HW 2 DSC 424

Sanket Praveen Patil

2024-02-04

Problem 1:

a) **Why do we use regularized regressions? Give examples of when you would use ridge versus lasso regressions?**

-> We use regularized regressions, such as ridge and lasso regressions, to handle situations where traditional linear regression models may overfit or become unstable due to multicollinearity (high correlation between predictor variables).

Ridge regression: It's useful when we have many correlated predictors and we want to shrink the coefficients towards zero without eliminating them entirely. For example, in a dataset where multiple predictors are highly correlated, ridge regression can help in stabilizing the estimates.

Lasso regression: It's beneficial when we want a sparse model with fewer predictors, as it tends to shrink less important predictors to exactly zero, effectively performing variable selection. For instance, in a dataset with a large number of predictors but only a few are expected to be truly influential, lasso regression can help in identifying and selecting those important predictors.

b) **How would we treat overfitting in a model with too many variables compared to the sample size?**

-> To treat overfitting in a model with too many variables compared to the sample size, we can employ various techniques:

Feature selection: We can use techniques like ridge regression or lasso regression, which penalize the coefficients of less important predictors, effectively performing feature selection and reducing the model complexity.

Cross-validation: We can use techniques like k-fold cross-validation to assess model performance on unseen data. By splitting the data into training and validation sets multiple times and evaluating the model's performance, we can detect and mitigate overfitting.

Regularization: Regularized regression techniques like ridge and lasso regressions can also help in reducing overfitting by penalizing large coefficients.

c) **How do we check the assumptions of linear regression in R? Give an example for how each assumption may be violated.**

-> In R, we can check the assumptions of linear regression using various diagnostic tools and visualizations:

Linearity: We can check for linearity by plotting the observed values against the predicted values from the linear regression model. If the relationship between the predictors and the response variable is not linear, it may violate the assumption of linearity.

Homoscedasticity: We can assess homoscedasticity by plotting the residuals (the differences between observed and predicted values) against the predicted values. If the spread of residuals is consistent across all levels of the predicted values, homoscedasticity is met. Violations may manifest as a funnel shape or patterns in the residual plot.

Independence of residuals: We can check for independence by examining the autocorrelation plot of residuals or using statistical tests like the Durbin-Watson test. If there is a pattern in the autocorrelation plot or if the Durbin-Watson test indicates significant autocorrelation, it suggests violations of independence.

Normality of residuals: We can assess normality by plotting a histogram or a Q-Q plot of the residuals. If the residuals are approximately normally distributed, the assumption is met. Violations may appear as skewed or heavy-tailed distributions in the histogram or deviations from the straight line in the Q-Q plot.

Problem 3:

1) **How are they applying Factoring Analysis?**

-> In this study, researchers wanted to create a tool with the help of Exploratoray Analysis and PCA to measure how good teachers are at using a mix of online and in-person teaching, called hybrid teaching. They have done this in below 5 steps :

   a) Understanding what to measure
   b) Checking if their tool makes sense
   c) Testing the questions
   d) Seeing how the questions fit together
   e) Making sure the tool is reliable

2) **What kind of factor rotation do they use?**

-> Used Promax rotation as a rotation method.

3) **How many factors do they concentrate on in their analysis? How did they arrive at these number of factors?**

-> In their analysis, the researchers identified and retained five factors that represented different aspects of hybrid education competence They arrived at the number of factors through the process of exploratory factor analysis (EFA), which is a statistical technique used to uncover the underlying structure of a set of variables. The researchers conducted exploratory factor analysis (EFA) on the collected data from educators in social and health care, and health sciences fields. During this process, they examined the eigenvalues associated with each factor. Eigenvalues represent the amount of variance explained by each factor. Factors with eigenvalues greater than 1 were considered for retention. After identifying the initial set of factors, the researchers examined the pattern of factor loadings for each variable. Factor loadings indicate the strength and direction of the relationship between variables and factors. They looked for variables that loaded strongly on each factor and interpreted the meaning of these factors based on the variables they comprised. Based on the pattern of factor loadings and the interpretability of the factors, the researchers determined the number of meaningful factors that best represented the dimensions of hybrid education competence. They named and interpreted these factors according to the variables that loaded onto them and the conceptual framework established in the study.

4) Explain the breakdown of the factors and the significance of their names.

-> Competence in planning and resourcing hybrid teaching: This means educators can effectively organize and gather everything they need for teaching both in-person and online at the same time. They're good at making schedules and choosing the right materials for lessons. Technological competence in hybrid teaching: This is about educators being comfortable using technology to teach. They know how to use computers, internet tools, and other gadgets to make sure their lessons run smoothly, whether students are in class or learning remotely. Interaction competence in hybrid teaching: This is about how well educators can encourage students to talk to each other and participate in class, whether they're in person or online. Good interaction means students feel involved and engaged in their learning. Digital pedagogy competence in hybrid teaching: This is about educators knowing how to teach effectively using digital tools. They understand different ways of teaching and testing students online, and they can adapt their teaching style to fit different situations. Ethical competence in hybrid teaching: This is about educators making sure they do the right thing when teaching online. They respect students' privacy, make sure everyone has a fair chance to learn, and follow rules for online behavior.

5) How do they evaluate the stability of the components (i.e., factorability)?

-> To evaluate the stability of the components, or factorability, the researchers used two main criteria:

a) Kaiser-Meyer-Olkin (KMO) Measure: This measure assesses the sampling adequacy for factor analysis. It indicates whether the variables in the dataset are suitable for factor analysis. A KMO value closer to 1 indicates better suitability, with values above 0.5 generally considered acceptable.

b) Bartlett's test of sphericity was used in order to assume factorability of correlation matrix.

6) Do they use these factors in later analysis, such as regression? If so, what do they discover?

-> Yes, the factors like planning lessons, using technology, interacting with students, teaching online skills, and being fair were important for teachers in hybrid classes. They used these factors to understand how good they were at hybrid teaching. They found out that they needed to look at each factor more closely to really understand how well they were doing. Overall, the study showed that it's important for teachers to be good at all these things to teach hybrid classes well.

7) What overall conclusions does Factor Analysis allow them to draw?

-> Factor Analysis helped researchers understand what skills are important for educators who teach both online and in-person. They created a tool called HybridEduCom to measure these skills, like planning lessons, using technology, interacting with students, and being ethical. They tested this tool with 206 educators and found it reliable. This means it can be used to improve teaching and training programs. Overall, the study shows how important it is for educators to be skilled in both online and traditional teaching methods, especially in today's digital world.

