Exercise 06 - Principal component analysis (PCA) and Exloratory factor analysis (EFA)

Zoltan Kekecs

November 22, 2021

Contents

1 Abstract

In this exercise we will deal with the "curse of dimensionality". This is a problem which we encounter if we have a lot of predictors/parameters in our regression model. If the number of paramters in the model is too high in comparison with the number of observations in the dataset, there is a large threat for overfitting, and the model coefficients will be less useful for prediction in new data from the target population. In the current exercise we will learn methods for dimensionality reduction, to reduce the number of predictors with as little information loss as possible. We will do this via Principal Component Analysis. Additionally, we will learn a related technique called Exploratory Factor Analysis which uses a similar logic to explore underlying/latent factors governing a set of observed variables. This exercise is partially built on the Factor Analysis course in DataCamp and the UCLA Factor Analsysis guide: https://stats.idre.ucla.edu/spss/seminars/efa-spss/

2 Data management and descriptives

2.1 Load packages

We will work with the following packages in this exercise:

```
library(GGally)  # for ggcorr
library(corrr)  # network_plot
library(ggcorrplot)  # for ggcorrplot
library(FactoMineR)  # multiple PCA functions
library(factoextra)  # visualisation functions for PCA (e.g. fviz_pca_var)
library(paran)  # for paran

library(psych)  # for the mixedCor, cortest.bartlett, KMO, fa functions
library(car)  # for vif
library(GPArotation)  # for the psych fa function to have the required rotation functionalities
library(MVN)  # for mvn function
library(ICS)  # for multivariate skew and kurtosis test
library(tidyverse)  # for tidy code
```

2.2 Load custom functions

This fviz_loadnings_with_cor() is a custom function that will be used in this exercise the visualize some of the results of the principal component analysis and exploratory factor analysis.

```
fviz_loadnings_with_cor <- function(mod, axes = 1, loadings_above = 0.4) {
    require(factoextra)
    require(dplyr)
    require(ggplot2)

    if (!is.na(as.character(mod$call$call)[1])) {
        if (as.character(mod$call$call)[1] == "PCA") {
            contrib_and_cov = as.data.frame(rbind(mod[["var"]][["contrib"]],
                mod[["var"]][["cor"]]))

            vars = rownames(mod[["var"]][["contrib"]])
            attribute_type = rep(c("contribution", "correlation"),
                each = length(vars))
            contrib_and_cov = cbind(contrib_and_cov, attribute_type)
            contrib_and_cov

            plot_data = cbind(as.data.frame(cbind(contrib_and_cov[contrib_and_cov[,
                "attribute_type"] == "contribution", axes], contrib_and_cov[contrib_and_cov[,
```

```
            "attribute_type"] == "correlation", axes])),
            vars)
        names(plot_data) = c("contribution", "correlation",
            "vars")

        plot_data = plot_data %>%
            mutate(correlation = round(correlation, 2))

        plot = plot_data %>%
            ggplot() + aes(x = reorder(vars, contribution),
            y = contribution, gradient = correlation, label = correlation) +
            geom_col(aes(fill = correlation)) + geom_hline(yintercept = mean(plot_data$contribution
            col = "red", lty = "dashed") + scale_fill_gradient2() +
            xlab("variable") + coord_flip() + geom_label(color = "black",
            fontface = "bold", position = position_dodge(0.5))

    }
} else if (!is.na(as.character(mod$Call)[1])) {

    if (as.character(mod$Call)[1] == "fa") {
        loadings_table = mod$loadings %>%
            matrix(ncol = ncol(mod$loadings)) %>%
            as_tibble() %>%
            mutate(variable = mod$loadings %>%
              rownames()) %>%
            gather(factor, loading, -variable) %>%
            mutate(sign = if_else(loading >= 0, "positive",
              "negative"))

        if (!is.null(loadings_above)) {
            loadings_table[abs(loadings_table[, "loading"]) <
              loadings_above, "loading"] = NA
            loadings_table = loadings_table[!is.na(loadings_table[,
              "loading"]), ]
        }

        if (!is.null(axes)) {

            loadings_table = loadings_table %>%
              filter(factor == paste0("V", axes))
        }

        plot = loadings_table %>%
            ggplot() + aes(y = loading %>%
            abs(), x = reorder(variable, abs(loading)), fill = loading,
            label = round(loading, 2)) + geom_col(position = "dodge") +
            scale_fill_gradient2() + coord_flip() + geom_label(color = "black",
            fill = "white", fontface = "bold", position = position_dodge(0.5)) +
            facet_wrap(~factor) + labs(y = "Loading strength",
            x = "Variable")
    }
}
```

```
    return(plot)

}
```

2.3 The Humor Styles Questionnaire

In this exercise we will work with a dataset containing data about people's style of humor. You can download the dataset from this GitHub link: https://github.com/kekecsz/SIMM32/blob/master/2020/Lab_6/Humor%20Style%20Questionnaire.sav (The dataset was originally downloaded from https://openpsychometrics.org/_rawdata/.) This data was collection with an interactive online version of the Humor Styles Questionnaire from Martin, R. A., Puhlik-Doris, P., Larsen, G., Gray, J., & Weir, K. (2003). Individual differences in uses of humor and their relation to psychological well-being: Development of the Humor Styles Questionnaire. Journal of Research in Personality, 37, 48-75. The variables Q1 through Q32 were statements rated on a five point scale where 1=Never or very rarely true, 2=Rarely true, 3= Sometimes true, 4= Often true, 5=Very often or always true (-1=did not select an answer). The exact statements were:

Q1. I usually don't laugh or joke around much with other people. Q2. If I am feeling depressed, I can usually cheer myself up with humor. Q3. If someone makes a mistake, I will often tease them about it. Q4. I let people laugh at me or make fun at my expense more than I should. Q5. I don't have to work very hard at making other people laugh—I seem to be a naturally humorous person. Q6. Even when I'm by myself, I'm often amused by the absurdities of life. Q7. People are never offended or hurt by my sense of humor. Q8. I will often get carried away in putting myself down if it makes my family or friends laugh. Q9. I rarely make other people laugh by telling funny stories about myself. Q10. If I am feeling upset or unhappy I usually try to think of something funny about the situation to make myself feel better. Q11. When telling jokes or saying funny things, I am usually not very concerned about how other people are taking it. Q12. I often try to make people like or accept me more by saying something funny about my own weaknesses, blunders, or faults. Q13. I laugh and joke a lot with my closest friends. Q14. My humorous outlook on life keeps me from getting overly upset or depressed about things. Q15. I do not like it when people use humor as a way of criticizing or putting someone down. Q16. I don't often say funny things to put myself down. Q17. I usually don't like to tell jokes or amuse people. Q18. If I'm by myself and I'm feeling unhappy, I make an effort to think of something funny to cheer myself up. Q19. Sometimes I think of something that is so funny that I can't stop myself from saying it, even if it is not appropriate for the situation. Q20. I often go overboard in putting myself down when I am making jokes or trying to be funny. Q21. I enjoy making people laugh. Q22. If I am feeling sad or upset, I usually lose my sense of humor. Q23. I never participate in laughing at others even if all my friends are doing it. Q24. When I am with friends or family, I often seem to be the one that other people make fun of or joke about. Q25. I don't often joke around with my friends. Q26. It is my experience that thinking about some amusing aspect of a situation is often a very effective way of coping with problems. Q27. If I don't like someone, I often use humor or teasing to put them down. Q28. If I am having problems or feeling unhappy, I often cover it up by joking around, so that even my closest friends don't know how I really feel. Q29. I usually can't think of witty things to say when I'm with other people. Q30. I don't need to be with other people to feel amused – I can usually find things to laugh about even when I'm by myself. Q31. Even if something is really funny to me, I will not laugh or joke about it if someone will be offended. Q32. Letting others laugh at me is my way of keeping my friends and family in good spirits.

On the next page test takers were prompted for some more variables:

age.

gender. chosen from drop down list (1=male, 2=female, 3=other)

accuracy. How accurate they thought their answers were on a scale from 0 to 100, answers were entered as text and parsed to an integer. They were instructed to enter a 0 if they did not want to be included in research.

life_stress. This is a new variable that was only added for the purpose of this exercise. This was not in the original study. This variable was simulated (not real). Let's imagine that participants had to "Rate on

average how stressful do you think your life is (thinking about the bast year) on a scale of 0-7, where 0 means not at all stressful, and 7 means extremely stressful."

There are some missing values in this dataset. For the sake of simplicity, we will just exclude these from the dataset for now.

```
hsq <- read_csv("https://raw.githubusercontent.com/kekecsz/SIMM61-Course-materials/main/Exercise_06%20-%
```

```
## Rows: 1071 Columns: 40
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (40): Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q10, Q11, Q12, Q13, Q14, Q15, ...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
hsq <- hsq %>%
    drop_na()


hsq %>%
    describe()
```

2.4 Checking data

We can explor the basic descriptive statistics first.

```
str(hsq)

summary(hsq)
```

Let's say we would like to determine which are the most important features in humor style that could help us determine life_stress. One way for doing this would be to fit a linear regression model with life_stress as a dependent and Q1-Q32 of the questions as predictors.

When we run this model, based on the significance of the regression equations, it seems that the model is significantly better than the null model, and there are several items that have significant added predictive power to the model, so it is worth taking humor style into account in determining stress.

But if we want to know which aspects of humor style are important, it is difficult to get that from this regression analysis. On the one hand, we cannot have complete confidence in the p-values, because we have run 32 statistical tests, testing the explanatory power of the 32 items, which greatly increase the probability of a type-one error (false positive). In other words, there is a high probability that some of the significantly marked predictors are not related to stress in the population.

