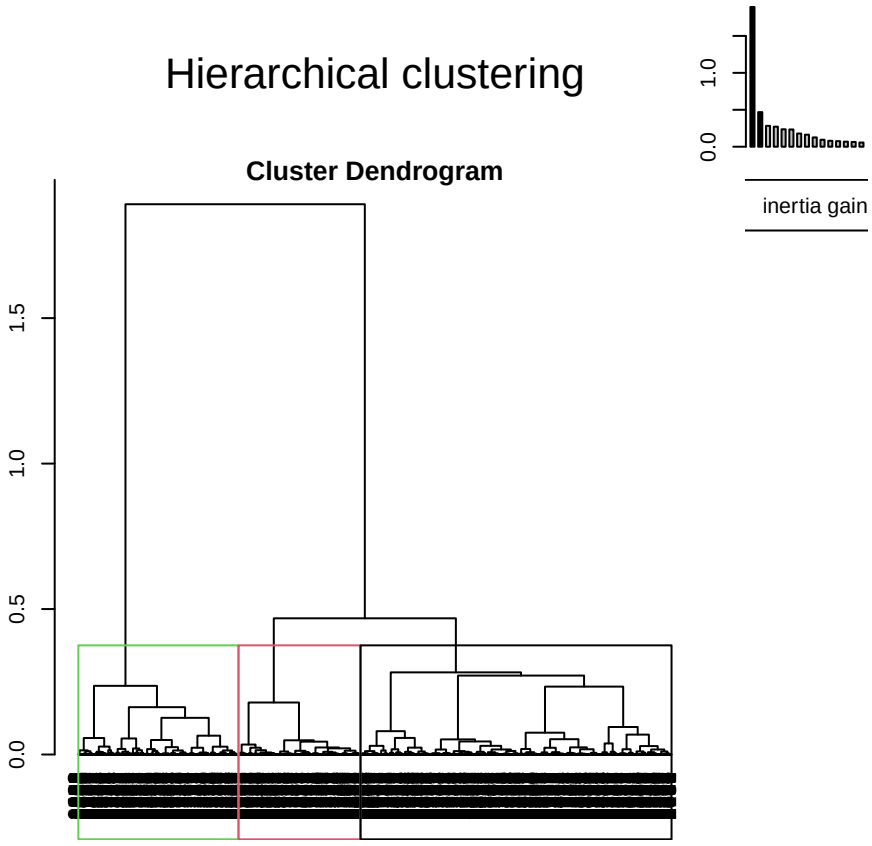




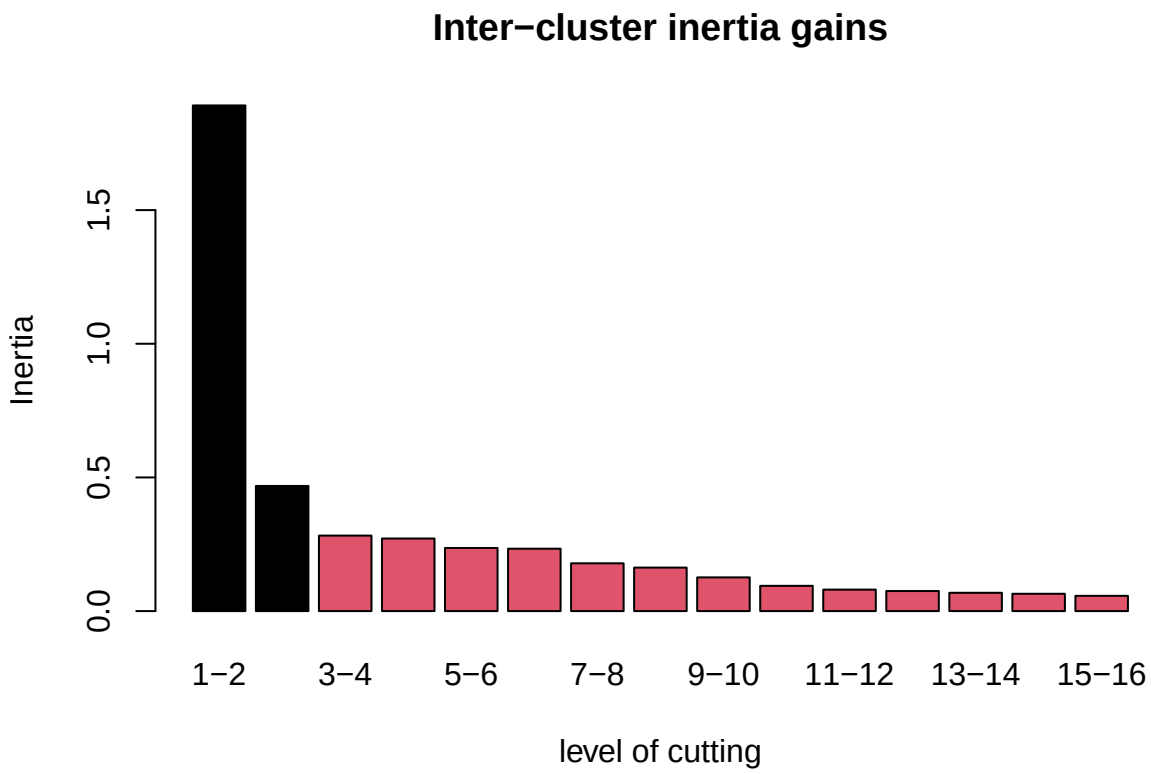

```
# Principal components with 3D map, choice and tree option

plot(hcpc, choice="map")
```



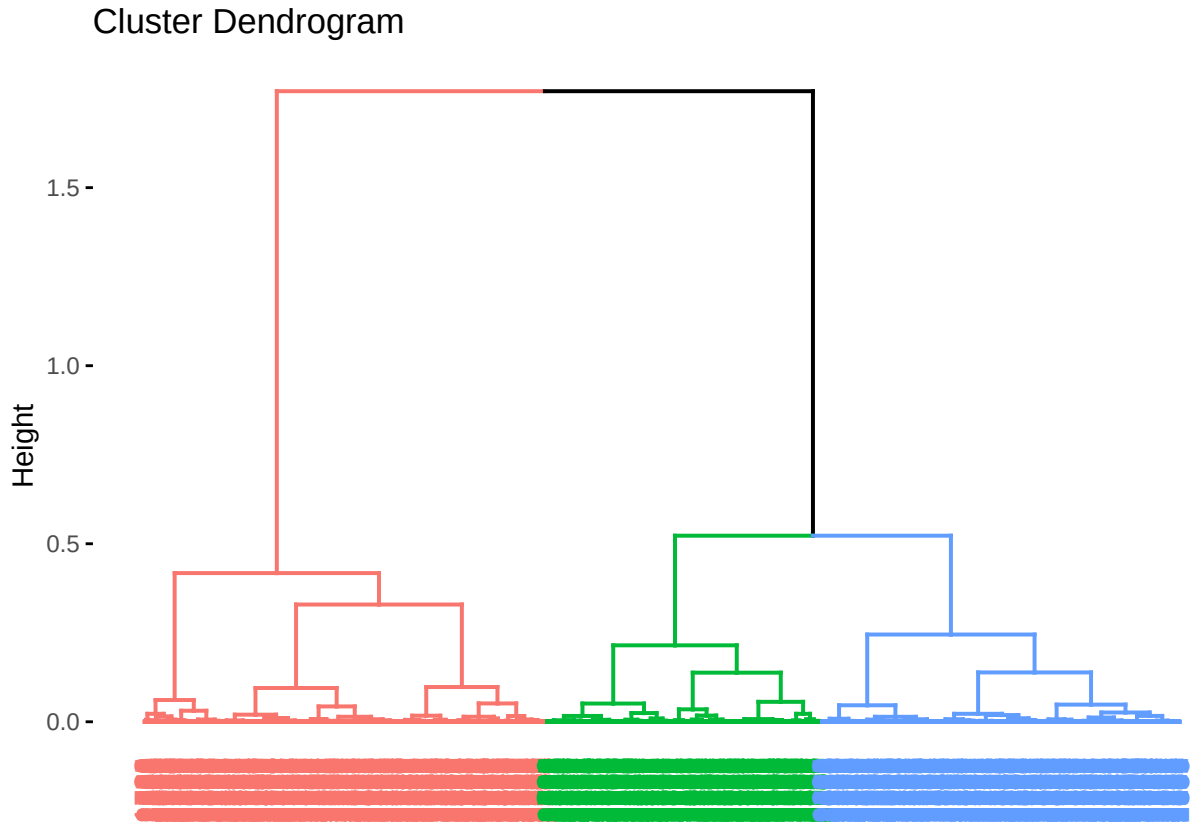

```
plot(hcpc, choice="tree")
```



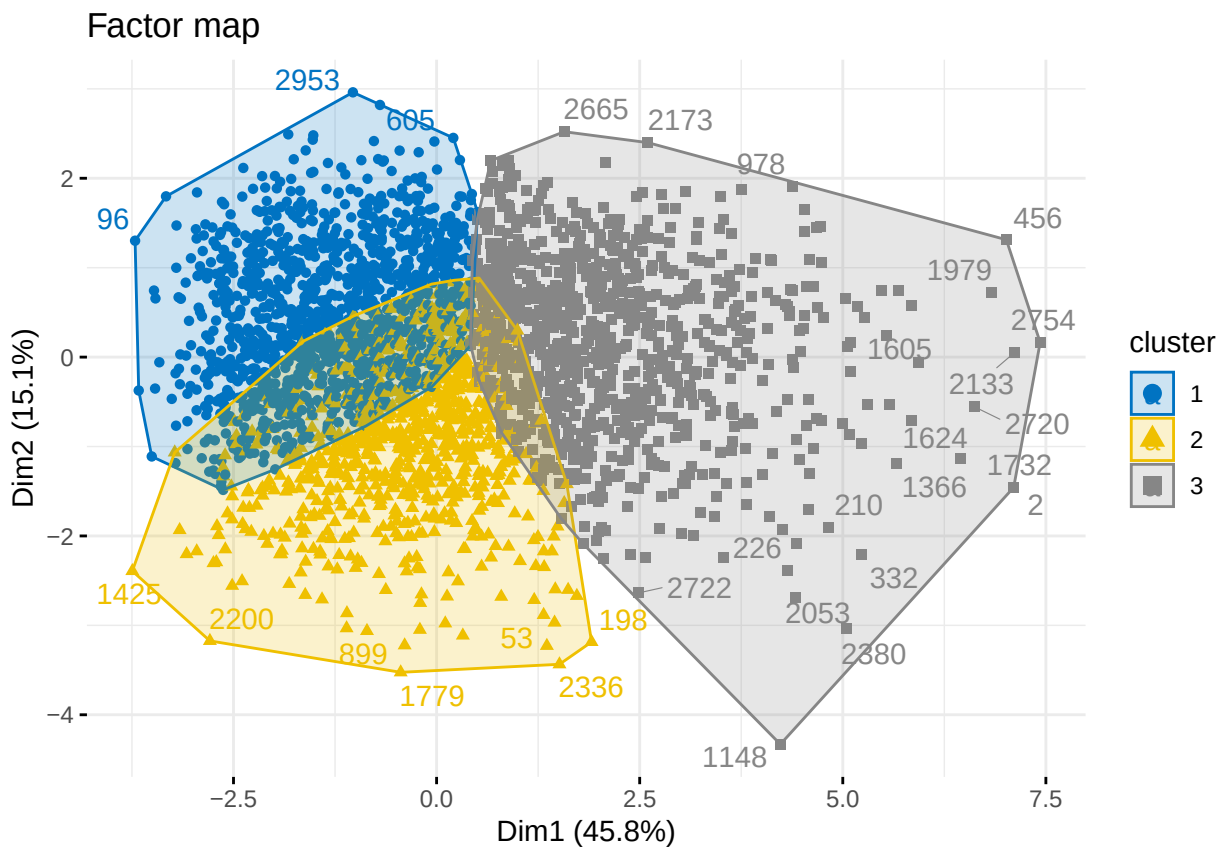

```
plot(hcpc, choice="bar")
```




```
fviz_dend(res.hcpc,repel=TRUE)
```




```
# to visualize individuals on the principal component map and to color individuals according to the clu

fviz_cluster(res.hcpc,
             repel = TRUE, # Avoid label overlapping
             show.clust.cent = TRUE, # Show cluster centers
             palette = "jco", # Color palette see ?ggpubr::ggpar
             ggtheme = theme_minimal(),
             main = "Factor map"
)
```




```
#check which variables are associated to which clusters
list_t <- (res.hcpc$desc.var$quanti)
```

```
list_t
```

```