Problem 4:

Load necessary libraries

```
library(DescTools)

## Warning: package 'DescTools' was built under R version 4.3.2

library(Hmisc) #Describe Function

## Warning: package 'Hmisc' was built under R version 4.3.2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:DescTools':
##
##     %nin%, Label, Mean, Quantile

## The following objects are masked from 'package:base':
##
##     format.pval, units

library(psych) #Multiple Functions for Statistics and Multivariate Analysis
```

```
##
## Attaching package: 'psych'

## The following object is masked from 'package:Hmisc':
##
##     describe

## The following objects are masked from 'package:DescTools':
##
##     AUC, ICC, SD

library(GGally) #ggpairs Function

## Warning: package 'GGally' was built under R version 4.3.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.3.2

##
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':
##
##     %+%, alpha

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

library(ggplot2) #ggplot2 Functions
library(vioplot) #Violin Plot Function

## Warning: package 'vioplot' was built under R version 4.3.2

## Loading required package: sm

## Warning: package 'sm' was built under R version 4.3.2

## Package 'sm', version 2.2-5.7: type help(sm) for summary information

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 4.3.2

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

library(corrplot) #Plot Correlations
```

```
## Warning: package 'corrplot' was built under R version 4.3.2

## corrplot 0.92 loaded

library(REdaS) #Bartletts Test of Sphericity

## Warning: package 'REdaS' was built under R version 4.3.2

## Loading required package: grid

library(psych) #PCA/FA functions
library(factoextra) #PCA Visualizations

## Warning: package 'factoextra' was built under R version 4.3.2

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

library("FactoMineR") #PCA functions

## Warning: package 'FactoMineR' was built under R version 4.3.2

library(ade4) #PCA Visualizations

## Warning: package 'ade4' was built under R version 4.3.2

##
## Attaching package: 'ade4'

## The following object is masked from 'package:FactoMineR':
##
##     reconst
```

Read the CSV file

```
df <-
read.csv("D:/Assignments_Depaul/DSC_424_Advance_Data_Analysis/HW2/BIG5.csv",
header = TRUE)
dim(df)

## [1] 19719    50

names(df)

##  [1] "E1"  "E2"  "E3"  "E4"  "E5"  "E6"  "E7"  "E8"  "E9"  "E10" "N1"
"N2"
## [13] "N3"  "N4"  "N5"  "N6"  "N7"  "N8"  "N9"  "N10" "A1"  "A2"  "A3"
"A4"
## [25] "A5"  "A6"  "A7"  "A8"  "A9"  "A10" "C1"  "C2"  "C3"  "C4"  "C5"
"C6"
## [37] "C7"  "C8"  "C9"  "C10" "O1"  "O2"  "O3"  "O4"  "O5"  "O6"  "O7"
"O8"
## [49] "O9"  "O10"
```

checking if data has NA values

```r
sum(is.na(df))

## [1] 0
```

Checking the structure of data

```r
str(df)

## 'data.frame':    19719 obs. of  50 variables:
##  $ E1 : int  4 2 5 2 3 1 5 4 3 1 ...
##  $ E2 : int  2 2 1 5 1 5 1 3 1 4 ...
##  $ E3 : int  5 3 1 2 3 2 5 5 5 2 ...
##  $ E4 : int  2 3 4 4 3 4 1 3 1 5 ...
##  $ E5 : int  5 3 5 3 3 1 5 5 5 2 ...
##  $ E6 : int  1 3 1 4 1 3 1 1 1 4 ...
##  $ E7 : int  4 1 1 3 3 2 5 4 5 1 ...
##  $ E8 : int  3 5 5 4 1 4 4 3 2 4 ...
##  $ E9 : int  5 1 5 4 3 1 4 4 5 1 ...
##  $ E10: int  1 5 1 5 5 5 1 3 3 5 ...
##  $ N1 : int  1 2 5 5 3 1 2 1 2 5 ...
##  $ N2 : int  5 3 1 4 3 5 4 4 4 2 ...
##  $ N3 : int  2 4 5 4 3 4 2 4 5 5 ...
##  $ N4 : int  5 2 5 2 4 5 4 4 3 2 ...
##  $ N5 : int  1 3 5 4 3 1 2 1 3 3 ...
##  $ N6 : int  1 4 5 5 3 4 2 1 5 4 ...
##  $ N7 : int  1 3 5 5 3 4 3 1 5 3 ...
##  $ N8 : int  1 2 5 5 3 1 2 1 4 2 ...
##  $ N9 : int  1 2 5 4 3 5 2 1 3 3 ...
##  $ N10: int  1 4 5 5 4 2 2 1 3 4 ...
##  $ A1 : int  1 1 5 2 5 2 5 2 1 2 ...
##  $ A2 : int  5 3 1 5 5 2 5 5 5 3 ...
##  $ A3 : int  1 3 5 4 3 3 1 1 1 1 ...
##  $ A4 : int  5 4 5 4 5 4 5 4 5 4 ...
##  $ A5 : int  2 4 1 3 1 3 1 3 1 2 ...
##  $ A6 : int  3 4 5 5 5 4 5 3 5 4 ...
##  $ A7 : int  1 2 1 3 1 3 1 1 1 3 ...
##  $ A8 : int  5 3 5 4 5 5 5 3 5 3 ...
##  $ A9 : int  4 4 5 4 5 5 4 4 5 3 ...
##  $ A10: int  5 3 5 3 5 3 5 5 4 2 ...
##  $ C1 : int  4 4 4 3 3 2 2 4 4 5 ...
##  $ C2 : int  1 1 1 3 1 5 4 2 3 2 ...
##  $ C3 : int  5 3 5 4 5 4 3 5 5 4 ...
##  $ C4 : int  1 2 1 5 3 3 3 1 2 2 ...
##  $ C5 : int  5 3 5 1 3 3 3 4 5 3 ...
##  $ C6 : int  1 1 1 4 1 4 3 1 2 2 ...
##  $ C7 : int  4 5 5 5 1 5 3 4 5 4 ...
##  $ C8 : int  1 1 1 4 3 3 3 1 2 2 ...
##  $ C9 : int  4 4 5 2 3 5 3 3 4 4 ...
```

```
##  $ C10: int  5 4 5 3 3 3 3 5 3 4 ...
##  $ O1 : int  4 3 4 4 3 4 3 3 3 4 ...
##  $ O2 : int  1 3 5 3 1 2 1 1 3 2 ...
##  $ O3 : int  3 3 5 5 1 1 5 5 5 5 ...
##  $ O4 : int  1 3 1 2 1 3 1 1 3 2 ...
##  $ O5 : int  5 2 5 4 3 3 4 4 5 4 ...
##  $ O6 : int  1 3 1 2 1 5 1 1 1 1 ...
##  $ O7 : int  4 3 5 5 3 5 4 5 5 4 ...
##  $ O8 : int  2 1 5 2 1 4 3 3 3 3 ...
##  $ O9 : int  5 3 5 5 5 5 3 2 4 4 ...
##  $ O10: int  5 2 5 5 3 3 4 5 5 4 ...
```

```
summary(df$E1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.000   3.000   2.629   4.000   5.000
```