Furthermore, it must be taken into account that the predictors are correlated, i.e. certain predictors explain similar responses to the variance of stress, and in this model it is possible that some predictors mask the effect of other correlated predictors.

```
mod_allitems = lm(life_stress ~ Q1 + Q2 + Q3 + Q4 + Q5 + Q6 +
    Q7 + Q8 + Q9 + Q10 + Q11 + Q12 + Q13 + Q14 + Q15 + Q16 +
    Q17 + Q18 + Q19 + Q20 + Q21 + Q22 + Q23 + Q24 + Q25 + Q26 +
    Q27 + Q28 + Q29 + Q30 + Q31 + Q32, data = hsq)

summary(mod_allitems)
```

```
##
## Call:
```

```
## lm(formula = life_stress ~ Q1 + Q2 + Q3 + Q4 + Q5 + Q6 + Q7 +
##     Q8 + Q9 + Q10 + Q11 + Q12 + Q13 + Q14 + Q15 + Q16 + Q17 +
##     Q18 + Q19 + Q20 + Q21 + Q22 + Q23 + Q24 + Q25 + Q26 + Q27 +
##     Q28 + Q29 + Q30 + Q31 + Q32, data = hsq)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3528 -0.8639  0.0307  0.8807  3.8981
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.084097   0.623579   0.135 0.892750
## Q1          -0.097889   0.050921  -1.922 0.054856 .
## Q2           0.007704   0.047341   0.163 0.870764
## Q3           0.030141   0.043687   0.690 0.490402
## Q4           0.170860   0.045619   3.745 0.000191 ***
## Q5          -0.066691   0.053180  -1.254 0.210121
## Q6           0.026375   0.055614   0.474 0.635433
## Q7          -0.023993   0.046146  -0.520 0.603235
## Q8           0.105826   0.050916   2.078 0.037935 *
## Q9          -0.052866   0.039836  -1.327 0.184797
## Q10          0.009067   0.050208   0.181 0.856732
## Q11         -0.015873   0.039779  -0.399 0.689967
## Q12          0.202653   0.044006   4.605 4.68e-06 ***
## Q13          0.026218   0.067533   0.388 0.697937
## Q14          0.052088   0.045885   1.135 0.256577
## Q15          0.015961   0.040418   0.395 0.693010
## Q16          0.163045   0.043615   3.738 0.000196 ***
## Q17          0.027425   0.054744   0.501 0.616506
## Q18          0.010161   0.050137   0.203 0.839442
## Q19          0.034051   0.041819   0.814 0.415709
## Q20          0.208081   0.055749   3.732 0.000201 ***
## Q21         -0.103708   0.066913  -1.550 0.121496
## Q22          0.062369   0.039423   1.582 0.113967
## Q23          0.044103   0.041631   1.059 0.289695
## Q24          0.222733   0.043525   5.117 3.74e-07 ***
## Q25          0.031822   0.070080   0.454 0.649871
## Q26         -0.072567   0.049911  -1.454 0.146287
## Q27         -0.020883   0.038454  -0.543 0.587211
## Q28          0.201723   0.036348   5.550 3.70e-08 ***
## Q29         -0.031136   0.044079  -0.706 0.480130
## Q30          0.006414   0.046809   0.137 0.891043
## Q31         -0.011109   0.042425  -0.262 0.793497
## Q32          0.148716   0.045016   3.304 0.000990 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.302 on 960 degrees of freedom
## Multiple R-squared:  0.3438, Adjusted R-squared:  0.3219
## F-statistic: 15.72 on 32 and 960 DF,  p-value: < 2.2e-16
```

In fact, if we run a correlation matrix of all predictors, we see that almost every predictor variable is correlated significantly with the other predictors. The collinearity diagnostics (VIF) tells us that this multicollinearity is not problematic in our particular model (The VIFs are all below 3), but we can imagine that the correlation

of the variables could lead to issues with multicollinearity as well in another similar study.

```
hsq_items_only = hsq %>%
    select(Q1:Q32)

cor = hsq_items_only %>%
    cor()

cor

vif(mod_allitems)
```

2.5 The curse of dimensionality

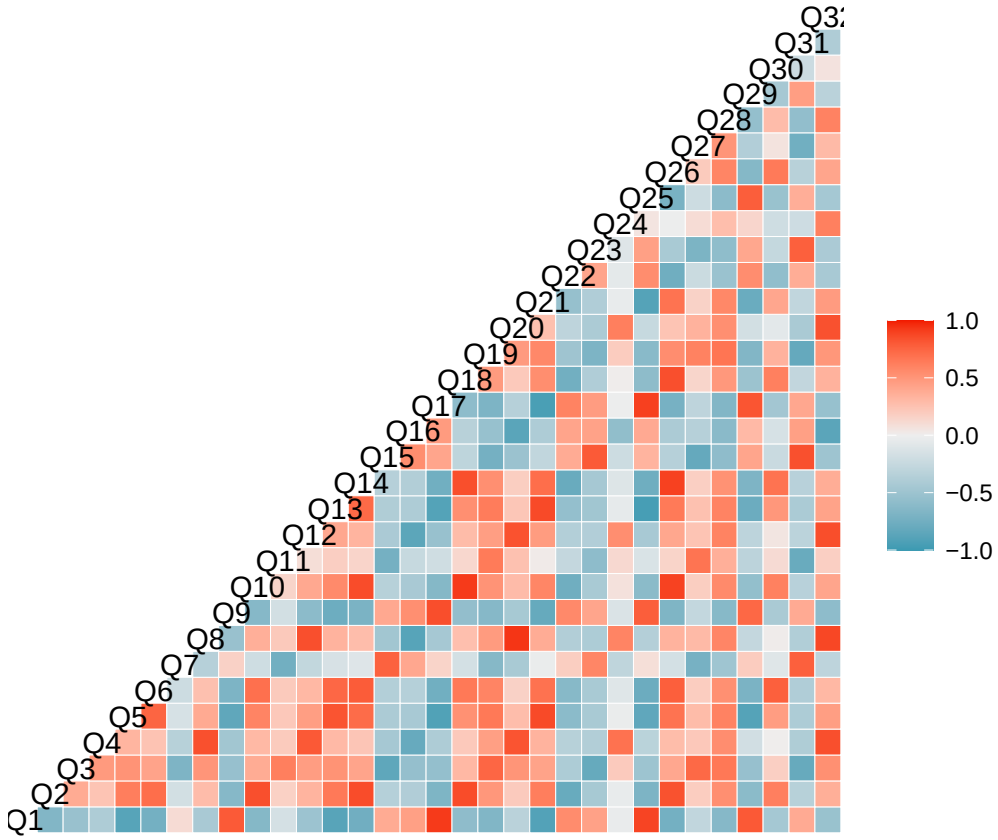
So entering 32 intercorrelated predictors into our model is not ideal, especially if we have a small sample size. This is sometimes referred to in the literature as the **curse of dimensionality**. Dimensionality refers to the fact that in statistical models such as regression, the more variables we have in the model, the higher dimensions are used in the math. For example, in simple regression, the regression line is truly a line that can be depicted in a two dimensional space: the dependent variable on the y axis and the predictor on the x axis. However, if we have two predictors (multiple regression), the regression line is actually a regression plane, that can only be depicted in a 3 dimensional space: the dependent variable on the y axis and the predictors on the x and z axes, and so on. In a linear regression with 32 predictors, the regression plane is 33 dimensional. The more dimensions we have, the more flexible our model is, able to fit to the nooks and crannies of the sampling error, leading to higher and higher risk of overfitting.

This problem gets more and more problematic as the number of dimensions or parameters in the model is getting close to the number of observations. If we fit a simple regression line (using only 1 predictor) on data with only 2 observations, there is a line that fits the data perfectly, since there is always a straight line that can connect two points. Since the regression will find the line which passes closest to the observations, with 2 observations we will end up with a regression model that has 0 error. This might seem good at first, but this is actually bad. This means that our model can perfectly fit the error in our sample, and will have almost no resemblance of the actual pattern in the population. The same goes if we have 3 observations and a model with 2 predictors: there is a plane (sheet) that can perfectly fit 3 points in a 3 dimensional space, so the model can again fit to the error and we don't know how much information we gain about the actual pattern in the whole population. So it is easy to see that the closer the number of dimensions is to the number of observations, the more flexible the model is, and the less informative the model coefficients would be for new data due to overfitting.

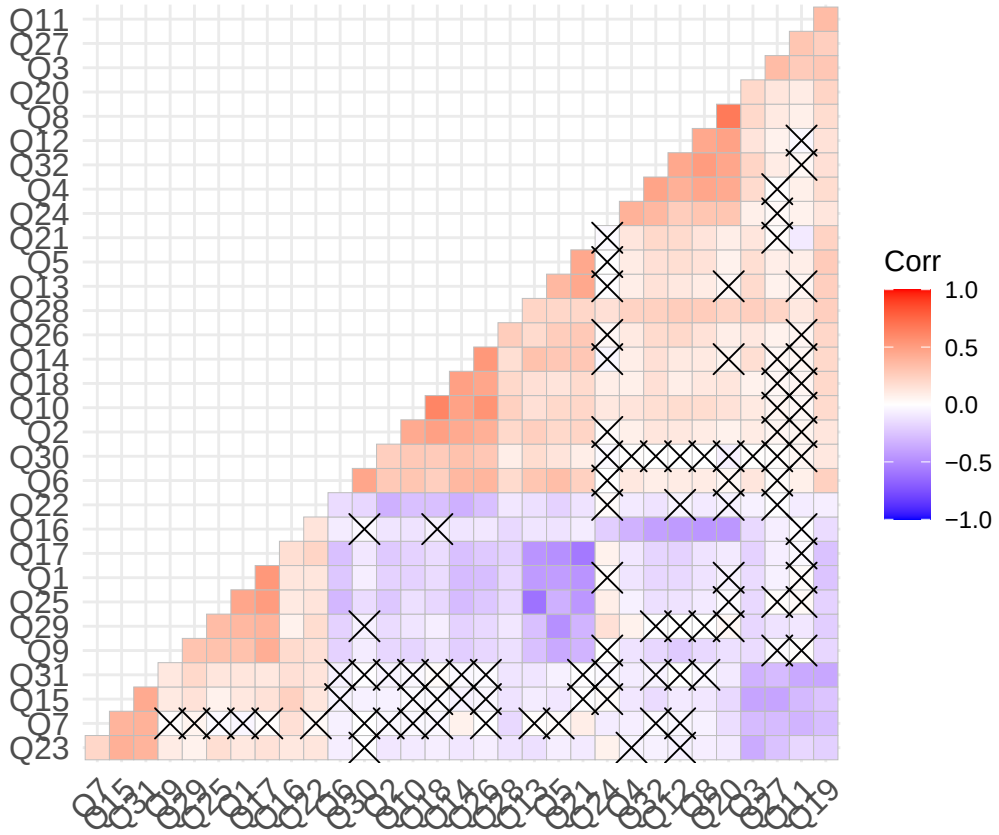
2.6 Visualizing the correlationsal structure

We can easily make many of the other aspects of the process transparent through visualization. However, when dealing with **a lot of variables**, visualization and other previously learned exploratory data analysis techniques may fail simply because there is too much information that is difficult to analyze and visualize. This is well illustrated in the graphs below.