## $`1`
##                    v.test Mean in category  Overall mean sd in category
## length.prof       8.101861        0.1722884 -1.221870e-16      1.0165047
## n.updates.photo -13.568300       -0.2885339 -1.281938e-16      0.9351926
## sent.ana        -13.582462       -0.2888350  4.187680e-17      0.8964901
## n.photos        -23.730587       -0.5046378 -2.109424e-18      0.6332687
## n.matches       -32.352769       -0.6879911 -4.576547e-17      0.4589857
## score           -32.630702       -0.6939014  1.526123e-18      0.3969877
## score_log       -35.844592       -0.7622457  4.439620e-17      0.6799351
##                  Overall sd       p.value
## length.prof       0.9998333  5.412496e-16
## n.updates.photo   0.9998333  6.173693e-42
## sent.ana          0.9998333  5.088672e-42
## n.photos          0.9998333 1.743433e-124
## n.matches         0.9998333 1.268357e-229
## score             0.9998333 1.505299e-233
## score_log         0.9998333 2.232809e-281
##
## $`2`
##                    v.test Mean in category  Overall mean sd in category
```

```
## n.photos         32.946901       1.01217570 -2.109424e-18      0.8529973
## score_log        -2.072623      -0.06367393  4.439620e-17      0.6184131
## n.updates.photo  -5.756546      -0.17684927 -1.281938e-16      0.8908649
## score            -6.864311      -0.21088142  1.526123e-18      0.4897820
## length.prof      -8.230602      -0.25285582 -1.221870e-16      0.9377558
## n.matches        -8.458622      -0.25986092 -4.576547e-17      0.5360738
## sent.ana        -10.040448      -0.30845687  4.187680e-17      0.8964963
##                 Overall sd       p.value
## n.photos         0.9998333 4.685655e-238
## score_log        0.9998333  3.820739e-02
## n.updates.photo  0.9998333  8.585262e-09
## score            0.9998333  6.681277e-12
## length.prof      0.9998333  1.862743e-16
## n.matches        0.9998333  2.705550e-17
## sent.ana         0.9998333  1.012132e-23
##
## $`3`
##                   v.test Mean in category  Overall mean sd in category
## n.matches       42.43251        1.1433090 -4.576547e-17      0.8080795
## score           41.22053        1.1106532  1.526123e-18      0.9195985
## score_log       40.10943        1.0807155  4.439620e-17      0.5305084
## sent.ana        23.95130        0.6453482  4.187680e-17      0.9047566
## n.updates.photo 19.88483        0.5357803 -1.281938e-16      0.9560131
## n.photos        -5.90241       -0.1590356 -2.109424e-18      0.9022894
##                 Overall sd       p.value
## n.matches        0.9998333  0.000000e+00
## score            0.9998333  0.000000e+00
## score_log        0.9998333  0.000000e+00
## sent.ana         0.9998333 8.955483e-127
## n.updates.photo  0.9998333  5.507270e-88
## n.photos         0.9998333  3.582285e-09
```

```
#generate xtable
#xtable(do.call(rbind.data.frame, list_t ))
```