Checking the corrplot matrix

```
cor_matrix <- cor(df)

library(caret)

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following objects are masked from 'package:DescTools':
##
##     MAE, RMSE

highly_correlated_vars <- findCorrelation(cor_matrix, cutoff = 0.75)
colnames(df[highly_correlated_vars])

## [1] "N8"
```

As we have high correlation between variables N8 and N7, we will remove one of the variable

```
df <- df[, -highly_correlated_vars]
dim(df)

## [1] 19719    49
```

PCA_Plot functions

```
PCA_Plot = function(pcaData)
{
  library(ggplot2)
```

```
  theta = seq(0,2*pi,length.out = 100)
  circle = data.frame(x = cos(theta), y = sin(theta))
  p = ggplot(circle,aes(x,y)) + geom_path()

  loadings = data.frame(pcaData$rotation, .names =
row.names(pcaData$rotation))
  p + geom_text(data=loadings, mapping=aes(x = PC1, y = PC2, label = .names,
colour = .names, fontface="bold")) +
    coord_fixed(ratio=1) + labs(x = "PC1", y = "PC2")
}

PCA_Plot_Secondary = function(pcaData)
{
  library(ggplot2)

  theta = seq(0,2*pi,length.out = 100)
  circle = data.frame(x = cos(theta), y = sin(theta))
  p = ggplot(circle,aes(x,y)) + geom_path()

  loadings = data.frame(pcaData$rotation, .names =
row.names(pcaData$rotation))
  p + geom_text(data=loadings, mapping=aes(x = PC3, y = PC4, label = .names,
colour = .names, fontface="bold")) +
    coord_fixed(ratio=1) + labs(x = "PC3", y = "PC4")
}

PCA_Plot_Psyc = function(pcaData)
{
  library(ggplot2)

  theta = seq(0,2*pi,length.out = 100)
  circle = data.frame(x = cos(theta), y = sin(theta))
  p = ggplot(circle,aes(x,y)) + geom_path()

  loadings = as.data.frame(unclass(pcaData$loadings))
  s = rep(0, ncol(loadings))
  for (i in 1:ncol(loadings))
  {
    s[i] = 0
    for (j in 1:nrow(loadings))
      s[i] = s[i] + loadings[j, i]^2
    s[i] = sqrt(s[i])
  }

  for (i in 1:ncol(loadings))
    loadings[, i] = loadings[, i] / s[i]

  loadings$.names = row.names(loadings)
```

```
  p + geom_text(data=loadings, mapping=aes(x = PC1, y = PC2, label = .names,
colour = .names, fontface="bold")) +
    coord_fixed(ratio=1) + labs(x = "PC1", y = "PC2")
}

PCA_Plot_Psyc_Secondary = function(pcaData)
{
  library(ggplot2)

  theta = seq(0,2*pi,length.out = 100)
  circle = data.frame(x = cos(theta), y = sin(theta))
  p = ggplot(circle,aes(x,y)) + geom_path()

  loadings = as.data.frame(unclass(pcaData$loadings))
  s = rep(0, ncol(loadings))
  for (i in 1:ncol(loadings))
  {
    s[i] = 0
    for (j in 1:nrow(loadings))
      s[i] = s[i] + loadings[j, i]^2
    s[i] = sqrt(s[i])
  }

  for (i in 1:ncol(loadings))
    loadings[, i] = loadings[, i] / s[i]

  loadings$.names = row.names(loadings)

  print(loadings)
  p + geom_text(data=loadings, mapping=aes(x = PC3, y = PC4, label = .names,
colour = .names, fontface="bold")) +
    coord_fixed(ratio=1) + labs(x = "PC3", y = "PC4")
}
```

PCA / FA

Test KMO Sampling Adequancy

```
library(psych)
KMO(df)

## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = df)
## Overall MSA =  0.91
## MSA for each item =
##   E1   E2   E3   E4   E5   E6   E7   E8   E9  E10   N1   N2   N3   N4   N5
N6
```

```
## 0.94 0.93 0.96 0.95 0.95 0.94 0.94 0.90 0.92 0.95 0.92 0.90 0.91 0.89 0.95
0.91
##   N7   N9  N10   A1   A2   A3   A4   A5   A6   A7   A8   A9  A10   C1   C2
C3
## 0.93 0.91 0.93 0.90 0.94 0.90 0.89 0.92 0.90 0.91 0.95 0.90 0.96 0.92 0.86
0.90
##   C4   C5   C6   C7   C8   C9  C10   O1   O2   O3   O4   O5   O6   O7   O8
O9
## 0.93 0.91 0.88 0.89 0.94 0.89 0.91 0.77 0.84 0.80 0.81 0.86 0.83 0.91 0.75
0.90
##  O10
## 0.85
```

- The overall MSA is 0.91, which suggests that your dataset is suitable for factor analysis.
- Additionally, the MSA for each individual item is generally high, with most variables having an MSA above 0.8, indicating that each variable contributes adequately to the factor analysis.

Test Bartlett's test of Sphericity

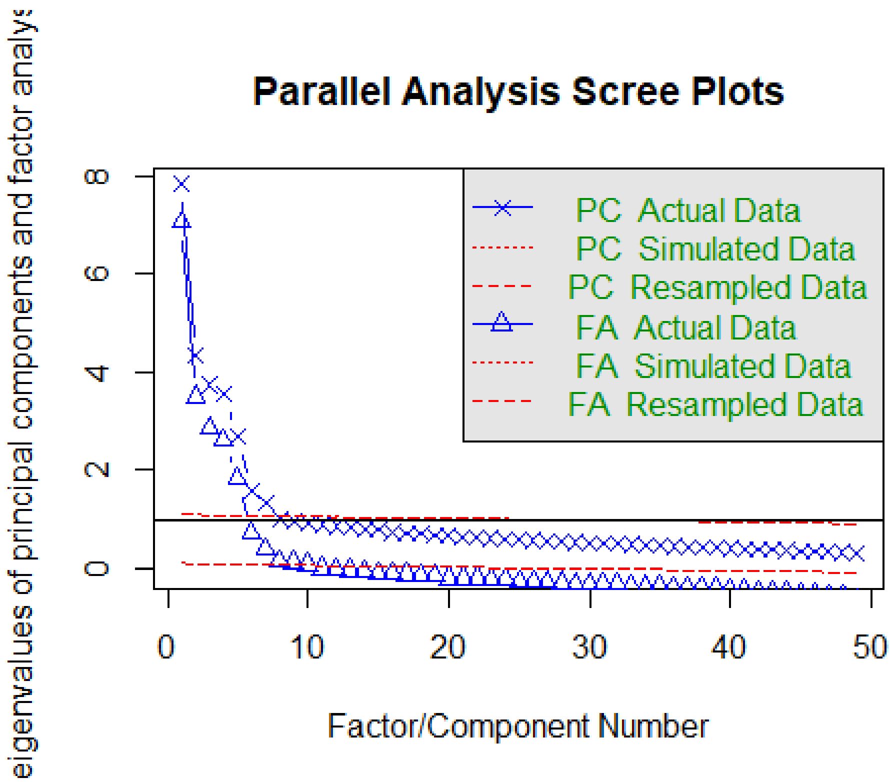
```
library(REdaS)
bart_spher(df)
```

```
##  Bartlett's Test of Sphericity
##
## Call: bart_spher(x = df)
##
##      X2 = 355580.866
##      df = 1176
## p-value < 2.22e-16
```