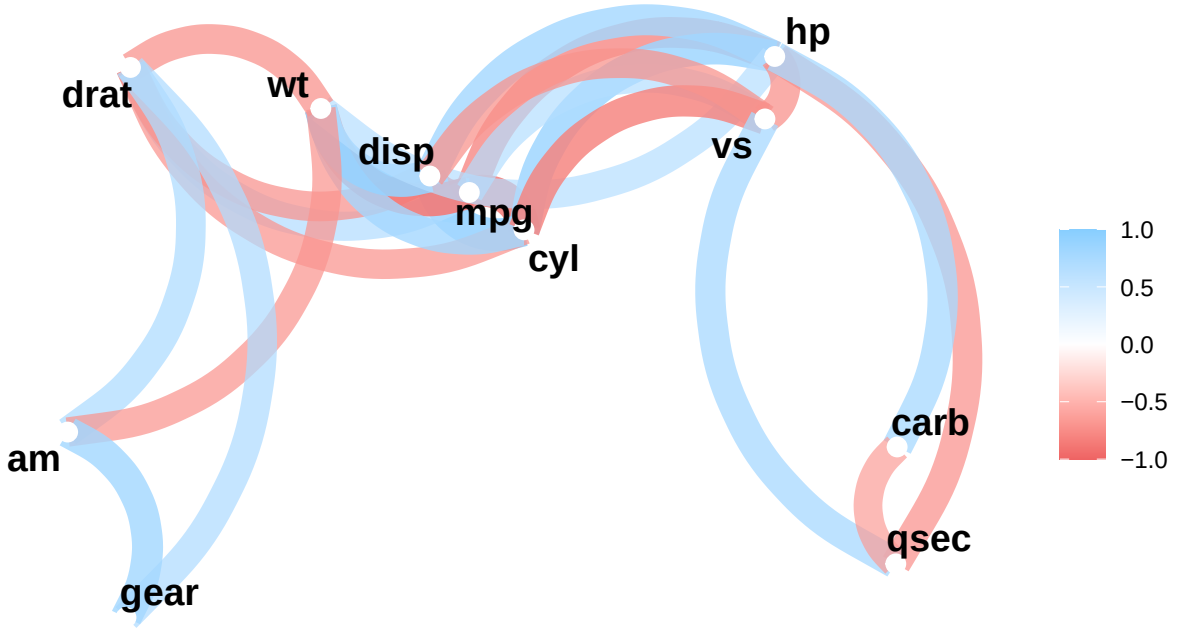
```
ggcorr(cor)
```




```
ggcorrplot(cor(hsq_items_only), p.mat = cor_pmat(hsq_items_only),
    hc.order = TRUE, type = "lower")
```

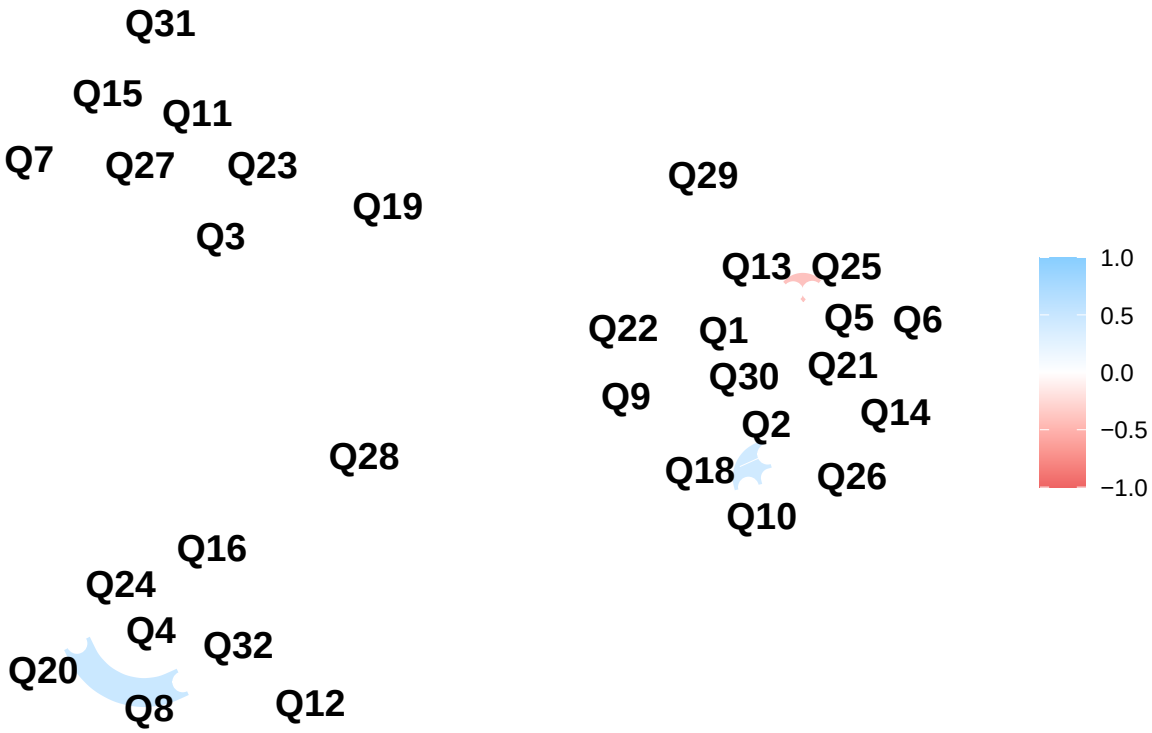



```
# from a different dataset, mtcars, because there are too
# many variables in hsq to be displayed here
cor(mtcars) %>%
    network_plot(min_cor = 0.6)
```




```
cor(hsq_items_only) %>%
    network_plot(min_cor = 0.6)
```

```
## Warning: ggrepel: 1 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



3 Principal component analysis (PCA)

3.1 How to build a PCA model

When facing the curse of dimensionality, one of the most straightforward options is to reduce the number of dimensions. We could do this by dropping some of the variables from the model, but based on what? If we did it by dropping the predictors with the lowest predictive power or correlation with the outcome we would be again contributing to overfitting, so this is not desirable. A better option would be to merge some of our more similar variables together, this way, decreasing the number of predictors overall, but still keeping information from all original predictors.

When we look at the correlation matrix of predictor variables we can notice that there are "clusters" of variables that are more closely related to each other than the other variables. For example variables Q1, Q5, Q13, Q17, and Q25 seem to form such a cluster. They are all correlated at around 0.4-0.5. This means that we could merge these variables into a single variable (for example by averaging them), and this way reducing the number of predictors by four.

This is essentially what we do in a controlled systematic way in principlal component analysis (PCA). In PCA we **identify the clusters of variables** that hold similar information to each other, and **create a "principal component"** out of these variables which **keeps the most possible variability within the data**.

3.2 How does PCA work

In the PCA, our goal is usually to reduce the number of dimensions with which we can describe our data. We do this by first looking for new dimensions that explain as much of the variance of the data as possible, and then discarding those dimensions that have a relatively small variance.

In PCA, we first identify a primary dimension that gives the largest variance of the data (in a data cloud this is the line that spans through the largest difference / deviation in the data). We then identify a new dimension that is perpendicular to the first dimension, which can explain the largest **remaining** variance in the data, and so on, until we reach the number of values originally put into the principal component analysis. (Since the dimensions are **perpendicular to each other**, the variance explained by two dimensions is never overlapping, there is **no redundancy** in the variance explained by the dimensions, i.e. the degree components are not correlated with each other.) Thus, the principal component analysis creates a new coordinate system in which the data can be viewed.

It is important that the dimensions of the new coordinate system largely differn in how much variability is "tored" or explained by them. The first few "principal components" (dimensions) cover a very large part of the total variance of the data, and in the remaining dimensions the data usually show little variability. The variance contained in a given dimension is called **eigenvalue**. As we move from the first to the last dimension, the eigenvalue (i.e. the variance observed in the dimension) decreases. This allows us to decrease the number of dimensions, because on the dimensions identified at the end of the PCA process, the variability will be negligible, so we can discard these dimensions and keep only the **most useful dimensions**.

Let's imagine, for example, that we are interested in the performance of first-graders on a test. In a study we measure the verbal skills, sociability and age of the children. It turns out that almost all first-graders in the sample are 6 years old, so there is almost no variability in age (if we only measure in years). We can ignore this fact in our research, since this data on this variable do not vary that much. On the other hand, in sociability and verbal skills, we see a greater diversity, difference between the children, so these are potentially important indicators in our research that can help us differentiate the children. During the PCA process we artificially generate new variables that either explain a lot of the variance or very little variance, so that in the end we can discard the ones with little variance and keep a few variables that explain a lot of variance. This way we can discard some variables (that explain low variance), while still retaining as much information as possible about the association of the data.

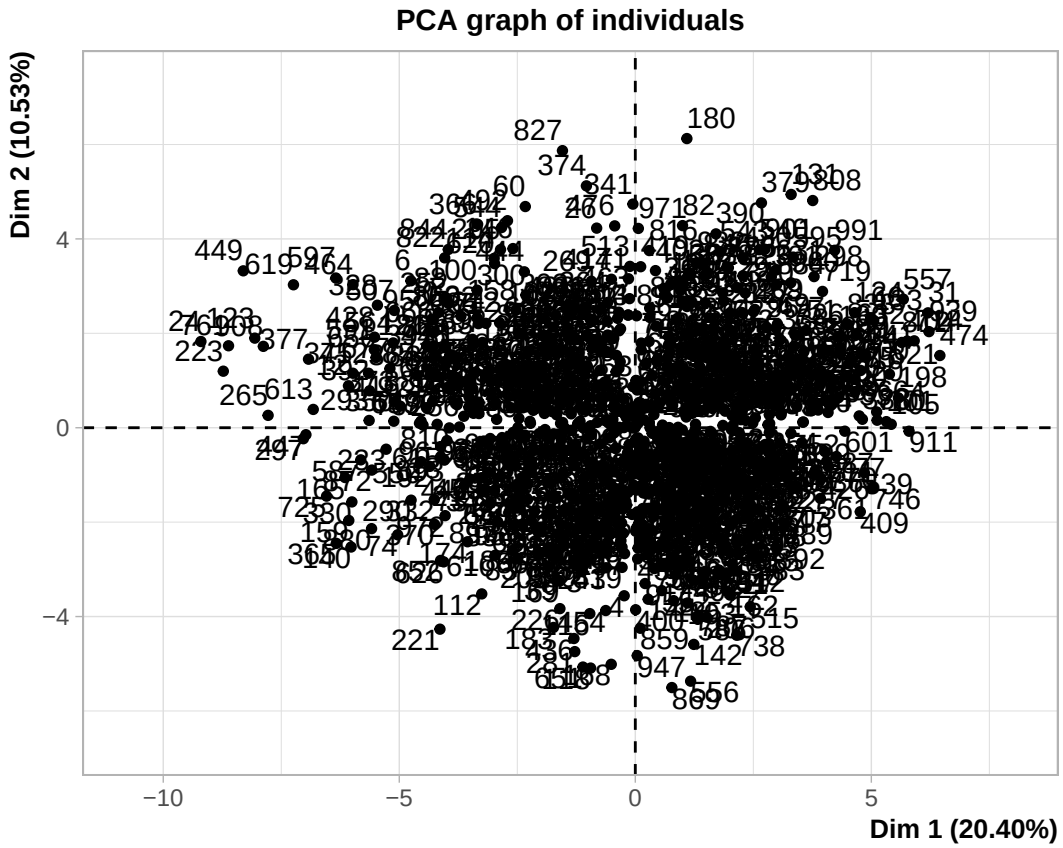
3.3 Using PCA in R

The principal component analysis can be performed with the **PCA() (principal component analysis)** function from the FactoMineR package. I saved the result of the analysis in a model object pca_mod. This will output two **figures of the two most important principal components (dimensions)** that describe the data most accurately.

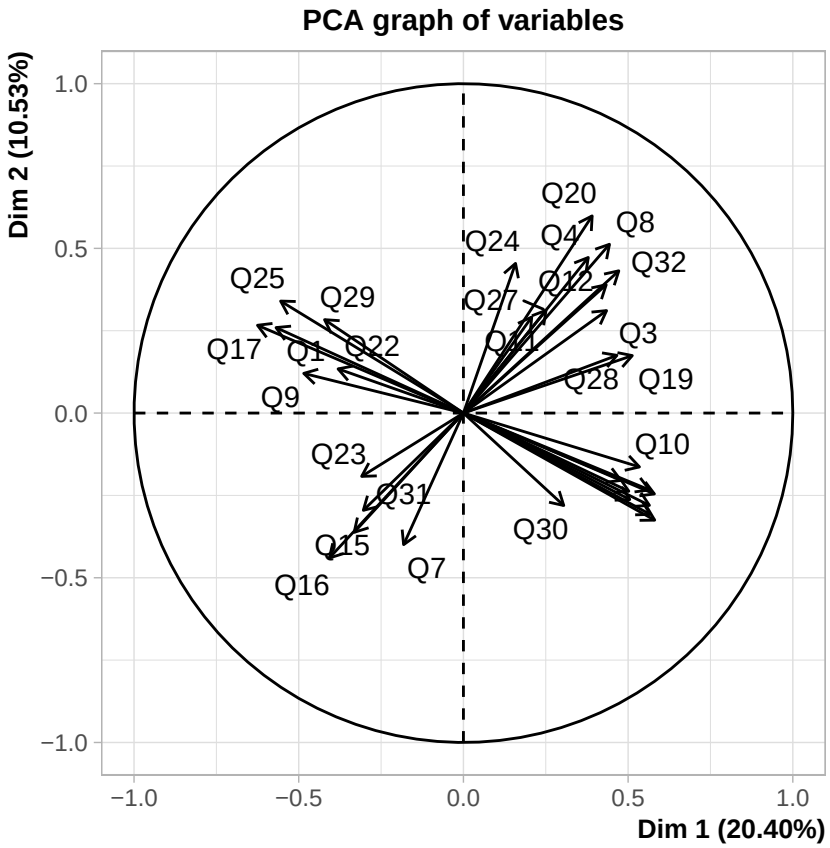
One of them shows **where each observation** (in this case, each car model) **is located in the two dimensions**. The second is about the correlation of the **dimensions with the original variables**. The dashed lines show the principal components. The closer the arrows are to the dashed line, the more the variable correlations with the dimension in question compared to the other dimension.

In fact, the items Q9 and Q10 are much better described by Dim1 than by Dim2 (the arrow's direction shows that Q9 is negatively correlated with Dim1, while Q10 is positively correlated with Dim1). On the other hand, the arrow of Q8 is located between the two dimensions, which means that it is correlated with both of them to almost the same degree (this can be very small or very large). The length of the arrow indicates the level of correlation.