- The small p-value (< 2.22e-16) suggests that the observed correlation matrix is significantly different from the identity matrix, providing evidence against the null hypothesis of sphericity.

Parallel Analysis (Horn's parallel analysis)

```
comp <- fa.parallel(df)
```




```
## Parallel analysis suggests that the number of factors =  10  and the
number of components =  7

comp

## Call: fa.parallel(x = df)
## Parallel analysis suggests that the number of factors =  10  and the
number of components =  7
##
##  Eigen Values of
##    Original factors Resampled data Simulated data Original components
## 1              7.08           0.10           0.10                7.84
## 2              3.50           0.09           0.09                4.35
## 3              2.87           0.08           0.08                3.75
## 4              2.62           0.08           0.08                3.55
## 5              1.82           0.07           0.07                2.69
## 6              0.71           0.07           0.07                1.58
## 7              0.38           0.06           0.06                1.32
## 8              0.14           0.06           0.06                1.01
## 9              0.11           0.06           0.06                0.96
## 10             0.08           0.05           0.05                0.92
##    Resampled components Simulated components
## 1                  1.10                 1.10
## 2                  1.09                 1.09
## 3                  1.08                 1.08
## 4                  1.07                 1.07
## 5                  1.07                 1.07
```

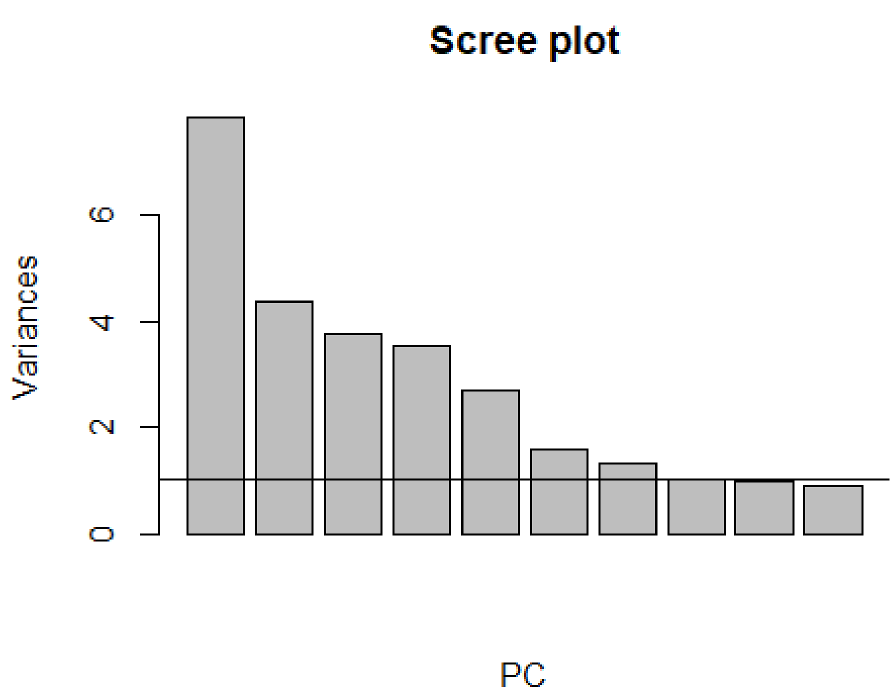
```
## 6                      1.06                     1.07
## 7                      1.06                     1.06
## 8                      1.06                     1.06
## 9                      1.05                     1.05
## 10                     1.05                     1.05
```

—————————————— Create PCA

```
PCA = prcomp(df, center = T, scale = T)
```

Checking the scree plot

```
plot(PCA, main="Scree plot", xlab="PC")
abline(1,0)
```

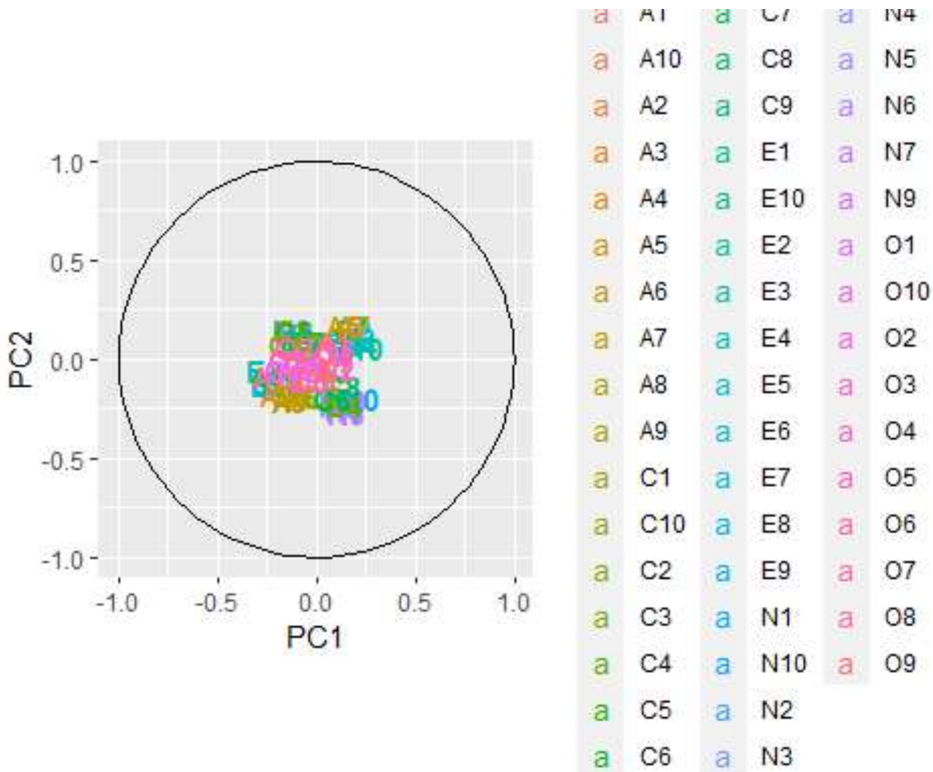



```
summary(PCA)

## Importance of components:
##                          PC1     PC2     PC3     PC4     PC5     PC6
PC7
## Standard deviation     2.80 2.08666 1.93594 1.88371 1.63984 1.25626
1.15086
## Proportion of Variance 0.16 0.08886 0.07649 0.07242 0.05488 0.03221
0.02703
## Cumulative Proportion  0.16 0.24883 0.32531 0.39773 0.45261 0.48482
0.51185
```