```
pca_mod <- PCA(hsq_items_only)
```




```
## Warning: ggrepel: 8 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



We have to take care that categorical variables should not be entered in the PCA.

In the PCA() function, we have the possibility to specify the values in the database that we do not want to include in the PCA model.

Continuous variables that we don't want to include in the PCA should be specified in the quanti.sup parameter, and those that are categorical in the quali.sup parameter. Here we have to specify the column number of the added column, not the name, so we have to look it up first. This can be done with the function which(names(hsq) == "name of the column").

```
which(names(hsq) == "gender")
```

```
## [1] 38
```

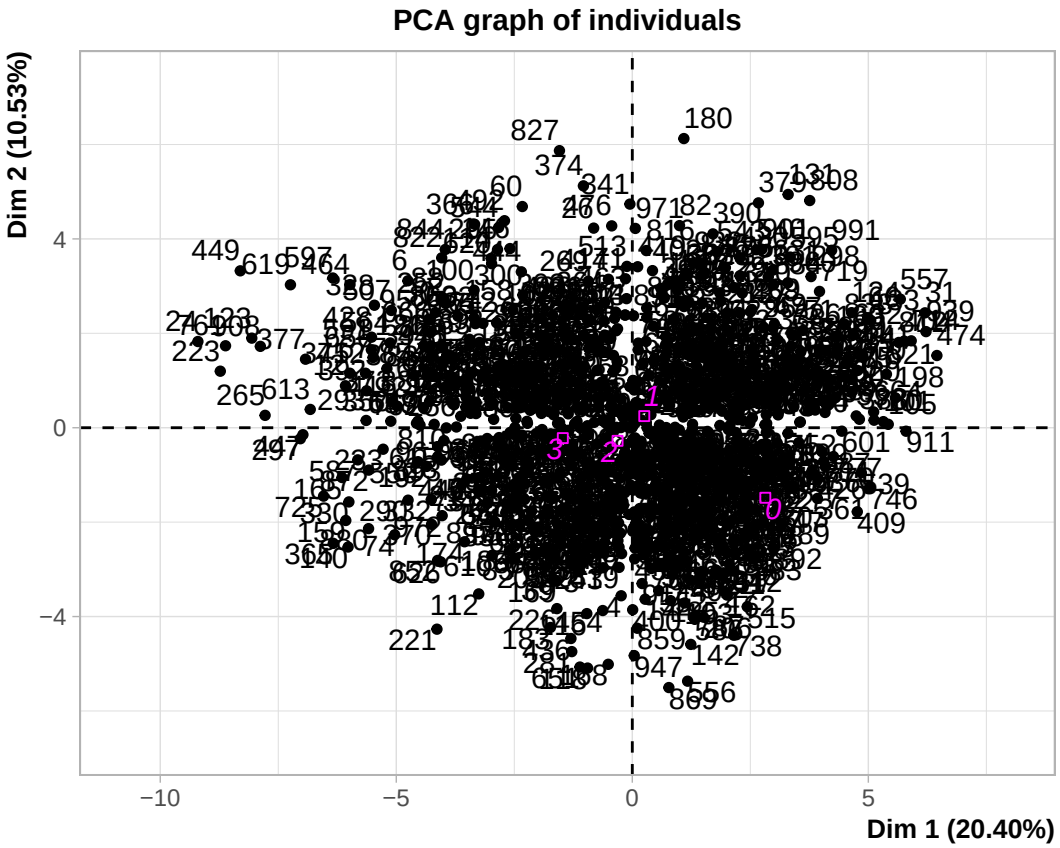
```
which(names(hsq) == "affiliative")
```

```
## [1] 33
```

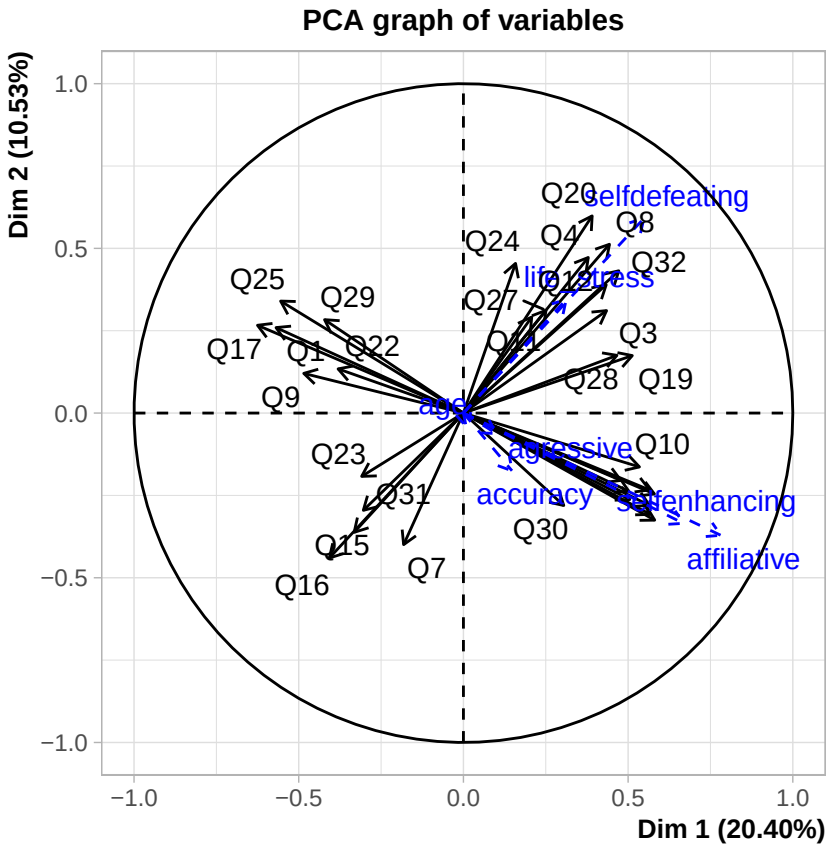
```
which(names(hsq) == "life_stress")
```

```
## [1] 40
```

```
pca_mod2 <- PCA(hsq, quanti.sup = c(33, 34, 35, 36, 37, 39, 40),
    quali.sup = 38)
```




```
## Warning: ggrepel: 8 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```