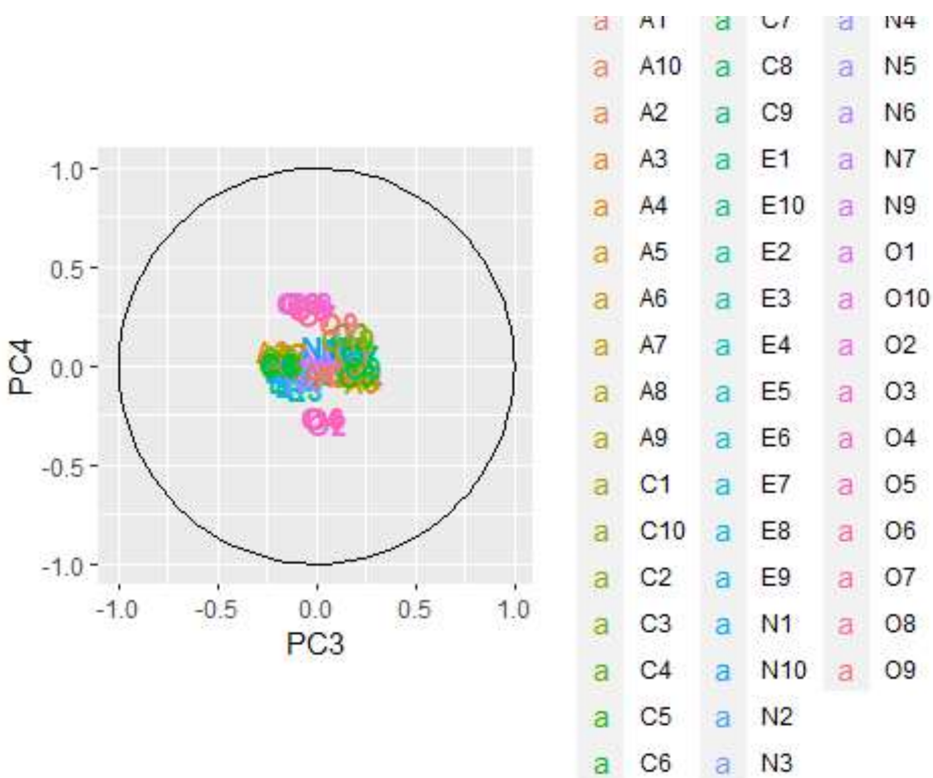
```
##                               PC8     PC9    PC10    PC11    PC12    PC13
PC14
## Standard deviation     1.00257 0.98170 0.95752 0.94676 0.92483 0.90711
0.8964
## Proportion of Variance 0.02051 0.01967 0.01871 0.01829 0.01746 0.01679
0.0164
## Cumulative Proportion  0.53236 0.55203 0.57074 0.58903 0.60649 0.62328
0.6397
##                           PC15    PC16   PC17    PC18    PC19    PC20
PC21
## Standard deviation     0.88390 0.85605 0.8488 0.83972 0.81483 0.81350
0.79710
## Proportion of Variance 0.01594 0.01496 0.0147 0.01439 0.01355 0.01351
0.01297
## Cumulative Proportion  0.65562 0.67058 0.6853 0.69967 0.71322 0.72673
0.73970
##                          PC22    PC23    PC24    PC25    PC26    PC27
PC28
## Standard deviation     0.7887 0.77848 0.76522 0.75968 0.75213 0.74286
0.73162
## Proportion of Variance 0.0127 0.01237 0.01195 0.01178 0.01154 0.01126
0.01092
## Cumulative Proportion  0.7524 0.76476 0.77671 0.78849 0.80003 0.81129
0.82222
##                           PC29    PC30    PC31    PC32    PC33    PC34
PC35
## Standard deviation     0.72390 0.70963 0.70885 0.69915 0.69799 0.68716
0.66921
## Proportion of Variance 0.01069 0.01028 0.01025 0.00998 0.00994 0.00964
0.00914
## Cumulative Proportion  0.83291 0.84319 0.85344 0.86342 0.87336 0.88300
0.89214
##                           PC36    PC37    PC38    PC39    PC40   PC41
PC42
## Standard deviation     0.66799 0.65960 0.64937 0.64482 0.63508 0.6301
0.61614
## Proportion of Variance 0.00911 0.00888 0.00861 0.00849 0.00823 0.0081
0.00775
## Cumulative Proportion  0.90125 0.91012 0.91873 0.92722 0.93545 0.9435
0.95130
##                           PC43    PC44   PC45    PC46    PC47    PC48
PC49
## Standard deviation     0.60971 0.60295 0.5896 0.58605 0.57028 0.56833
0.55842
## Proportion of Variance 0.00759 0.00742 0.0071 0.00701 0.00664 0.00659
0.00636
## Cumulative Proportion  0.95888 0.96630 0.9734 0.98041 0.98704 0.99364
1.00000
```