```
summary(pca_mod2)
```

```
##
## Call:
## PCA(X = hsq, quanti.sup = c(33, 34, 35, 36, 37, 39, 40), quali.sup = 38)
##
##
## Eigenvalues
##                        Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6   Dim.7
## Variance               6.527   3.370   2.800   2.256   1.200   1.061   1.037
## % of var.             20.396  10.531   8.751   7.050   3.749   3.315   3.241
## Cumulative % of var.  20.396  30.927  39.677  46.728  50.476  53.791  57.032
##                        Dim.8   Dim.9  Dim.10  Dim.11  Dim.12  Dim.13  Dim.14
## Variance               0.926   0.825   0.789   0.772   0.722   0.690   0.662
## % of var.              2.895   2.579   2.467   2.412   2.255   2.157   2.070
## Cumulative % of var.  59.927  62.506  64.973  67.385  69.640  71.797  73.867
##                       Dim.15  Dim.16  Dim.17  Dim.18  Dim.19  Dim.20  Dim.21
## Variance               0.624   0.611   0.579   0.555   0.544   0.530   0.506
## % of var.              1.951   1.911   1.808   1.736   1.699   1.658   1.581
## Cumulative % of var.  75.818  77.728  79.537  81.273  82.972  84.630  86.211
##                       Dim.22  Dim.23  Dim.24  Dim.25  Dim.26  Dim.27  Dim.28
## Variance               0.488   0.484   0.467   0.442   0.419   0.412   0.398
## % of var.              1.524   1.512   1.459   1.383   1.308   1.287   1.245
## Cumulative % of var.  87.735  89.247  90.706  92.089  93.397  94.684  95.929
##                       Dim.29  Dim.30  Dim.31  Dim.32
## Variance               0.363   0.348   0.307   0.285
```

```
## % of var.              1.135   1.086   0.959   0.890
## Cumulative % of var.  97.064  98.151  99.110 100.000
##
## Individuals (the 10 first)
##                   Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3
## 1              |  3.886 | -0.878  0.012  0.051 | -1.645  0.081  0.179 | -1.133
## 2              |  4.727 | -2.213  0.076  0.219 | -0.739  0.016  0.024 |  0.932
## 3              |  3.771 | -0.306  0.001  0.007 | -1.287  0.049  0.116 |  0.197
## 4              |  4.390 | -0.236  0.001  0.003 | -3.560  0.379  0.658 |  0.309
## 5              |  3.698 | -1.915  0.057  0.268 |  0.464  0.006  0.016 | -0.026
## 6              |  7.583 | -4.753  0.349  0.393 |  3.112  0.289  0.168 | -0.131
## 7              |  4.404 |  1.913  0.056  0.189 | -1.233  0.045  0.078 | -0.842
## 8              |  4.817 | -1.521  0.036  0.100 | -1.757  0.092  0.133 | -2.019
## 9              |  9.092 | -3.998  0.247  0.193 |  1.392  0.058  0.023 | -2.109
## 10             |  5.934 |  2.772  0.119  0.218 | -1.989  0.118  0.112 |  1.588
##                   ctr   cos2
## 1              0.046  0.085 |
## 2              0.031  0.039 |
## 3              0.001  0.003 |
## 4              0.003  0.005 |
## 5              0.000  0.000 |
## 6              0.001  0.000 |
## 7              0.026  0.037 |
## 8              0.147  0.176 |
## 9              0.160  0.054 |
## 10             0.091  0.072 |
##
## Variables (the 10 first)
##                   Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr
## Q1             | -0.569  4.959  0.324 |  0.259  1.997  0.067 |  0.067  0.158
## Q2             |  0.502  3.863  0.252 | -0.237  1.671  0.056 |  0.123  0.539
## Q3             |  0.435  2.893  0.189 |  0.311  2.870  0.097 | -0.370  4.876
## Q4             |  0.379  2.202  0.144 |  0.472  6.620  0.223 |  0.352  4.418
## Q5             |  0.580  5.151  0.336 | -0.246  1.789  0.060 | -0.064  0.144
## Q6             |  0.505  3.908  0.255 | -0.263  2.045  0.069 |  0.022  0.018
## Q7             | -0.181  0.505  0.033 | -0.399  4.735  0.160 |  0.464  7.677
## Q8             |  0.443  3.012  0.197 |  0.512  7.791  0.263 |  0.382  5.201
## Q9             | -0.485  3.601  0.235 |  0.120  0.431  0.015 |  0.010  0.004
## Q10            |  0.534  4.376  0.286 | -0.163  0.793  0.027 |  0.242  2.084
##                   cos2
## Q1             0.004 |
## Q2             0.015 |
## Q3             0.137 |
## Q4             0.124 |
## Q5             0.004 |
## Q6             0.000 |
## Q7             0.215 |
## Q8             0.146 |
## Q9             0.000 |
## Q10            0.058 |
##
## Supplementary continuous variables
##                   Dim.1   cos2    Dim.2   cos2    Dim.3   cos2
## affiliative    |  0.781  0.611 | -0.371  0.138 | -0.126  0.016 |
```

```
## selfenhancing |  0.668  0.446 | -0.335  0.112 |  0.214  0.046 |
## agressive     |  0.100  0.010 | -0.062  0.004 |  0.193  0.037 |
## selfdefeating |  0.548  0.300 |  0.591  0.349 |  0.451  0.204 |
## age           |  0.008  0.000 | -0.041  0.002 |  0.026  0.001 |
## accuracy      |  0.147  0.022 | -0.175  0.031 | -0.054  0.003 |
## life_stress   |  0.312  0.097 |  0.345  0.119 |  0.251  0.063 |
##
## Supplementary categories
##                    Dist    Dim.1   cos2 v.test    Dim.2   cos2 v.test    Dim.3
## 0              |  3.616 |  2.818  0.607  2.471 | -1.484  0.168 -1.811 | -0.438
## 1              |  0.436 |  0.254  0.339  3.394 |  0.246  0.319  4.583 | -0.183
## 2              |  0.514 | -0.313  0.370 -3.460 | -0.278  0.292 -4.276 |  0.220
## 3              |  2.918 | -1.473  0.255 -1.636 | -0.220  0.006 -0.340 |  0.347
##                 cos2 v.test
## 0              0.015 -0.587 |
## 1              0.176 -3.734 |
## 2              0.184  3.721 |
## 3              0.014  0.588 |
```

3.4 How many components to extract

PCA is a dimensional reduction technique, where **our goal is to have fewer dimensions** by the end of the analysis than we started with. However, if you look at the model summary, you can see that the PCA generated **exactly as many dimensions as there were instruments** in the default mode. Not very helpful is it?

Now we need to specify in the PCA function **how many dimensions we want to keep** out of the many dimensions extracted. But how can we tell how many dimensions are the ideal number?

There are multiple ways to evaluate this.

1. Scree test

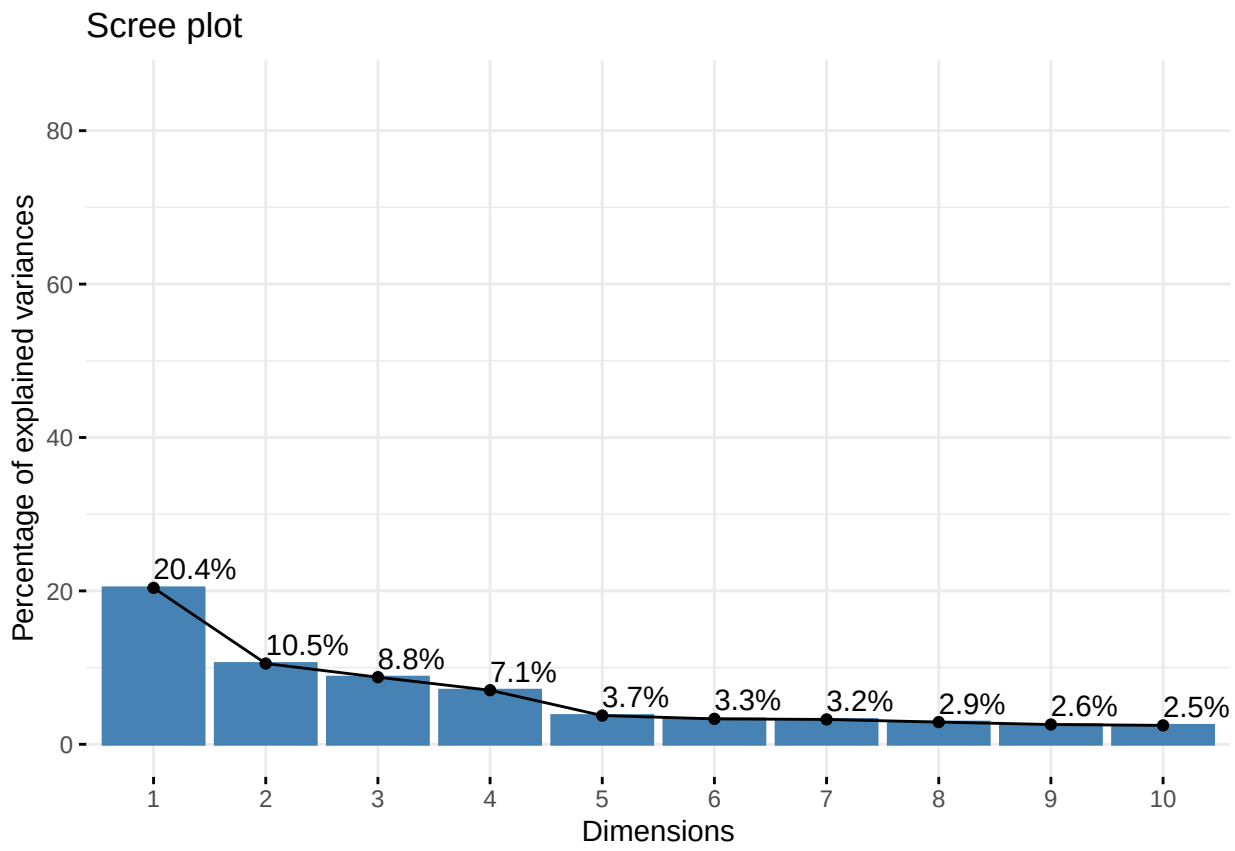
A well accepted method for deciding about the number of dimensions to retain is the Scree test.

First, we ned to visualize the eigenvalues (amount of variance contained in each dimentsion) using the fviz_screeplot() function. The scree plot shows the eigenvalue of each dimension connected by a line.

The scree test is a visual analysis of the scree plot that starts with **finding the "elbow" in the line**. This is the point where the eigenvalues seem to level off, where no more substantial break can be seen in the slope of the line. Components **to the left of this point should be retained**. **This does not include the last break point**, so everything *before* that point is retained, and every other dimension, including the one which represents the final breaking point, is excluded. This method is commonly used and is accepted in the literature, but it can be subjective and different researchers might come to different conclusions based on this.

This rule suggests that we should retain 4 component, since the last substantial break in the slope is at the 5th factor.