#Check PCA visualizations

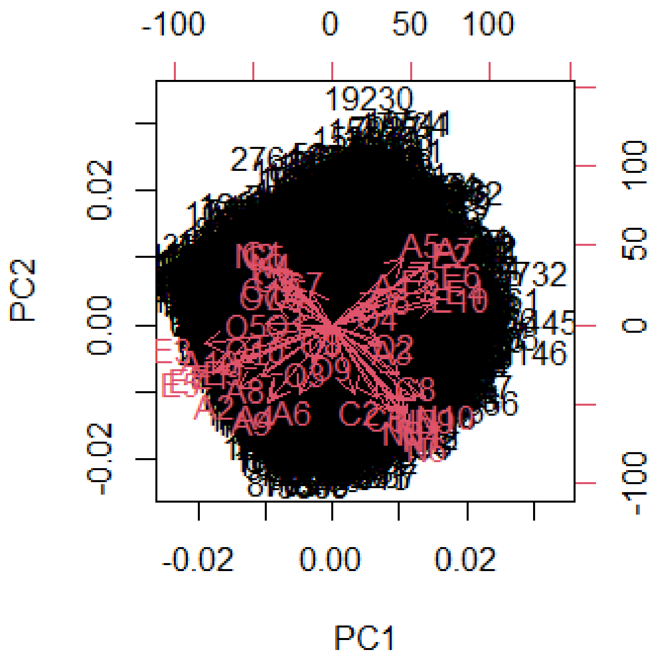
```r
PCA_Plot(PCA) #PCA_plot1
```




```r
PCA_Plot_Secondary(PCA) #PCA_Plot2
```



```r
biplot(PCA) #Biplot
```



Running PCA again after removing the irrelevant variables

```r
PCA2 = psych::principal(df, rotate="varimax", nfactors=7, scores=TRUE)
print(PCA2$loadings, cutoff=.4, sort=T)

##
## Loadings:
##     RC1    RC2    RC5    RC3    RC4    RC7    RC6
## E1   0.690
## E2  -0.736
## E3   0.653
## E4  -0.754
## E5   0.741
## E6  -0.636
## E7   0.743
## E8  -0.628
## E9   0.653
## E10 -0.698
## N1          0.750
## N2         -0.595
## N3          0.684
## N5          0.596
## N6          0.772
## N7          0.684
## N9          0.739
```

```
## N10         0.636
## A2                 0.582
## A4                 0.807
## A5                -0.687
## A6                 0.658
## A7                -0.644
## A8                 0.654
## A9                 0.737
## C1                        0.642
## C2                       -0.607
## C4                       -0.605
## C5                        0.671
## C6                       -0.657
## C7                        0.601
## C8                       -0.533
## C9                        0.672
## C10                       0.519
## O3                               0.732
## O5                               0.578
## O6                              -0.765
## O10                              0.697
## O1                                      0.753
## O8                                      0.760
## N4                                             0.402
## A1                -0.473
## A3                -0.437
## A10                0.443
## C3                        0.453
## O2                              -0.426
## O4                              -0.444        0.423
## O7                                      0.499
## O9
##
##                   RC1   RC2   RC5   RC3   RC4   RC7   RC6
## SS loadings    5.496 4.609 4.317 3.911 2.684 2.438 1.625
## Proportion Var 0.112 0.094 0.088 0.080 0.055 0.050 0.033
## Cumulative Var 0.112 0.206 0.294 0.374 0.429 0.479 0.512
```

```
ls(PCA2)
```

```
##  [1] "Call"          "chi"           "communality"  "complexity"
"criteria"
##  [6] "dof"           "EPVAL"         "factors"      "fit"           "fit.off"
## [11] "fn"            "loadings"      "n.obs"        "null.dof"
"null.model"
## [16] "objective"     "PVAL"          "r.scores"     "R2"
"residual"
## [21] "rms"           "rot.mat"       "rotation"     "scores"
"STATISTIC"
```

```
## [26] "Structure"    "uniquenesses" "Vaccounted"   "valid"        "values"
## [31] "weights"
```

Checking eigen vlues

```
PCA2$values

##  [1] 7.8383695 4.3541466 3.7478700 3.5483684 2.6890906 1.5781788 1.3244895
##  [8] 1.0051468 0.9637321 0.9168389 0.8963499 0.8553160 0.8228449 0.8035898
## [15] 0.7812767 0.7328240 0.7204581 0.7051375 0.6639407 0.6617878 0.6353625
## [22] 0.6220714 0.6060383 0.5855604 0.5771212 0.5656941 0.5518451 0.5352635
## [29] 0.5240321 0.5035677 0.5024676 0.4888155 0.4871875 0.4721858 0.4478460
## [36] 0.4462173 0.4350674 0.4216820 0.4157906 0.4033256 0.3970326 0.3796229
## [43] 0.3717471 0.3635453 0.3476576 0.3434540 0.3252142 0.3229965 0.3118317
```

To select number of components

```
table(PCA2$values > 1)

##
## FALSE  TRUE
##    41     8

eigenvalues <- PCA2$values
```

Calculate the cumulative sum of eigenvalues

```
cumulative_variance <- cumsum(eigenvalues) / sum(eigenvalues)
```

Find the number of components needed to explain 100% of the variance

```
num_components_100 <- which.max(cumulative_variance >= 1)
```

Print the result

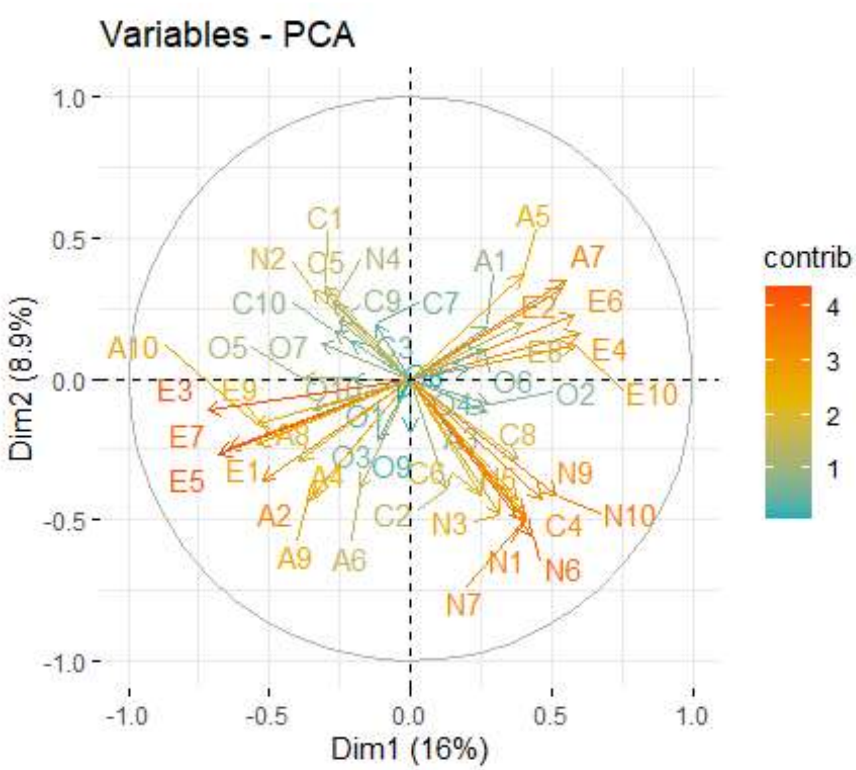
```
cat("Number of components to explain 100% of the variance:",
num_components_100, "\n")

## Number of components to explain 100% of the variance: 49

desired_variance_explained <- 0.95
num_components_desired <- which.max(cumulative_variance >=
desired_variance_explained)
cat("Number of components to explain 95% of the variance:",
num_components_desired, "\n")

## Number of components to explain 95% of the variance: 42
```

Question A:

- With the help of above information, we can see we required all 49 components to explain 100% of the variance.
- With the help of eigen values method, we can see 7 components have eigen values greater than 1.
- Hence 7 principle components were selected by scree plot.
- Number of components to explain 95% of the variance: 42
- We will go with number of components as 7 as more components increase model complexity, which may lead to overfitting, especially if we have limited data.