```
fviz_screeplot(pca_mod2, addlabels = TRUE, ylim = c(0, 85))
```



The Kaiser-Guttman rule

One strategy that is objective and has the same result for each researcher is **the Kaiser-Guttman rule**. In this method we look at the Eigenvalues of the dimensions and **retain the ones that have Eigenvalue > 1**, and exclude every other dimension.

The eigenvalue of 1 has a bit of a special significance, because the average of the eigenvalues is always 1, whatever dimensions you use for describing the data. This also means that the eigenvalue for the original set of dimensions (defined by the original variables) also had an average eigenvalue of 1, and thus any dimension that has lower than 1 eigenvalue is explaining less variance than an average variable among the intial variables we started out with.

So dimensions with eigenvalues below 1 are considered less useful, since our initial goal was to combine the information from multiple variables into "super variables", but if the new "super variable" explains less variance than the average single original variable, it is not seen as so "super" after all.

This decision rule for retaining the components **is not commonly used**, or is only used in combination with other rules.

This rule suggests that in our example we should retain 7 components, since even the 7th component has higher eigenvalue than 1.

```
get_eigenvalue(pca_mod2)
```

```
##        eigenvalue variance.percent cumulative.variance.percent
## Dim.1   6.5266194       20.3956857                    20.39569
## Dim.2   3.3699443       10.5310758                    30.92676
## Dim.3   2.8002011        8.7506285                    39.67739
## Dim.4   2.2560527        7.0501646                    46.72755
```

```
## Dim.5   1.1995880       3.7487124                50.47627
## Dim.6   1.0608149       3.3150464                53.79131
## Dim.7   1.0370806       3.2408769                57.03219
## Dim.8   0.9263495       2.8948422                59.92703
## Dim.9   0.8252450       2.5788906                62.50592
## Dim.10  0.7894123       2.4669134                64.97284
## Dim.11  0.7718522       2.4120381                67.38487
## Dim.12  0.7216551       2.2551722                69.64005
## Dim.13  0.6903207       2.1572520                71.79730
## Dim.14  0.6622550       2.0695470                73.86685
## Dim.15  0.6242740       1.9508561                75.81770
## Dim.16  0.6114501       1.9107815                77.72848
## Dim.17  0.5786557       1.8082991                79.53678
## Dim.18  0.5554699       1.7358435                81.27263
## Dim.19  0.5437363       1.6991759                82.97180
## Dim.20  0.5304791       1.6577472                84.62955
## Dim.21  0.5059257       1.5810178                86.21057
## Dim.22  0.4877984       1.5243699                87.73494
## Dim.23  0.4839852       1.5124538                89.24739
## Dim.24  0.4668567       1.4589273                90.70632
## Dim.25  0.4424163       1.3825510                92.08887
## Dim.26  0.4185422       1.3079445                93.39681
## Dim.27  0.4118812       1.2871287                94.68394
## Dim.28  0.3984002       1.2450006                95.92894
## Dim.29  0.3633398       1.1354370                97.06438
## Dim.30  0.3476741       1.0864816                98.15086
## Dim.31  0.3069190       0.9591218                99.10998
## Dim.32  0.2848053       0.8900167               100.00000
```

Parallel analysis

The third, and currently most accepted, decision rule for the number of dimensions to retain in PCA is using **parallel analysis**. The idea of parallel analysis is that we generate/simulate datasets which have similar characteristics to our original dataset (the same number of observations and the same number of variables), but we simulate that these variables come from a population where the variables are uncorrelated. We repeat this many times. 1000-10000 is tipically used, the number depending on the level of precision we want to achieve. Generally the more the better, but the precision gained in the last 5000 iterations is not that great, so if you have a large dataset and a slow computer, you should first consider running a 1000 iterations, and only run the final analysis with 5-10000 iterations once you have arrived at your final model.

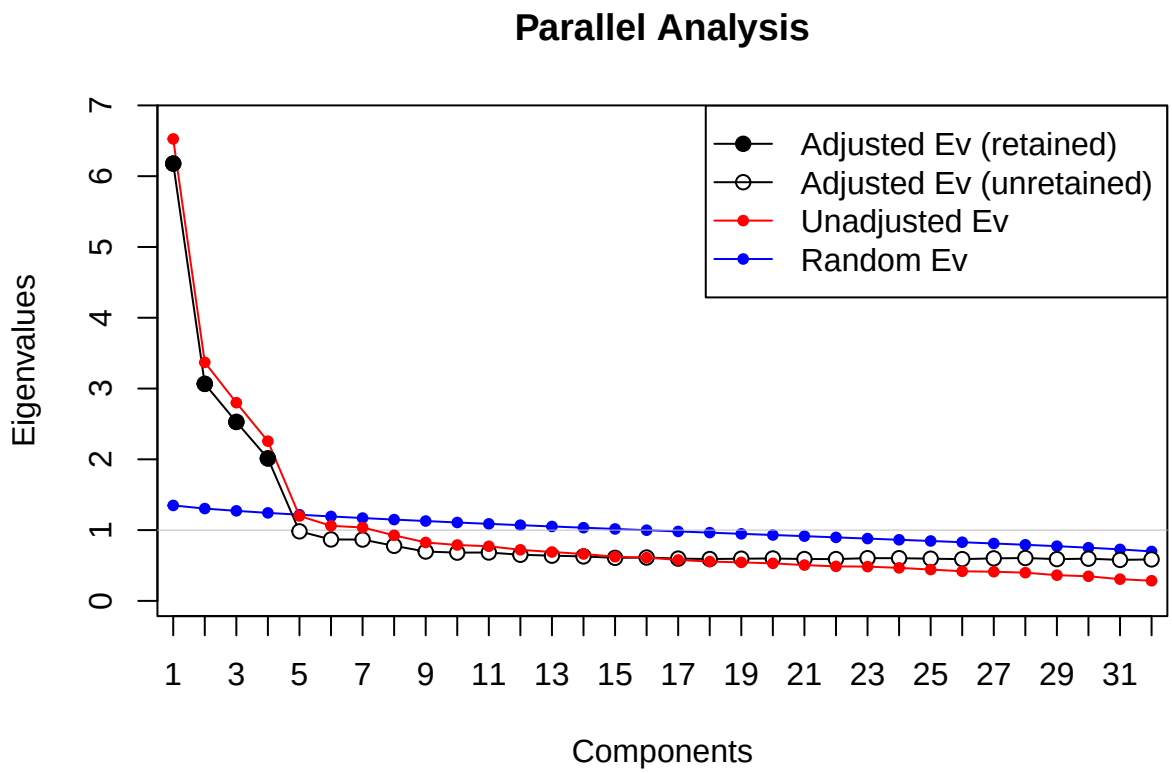
We perform the same PCA separately on all of these random datasets, and take the average eigenvalue for each component, then, we create a scree plot of these average eigenvalues as well, and compare it to the scree plot (eigenvalues) of our original dataset. Thus, essentially we use the eigenvalues derived from the random datasets as a sort of "null-model". that is, we get the eigenvalue pattern that we would get just by random chance if there is no correlation between the variables whatsoever.

We retain the components which have a higher eigenvalue than the average eigenvalue of the random data corresponding to the same component. In other words we can keep all components which are above the random data eigenvalue line on the scree plot.

This parallel analysis can be done with the **paran()** function from the paran package. This function can also visualize the null eigenvalue graph with the parameter graph = TRUE, which can be compared with the eigenvalue obtained in our data. The $Retained component of the output object shows how many dimensions the analysis proposes to retain. In this example, the parallel analysis proposes to retain 4 components (just like the scree test before).

```
pca_ret = paran(hsq_items_only, graph = TRUE)
```

```
##
## Using eigendecomposition of correlation matrix.
## Computing: 10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
##
##
## Results of Horn's Parallel Analysis for component retention
## 960 iterations, using the mean estimate
##
## ----------------------------------------------------
## Component   Adjusted    Unadjusted    Estimated
##             Eigenvalue  Eigenvalue    Bias
## ----------------------------------------------------
## 1           6.178165    6.526619      0.348454
## 2           3.065868    3.369944      0.304076
## 3           2.528125    2.800201      0.272075
## 4           2.012977    2.256052      0.243074
## ----------------------------------------------------
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
## (4 components retained)
```

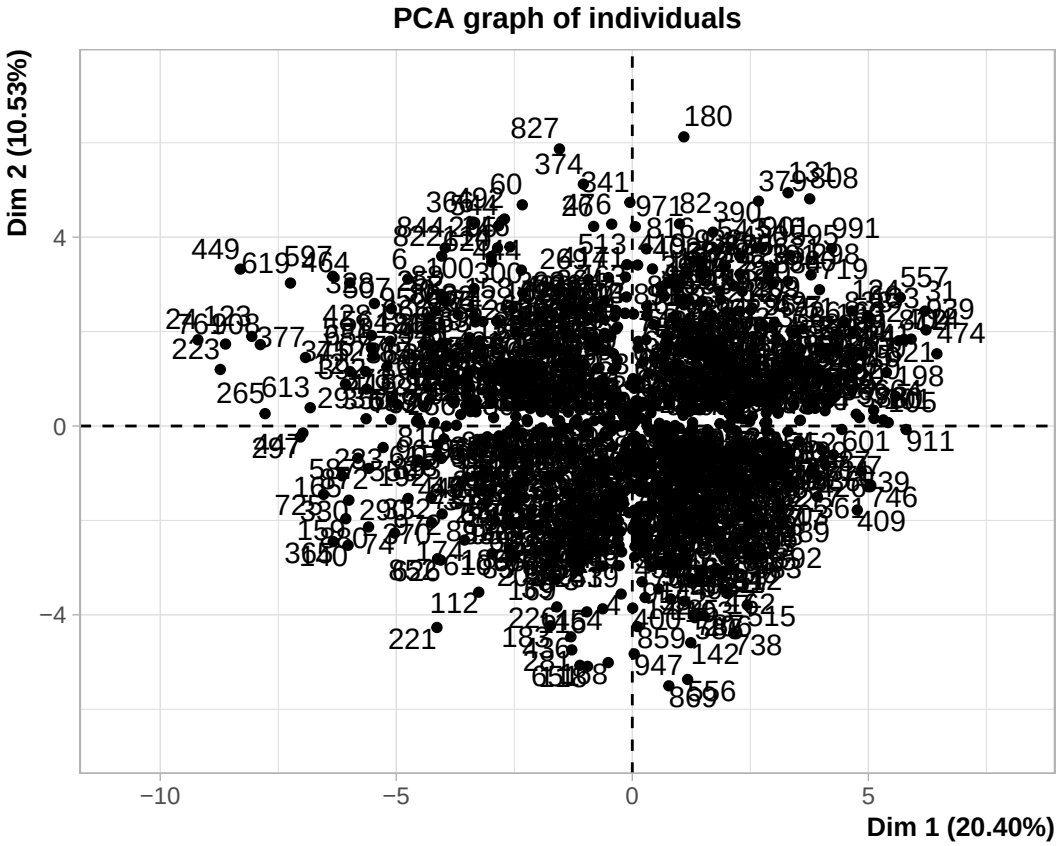