PCA Variables

```
pca_var<-fviz_pca_var(PCA,
                      col.var = "contrib", # Color by contributions to the PC
                      gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
                      repel = TRUE     # Avoid text overlapping
)
pca_var
```



————————-

Formula for PCA

Extract loadings for the first component

```
loadings_component1 <- PCA2$loadings[, 1]
loadings_component1

##          E1           E2           E3           E4           E5
E6
##  0.690045165 -0.736196366  0.653425797 -0.754384085  0.741191547 -
0.635667863
##          E7           E8           E9          E10           N1
N2
##  0.743373502 -0.628183613  0.652935923 -0.697650766 -0.095558041
0.063648008
##          N3           N4           N5           N6           N7
N9
## -0.133566179  0.111606257 -0.056885324 -0.070458486 -0.028862935 -
0.050704174
##         N10           A1           A2           A3           A4
A5
## -0.248997732 -0.043448309  0.351899078  0.122095479  0.023663682 -
0.155321393
##          A6           A7           A8           A9          A10
C1
## -0.036896716 -0.337645206  0.104640099  0.094105202  0.319517510
0.036618155
##          C2           C3           C4           C5           C6
C7
##  0.037404637 -0.054832522 -0.078295554  0.073806036 -0.024155779 -
0.043933440
##          C8           C9          C10           O1           O2
O3
## -0.081269727  0.055114996  0.018010669  0.058478223 -0.042731494
0.015369011
##          O4           O5           O6           O7           O8
O9
## -0.007984566  0.198047521 -0.099062554  0.068466509  0.021271903 -
0.160943113
##         O10
##  0.182947921
```

Question B:

- Compute the formula for the first component
- PC1 = 0.690 * E1 - 0.736 * E2 + 0.653 * E3 - 0.754 * E4 + 0.741 * E5 - 0.635 * E6 + 0.743 * E7 - 0.628 * E8 + 0.652 * E9 - 0.698 * E10 - 0.096 * N1 + 0.064 * N2 - 0.134 * N3 + 0.112 * N4 - 0.057 * N5 - 0.070 * N6 - 0.029 * N7 - 0.051 * N9 - 0.249 * N10 - 0.043 * A1 + 0.352 * A2 + 0.122 * A3 + 0.024 * A4 - 0.155 * A5 - 0.037 * A6 - 0.338 * A7 + 0.105 * A8 + 0.094 * A9 + 0.320 * A10 + 0.037 * C1 - 0.055 * C2 - 0.078 * C3 + 0.074 * C4 - 0.024 * C5 - 0.044 * C6 - 0.081 * C7 + 0.055 * C8 + 0.018 * C9 + 0.058 * C10 + 0.015 * O1 - 0.043 * O2 + 0.015 * O3 - 0.008 * O4 + 0.198 * O5 - 0.099 * O6 + 0.068 * O7 + 0.021 * O8 - 0.161 * O9 + 0.183 * O10
- After rotating the components, each component represents a linear combination of the original variables in such a way that the first component captures the maximum amount of variance in the data. This means that the first component contains the most information compared to any other component. Rotating the components ensures that each subsequent component captures the maximum remaining variance orthogonal to the previous components.
- The names of the components will be as follows

  Social Butterfly & Emotional Stability: How much you enjoy socializing and how well you handle stress and emotions.

  Sensitive & Moody: How easily you get upset and your tendency to experience mood swings.

  Empathetic & Caring: How much you care about others’ feelings and emotions.

  Organized & Responsible: How well you plan ahead and take care of your duties.

  Curious & Creative: How interested you are in new ideas and how imaginative you can be.

  Outgoing & Talkative: How much you enjoy talking to people and engaging in social activities.

  Efficient & Disciplined: How well you manage tasks and stick to routines and rules.

Question C:

—————— Calculating the scores

```
scores <- PCA2$scores
```

Calculating the principle components

```
for (i in 1:7) {
  # Get the column index for the current component
  component_col <- scores[, i]

  # Find the index of the subject with the highest score
  max_index <- which.max(component_col)

  # Find the index of the subject with the lowest score
  min_index <- which.min(component_col)

  # Print the results
  cat("Principal Component", i, ":\n")
  cat("Subject with the highest score:", "Index:", max_index, "Score:",
component_col[max_index], "\n")
  cat("Subject with the lowest score:", "Index:", min_index, "Score:",
component_col[min_index], "\n\n")
}

## Principal Component 1 :
## Subject with the highest score: Index: 4177 Score: 2.888451
## Subject with the lowest score: Index: 629 Score: -2.8107
##
## Principal Component 2 :
## Subject with the highest score: Index: 12089 Score: 2.763371
## Subject with the lowest score: Index: 19065 Score: -3.41609
##
## Principal Component 3 :
## Subject with the highest score: Index: 17760 Score: 2.327989
## Subject with the lowest score: Index: 9864 Score: -4.42771
##
## Principal Component 4 :
## Subject with the highest score: Index: 16225 Score: 2.818386
## Subject with the lowest score: Index: 15237 Score: -3.391097
##
## Principal Component 5 :
## Subject with the highest score: Index: 445 Score: 2.940589
## Subject with the lowest score: Index: 6106 Score: -4.530446
##
## Principal Component 6 :
## Subject with the highest score: Index: 13095 Score: 3.786076
```

```
## Subject with the lowest score: Index: 19065 Score: -5.637476
##
## Principal Component 7 :
## Subject with the highest score: Index: 2794 Score: 7.256328
## Subject with the lowest score: Index: 19065 Score: -10.69626
```

Create an empty data frame to store the scores

```
score_table <- data.frame(
  Subject_Index = numeric(),
  PC1 = numeric(),
  PC2 = numeric(),
  PC3 = numeric(),
  PC4 = numeric(),
  PC5 = numeric(),
  PC6 = numeric(),
  PC7 = numeric(),
  stringsAsFactors = FALSE
)
```