```
pca_ret$Retained
```

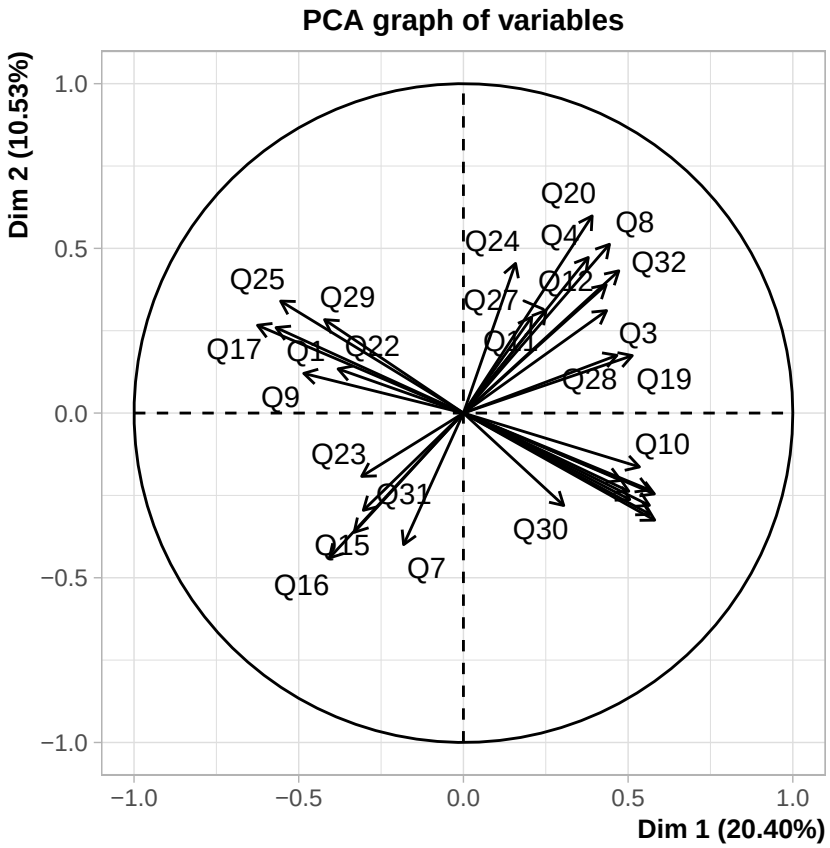
```
## [1] 4
```

Once we have the ideal set of dimensions, we can run our analysis again, but this time specifying how many dimensions we want, with the npc parameter.

```
pca_mod3 <- PCA(hsq_items_only, ncp = 4)
```




```
## Warning: ggrepel: 8 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```




```
summary(pca_mod3)
```

```
##
## Call:
## PCA(X = hsq_items_only, ncp = 4)
##
##
## Eigenvalues
##                        Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6   Dim.7
## Variance               6.527   3.370   2.800   2.256   1.200   1.061   1.037
## % of var.             20.396  10.531   8.751   7.050   3.749   3.315   3.241
## Cumulative % of var.  20.396  30.927  39.677  46.728  50.476  53.791  57.032
##                        Dim.8   Dim.9  Dim.10  Dim.11  Dim.12  Dim.13  Dim.14
## Variance               0.926   0.825   0.789   0.772   0.722   0.690   0.662
## % of var.              2.895   2.579   2.467   2.412   2.255   2.157   2.070
## Cumulative % of var.  59.927  62.506  64.973  67.385  69.640  71.797  73.867
##                       Dim.15  Dim.16  Dim.17  Dim.18  Dim.19  Dim.20  Dim.21
## Variance               0.624   0.611   0.579   0.555   0.544   0.530   0.506
## % of var.              1.951   1.911   1.808   1.736   1.699   1.658   1.581
## Cumulative % of var.  75.818  77.728  79.537  81.273  82.972  84.630  86.211
##                       Dim.22  Dim.23  Dim.24  Dim.25  Dim.26  Dim.27  Dim.28
## Variance               0.488   0.484   0.467   0.442   0.419   0.412   0.398
## % of var.              1.524   1.512   1.459   1.383   1.308   1.287   1.245
## Cumulative % of var.  87.735  89.247  90.706  92.089  93.397  94.684  95.929
##                       Dim.29  Dim.30  Dim.31  Dim.32
## Variance               0.363   0.348   0.307   0.285
```

```
## % of var.              1.135   1.086   0.959   0.890
## Cumulative % of var.  97.064  98.151  99.110 100.000
##
## Individuals (the 10 first)
##         Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr
## 1    |  3.886 | -0.878  0.012  0.051 | -1.645  0.081  0.179 | -1.133  0.046
## 2    |  4.727 | -2.213  0.076  0.219 | -0.739  0.016  0.024 |  0.932  0.031
## 3    |  3.771 | -0.306  0.001  0.007 | -1.287  0.049  0.116 |  0.197  0.001
## 4    |  4.390 | -0.236  0.001  0.003 | -3.560  0.379  0.658 |  0.309  0.003
## 5    |  3.698 | -1.915  0.057  0.268 |  0.464  0.006  0.016 | -0.026  0.000
## 6    |  7.583 | -4.753  0.349  0.393 |  3.112  0.289  0.168 | -0.131  0.001
## 7    |  4.404 |  1.913  0.056  0.189 | -1.233  0.045  0.078 | -0.842  0.026
## 8    |  4.817 | -1.521  0.036  0.100 | -1.757  0.092  0.133 | -2.019  0.147
## 9    |  9.092 | -3.998  0.247  0.193 |  1.392  0.058  0.023 | -2.109  0.160
## 10   |  5.934 |  2.772  0.119  0.218 | -1.989  0.118  0.112 |  1.588  0.091
##       cos2
## 1    0.085 |
## 2    0.039 |
## 3    0.003 |
## 4    0.005 |
## 5    0.000 |
## 6    0.000 |
## 7    0.037 |
## 8    0.176 |
## 9    0.054 |
## 10   0.072 |
##
## Variables (the 10 first)
##        Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr   cos2
## Q1   | -0.569  4.959  0.324 |  0.259  1.997  0.067 |  0.067  0.158  0.004 |
## Q2   |  0.502  3.863  0.252 | -0.237  1.671  0.056 |  0.123  0.539  0.015 |
## Q3   |  0.435  2.893  0.189 |  0.311  2.870  0.097 | -0.370  4.876  0.137 |
## Q4   |  0.379  2.202  0.144 |  0.472  6.620  0.223 |  0.352  4.418  0.124 |
## Q5   |  0.580  5.151  0.336 | -0.246  1.789  0.060 | -0.064  0.144  0.004 |
## Q6   |  0.505  3.908  0.255 | -0.263  2.045  0.069 |  0.022  0.018  0.000 |
## Q7   | -0.181  0.505  0.033 | -0.399  4.735  0.160 |  0.464  7.677  0.215 |
## Q8   |  0.443  3.012  0.197 |  0.512  7.791  0.263 |  0.382  5.201  0.146 |
## Q9   | -0.485  3.601  0.235 |  0.120  0.431  0.015 |  0.010  0.004  0.000 |
## Q10  |  0.534  4.376  0.286 | -0.163  0.793  0.027 |  0.242  2.084  0.058 |
```

3.5 Interpret the results of PCA

3.5.1 The componenets of the PCA model

The **model summary** provides further useful information.

The **Eigenvalues** section shows how much of the total variance of the **data is explained by each dimension (% of var)**, and the cumulative % of variance explained by each dimension, from the most important to the least important. In other words, the % of variance for Dim.3 (8.75) shows that the third extracted dimension can explain 8.75% of the variance of the data alone. And the dim Cumulative % of variance (39.68) for Dim.3 shows that Dim.3, together with Dim.1 and Dim.2, can explain 39.68% of the variance in the data. If we are only interested in the table containing the eigenvalue and the explained variance values, we can get it by listing only the component pca_mod3$eig.

The model summary also contains information in the variables component about how each **variable correlates with each new dimension** (in the Dim.1, Dim.2, Dim.3 ... columns in the "cor" component), and how