Function to add scores to the table

```
add_scores <- function(subject_index) {
  scores <- c(subject_index, scores[subject_index, ])
  score_table[nrow(score_table) + 1, ] <<- scores
}
```

Add scores for subjects with highest and lowest scores in each component

```
add_scores(4177)  # Highest score in PC1
add_scores(629)   # Lowest score in PC1
add_scores(12089) # Highest score in PC2
add_scores(19065) # Lowest score in PC2
add_scores(17760) # Highest score in PC3
add_scores(9864)  # Lowest score in PC3
add_scores(16225) # Highest score in PC4
add_scores(15237) # Lowest score in PC4
add_scores(445)   # Highest score in PC5
add_scores(6106)  # Lowest score in PC5
add_scores(13095) # Highest score in PC6
add_scores(19065) # Lowest score in PC6
add_scores(2794)  # Highest score in PC7
add_scores(19065) # Lowest score in PC7
```

Principal component scores for each subject

```
score_table
```

```
##    Subject_Index         PC1        PC2        PC3         PC4         PC5
## 1           4177  2.88845145  1.2276324 -4.2731038 -0.33845516  0.02214885
## 2            629 -2.81070007 -0.4277177 -0.2625800  2.24634745  1.48373484
## 3          12089  1.50539797  2.7633709  0.1321301  0.79036263 -0.02239631
## 4          19065  0.82295271 -3.4160903 -3.3699336 -1.89340867 -1.93516780
## 5          17760 -0.54037637 -0.9097200  2.3279893 -1.90736584 -2.71590676
## 6           9864  1.35689615  1.9526249 -4.4277101  0.84966841  0.69214470
## 7          16225  2.70210185  1.7877910 -3.5334901  2.81838589  0.63364772
## 8          15237  1.25215502 -2.0910859  1.1024790 -3.39109736 -0.12259676
## 9            445 -1.30827286  1.1403383 -0.3334187 -0.80812274  2.94058850
## 10          6106  2.03749873  1.3106709 -2.0970520 -0.92052086 -4.53044623
## 11         13095  0.51128892 -0.1667895  0.2983733  0.01662924 -3.23785340
## 12         19065  0.82295271 -3.4160903 -3.3699336 -1.89340867 -1.93516780
## 13          2794 -0.07389689  1.9353093  0.1876769  0.53864266 -0.98774862
## 14         19065  0.82295271 -3.4160903 -3.3699336 -1.89340867 -1.93516780
##           PC6         PC7
## 1   1.0087391   2.8599516
## 2  -0.5471832  -0.7500755
## 3  -2.8113657   0.4493482
## 4  -5.6374759 -10.6962557
## 5  -0.4677366   1.3965752
## 6   1.3167155  -0.1846107
## 7  -2.6256614   1.8378405
## 8  -0.5991815  -2.1571346
## 9  -0.5369472   3.3453047
## 10  0.3671863  -0.6065406
## 11  3.7860762   0.6306628
## 12 -5.6374759 -10.6962557
## 13  2.6313988   7.2563277
## 14 -5.6374759 -10.6962557
```

Question D:

—————————————— Factor Analysis

Conducting factor analysis

```r
fit = factanal(df, 7)
print(fit$loadings, cuttoff=0.4, sort=T)
```

```
##
## Loadings:
##     Factor1 Factor2 Factor3 Factor4 Factor5 Factor6 Factor7
## E1   0.657                                           0.181
## E2  -0.698           -0.112                          0.126
## E3   0.645  -0.265    0.265   0.132                  0.137
## E4  -0.714   0.133                                   0.127
```

```
## E5   0.730           0.217   0.101
## E6  -0.596          -0.139          -0.183  -0.103   0.229
## E7   0.730  -0.111   0.166                           0.120
## E8  -0.553
## E9   0.595                           0.132           0.134
## E10 -0.658   0.174
## N1           0.715
## N2          -0.554                                   0.290
## N3  -0.129   0.635   0.158
## N5           0.541          -0.126                   0.134
## N6           0.746
## N7           0.631          -0.173                   0.101
## N9           0.707  -0.166
## N10 -0.248   0.588          -0.175
## A2   0.347           0.535
## A4                   0.790
## A5  -0.153          -0.652                           0.195
## A6           0.145   0.596                  -0.106   0.136
## A7  -0.331          -0.610                           0.194
## A8   0.124           0.583
## A9                   0.709           0.103           0.100
## C1          -0.109           0.596           0.108
## C2                          -0.542           0.146   0.155
## C4           0.341          -0.572                   0.141
## C5                           0.619                   0.105
## C6           0.150          -0.602                   0.190
## C7                           0.532
## C9                           0.618
## O3                                   0.593
## O5   0.191                   0.171   0.626   0.146   0.170
## O6                                  -0.607           0.104
## O10  0.179                           0.740   0.113
## O1                                   0.302   0.725
## O8                                   0.259   0.732
## N4   0.122  -0.336                                   0.233
## A1                  -0.424                  -0.110   0.266
## A3           0.239  -0.392  -0.204           0.119   0.110
## A10  0.323  -0.144   0.391   0.149   0.121           0.199
## C3                            0.399   0.208   0.129
## C8           0.195  -0.142  -0.487                   0.134
## C10                           0.465   0.174   0.134   0.129
## O2           0.206                  -0.450  -0.247   0.274
## O4           0.108                  -0.424  -0.154   0.293
## O7          -0.157           0.197   0.373   0.300
## O9  -0.131   0.163   0.176           0.267   0.196
##
##                 Factor1 Factor2 Factor3 Factor4 Factor5 Factor6 Factor7
## SS loadings      5.007   4.048   3.755   3.309   2.617   1.480   0.979
## Proportion Var   0.102   0.083   0.077   0.068   0.053   0.030   0.020
## Cumulative Var   0.102   0.185   0.261   0.329   0.382   0.413   0.433
```

- Looking at the values, we can observe differences in the loadings between the two methods.Check below examples
- For variable E1, in factor analysis, it has a loading of 0.657 on Factor1, while in PCA, it has a loading of 0.690 on RC1.
- For variable A4, in factor analysis, it has a loading of 0.790 on Factor4, while in PCA, it has a loading of 0.807 on RC1.
- For variable O3, in factor analysis, it has a loading of 0.593 on Factor5, while in PCA, it has a loading of 0.732 on RC3.
- Despite these differences, the interpretation of the factors remains somewhat consistent. Both analyses identify similar latent constructs (e.g., Extraversion, Neuroticism, Agreeableness) based on the variables’ loadings on each factor.
- The underlying latent constructs remain similar, allowing for a consistent interpretation of the results in most cases.