much they contribute to the variance explained by the given variable (in the ctr component). This is a very important table, because from here we can read (besides the graphs) which variables are discribed best by thich dimensions (factors). You can learn if you look at the pca_mod3$var component.

```
# Get the summary the outputs.
summary(pca_mod3)
```

```
##
## Call:
## PCA(X = hsq_items_only, ncp = 4)
##
##
## Eigenvalues
##                        Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6   Dim.7
## Variance               6.527   3.370   2.800   2.256   1.200   1.061   1.037
## % of var.             20.396  10.531   8.751   7.050   3.749   3.315   3.241
## Cumulative % of var.  20.396  30.927  39.677  46.728  50.476  53.791  57.032
##                        Dim.8   Dim.9  Dim.10  Dim.11  Dim.12  Dim.13  Dim.14
## Variance               0.926   0.825   0.789   0.772   0.722   0.690   0.662
## % of var.              2.895   2.579   2.467   2.412   2.255   2.157   2.070
## Cumulative % of var.  59.927  62.506  64.973  67.385  69.640  71.797  73.867
##                       Dim.15  Dim.16  Dim.17  Dim.18  Dim.19  Dim.20  Dim.21
## Variance               0.624   0.611   0.579   0.555   0.544   0.530   0.506
## % of var.              1.951   1.911   1.808   1.736   1.699   1.658   1.581
## Cumulative % of var.  75.818  77.728  79.537  81.273  82.972  84.630  86.211
##                       Dim.22  Dim.23  Dim.24  Dim.25  Dim.26  Dim.27  Dim.28
## Variance               0.488   0.484   0.467   0.442   0.419   0.412   0.398
## % of var.              1.524   1.512   1.459   1.383   1.308   1.287   1.245
## Cumulative % of var.  87.735  89.247  90.706  92.089  93.397  94.684  95.929
##                       Dim.29  Dim.30  Dim.31  Dim.32
## Variance               0.363   0.348   0.307   0.285
## % of var.              1.135   1.086   0.959   0.890
## Cumulative % of var.  97.064  98.151  99.110 100.000
##
## Individuals (the 10 first)
##         Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr
## 1    |  3.886 | -0.878  0.012  0.051 | -1.645  0.081  0.179 | -1.133  0.046
## 2    |  4.727 | -2.213  0.076  0.219 | -0.739  0.016  0.024 |  0.932  0.031
## 3    |  3.771 | -0.306  0.001  0.007 | -1.287  0.049  0.116 |  0.197  0.001
## 4    |  4.390 | -0.236  0.001  0.003 | -3.560  0.379  0.658 |  0.309  0.003
## 5    |  3.698 | -1.915  0.057  0.268 |  0.464  0.006  0.016 | -0.026  0.000
## 6    |  7.583 | -4.753  0.349  0.393 |  3.112  0.289  0.168 | -0.131  0.001
## 7    |  4.404 |  1.913  0.056  0.189 | -1.233  0.045  0.078 | -0.842  0.026
## 8    |  4.817 | -1.521  0.036  0.100 | -1.757  0.092  0.133 | -2.019  0.147
## 9    |  9.092 | -3.998  0.247  0.193 |  1.392  0.058  0.023 | -2.109  0.160
## 10   |  5.934 |  2.772  0.119  0.218 | -1.989  0.118  0.112 |  1.588  0.091
##       cos2
## 1    0.085 |
## 2    0.039 |
## 3    0.003 |
## 4    0.005 |
## 5    0.000 |
## 6    0.000 |
## 7    0.037 |
## 8    0.176 |
```

```
## 9    0.054 |
## 10   0.072 |
##
## Variables (the 10 first)
##        Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr   cos2
## Q1  | -0.569  4.959  0.324 |  0.259  1.997  0.067 |  0.067  0.158  0.004 |
## Q2  |  0.502  3.863  0.252 | -0.237  1.671  0.056 |  0.123  0.539  0.015 |
## Q3  |  0.435  2.893  0.189 |  0.311  2.870  0.097 | -0.370  4.876  0.137 |
## Q4  |  0.379  2.202  0.144 |  0.472  6.620  0.223 |  0.352  4.418  0.124 |
## Q5  |  0.580  5.151  0.336 | -0.246  1.789  0.060 | -0.064  0.144  0.004 |
## Q6  |  0.505  3.908  0.255 | -0.263  2.045  0.069 |  0.022  0.018  0.000 |
## Q7  | -0.181  0.505  0.033 | -0.399  4.735  0.160 |  0.464  7.677  0.215 |
## Q8  |  0.443  3.012  0.197 |  0.512  7.791  0.263 |  0.382  5.201  0.146 |
## Q9  | -0.485  3.601  0.235 |  0.120  0.431  0.015 |  0.010  0.004  0.000 |
## Q10 |  0.534  4.376  0.286 | -0.163  0.793  0.027 |  0.242  2.084  0.058 |
```

```
pca_mod3$eig
```

```
##         eigenvalue percentage of variance cumulative percentage of variance
## comp 1   6.5266194             20.3956857                           20.39569
## comp 2   3.3699443             10.5310758                           30.92676
## comp 3   2.8002011              8.7506285                           39.67739
## comp 4   2.2560527              7.0501646                           46.72755
## comp 5   1.1995880              3.7487124                           50.47627
## comp 6   1.0608149              3.3150464                           53.79131
## comp 7   1.0370806              3.2408769                           57.03219
## comp 8   0.9263495              2.8948422                           59.92703
## comp 9   0.8252450              2.5788906                           62.50592
## comp 10  0.7894123              2.4669134                           64.97284
## comp 11  0.7718522              2.4120381                           67.38487
## comp 12  0.7216551              2.2551722                           69.64005
## comp 13  0.6903207              2.1572520                           71.79730
## comp 14  0.6622550              2.0695470                           73.86685
## comp 15  0.6242740              1.9508561                           75.81770
## comp 16  0.6114501              1.9107815                           77.72848
## comp 17  0.5786557              1.8082991                           79.53678
## comp 18  0.5554699              1.7358435                           81.27263
## comp 19  0.5437363              1.6991759                           82.97180
## comp 20  0.5304791              1.6577472                           84.62955
## comp 21  0.5059257              1.5810178                           86.21057
## comp 22  0.4877984              1.5243699                           87.73494
## comp 23  0.4839852              1.5124538                           89.24739
## comp 24  0.4668567              1.4589273                           90.70632
## comp 25  0.4424163              1.3825510                           92.08887
## comp 26  0.4185422              1.3079445                           93.39681
## comp 27  0.4118812              1.2871287                           94.68394
## comp 28  0.3984002              1.2450006                           95.92894
## comp 29  0.3633398              1.1354370                           97.06438
## comp 30  0.3476741              1.0864816                           98.15086
## comp 31  0.3069190              0.9591218                           99.10998
## comp 32  0.2848053              0.8900167                          100.00000
```

```
pca_mod3$var
```

```
## $coord
```

```
##          Dim.1      Dim.2       Dim.3       Dim.4
## Q1  -0.5688915  0.2594476  0.06654987  0.34960522
## Q2   0.5021178 -0.2373143  0.12284157  0.38417814
## Q3   0.4345612  0.3110200 -0.36950854  0.02163271
## Q4   0.3790587  0.4723304  0.35171209 -0.08316700
## Q5   0.5798081 -0.2455561 -0.06358979 -0.27455182
## Q6   0.5050051 -0.2625465  0.02225282  0.18236620
## Q7  -0.1814692 -0.3994541  0.46364367 -0.20124909
## Q8   0.4433948  0.5123844  0.38163200 -0.10676883
## Q9  -0.4847889  0.1204558  0.01028084  0.29329403
## Q10  0.5344008 -0.1634825  0.24158276  0.49608475
## Q11  0.2073377  0.2914197 -0.45068030  0.25594805
## Q12  0.4334654  0.3894721  0.38434674 -0.17975417
## Q13  0.5646683 -0.2802503 -0.09285473 -0.27570231
## Q14  0.5806946 -0.3250714  0.12846727  0.33887960
## Q15 -0.3316783 -0.3620736  0.52834991 -0.08033977
## Q16 -0.4069220 -0.4398788 -0.20043300  0.13094951
## Q17 -0.6251763  0.2665194  0.09778869  0.39147217
## Q18  0.4781092 -0.2045136  0.22455628  0.53160685
## Q19  0.5127333  0.1744786 -0.29166246  0.06097239
## Q20  0.3907534  0.5985140  0.31822294 -0.07475584
## Q21  0.5672222 -0.3077202  0.06856781 -0.36662308
## Q22 -0.3804674  0.1341635 -0.01415713 -0.25322693
## Q23 -0.3088811 -0.1922108  0.44485076 -0.07375981
## Q24  0.1581873  0.4544262  0.34650080  0.05169798
## Q25 -0.5549809  0.3399577  0.09318884  0.33462739
## Q26  0.5650082 -0.2333713  0.19231956  0.38389427
## Q27  0.2506611  0.3119157 -0.44034085  0.13298736
## Q28  0.4660091  0.1752175  0.03172224  0.06895569
## Q29 -0.4215349  0.2835606  0.30889584  0.26818215
## Q30  0.3045698 -0.2806700  0.04587713  0.38421190
## Q31 -0.3038096 -0.2951999  0.60015283 -0.07033962
## Q32  0.4717226  0.4318453  0.36617067 -0.12580811
##
## $cor
##          Dim.1      Dim.2       Dim.3       Dim.4
## Q1  -0.5688915  0.2594476  0.06654987  0.34960522
## Q2   0.5021178 -0.2373143  0.12284157  0.38417814
## Q3   0.4345612  0.3110200 -0.36950854  0.02163271
## Q4   0.3790587  0.4723304  0.35171209 -0.08316700
## Q5   0.5798081 -0.2455561 -0.06358979 -0.27455182
## Q6   0.5050051 -0.2625465  0.02225282  0.18236620
## Q7  -0.1814692 -0.3994541  0.46364367 -0.20124909
## Q8   0.4433948  0.5123844  0.38163200 -0.10676883
## Q9  -0.4847889  0.1204558  0.01028084  0.29329403
## Q10  0.5344008 -0.1634825  0.24158276  0.49608475
## Q11  0.2073377  0.2914197 -0.45068030  0.25594805
## Q12  0.4334654  0.3894721  0.38434674 -0.17975417
## Q13  0.5646683 -0.2802503 -0.09285473 -0.27570231
## Q14  0.5806946 -0.3250714  0.12846727  0.33887960
## Q15 -0.3316783 -0.3620736  0.52834991 -0.08033977
## Q16 -0.4069220 -0.4398788 -0.20043300  0.13094951
## Q17 -0.6251763  0.2665194  0.09778869  0.39147217
## Q18  0.4781092 -0.2045136  0.22455628  0.53160685
```

```
## Q19  0.5127333  0.1744786 -0.29166246  0.06097239
## Q20  0.3907534  0.5985140  0.31822294 -0.07475584
## Q21  0.5672222 -0.3077202  0.06856781 -0.36662308
## Q22 -0.3804674  0.1341635 -0.01415713 -0.25322693
## Q23 -0.3088811 -0.1922108  0.44485076 -0.07375981
## Q24  0.1581873  0.4544262  0.34650080  0.05169798
## Q25 -0.5549809  0.3399577  0.09318884  0.33462739
## Q26  0.5650082 -0.2333713  0.19231956  0.38389427
## Q27  0.2506611  0.3119157 -0.44034085  0.13298736
## Q28  0.4660091  0.1752175  0.03172224  0.06895569
## Q29 -0.4215349  0.2835606  0.30889584  0.26818215
## Q30  0.3045698 -0.2806700  0.04587713  0.38421190
## Q31 -0.3038096 -0.2951999  0.60015283 -0.07033962
## Q32  0.4717226  0.4318453  0.36617067 -0.12580811
##
## $cos2
##          Dim.1      Dim.2        Dim.3        Dim.4
## Q1  0.32363753 0.06731304 0.0044288853 0.1222238112
## Q2  0.25212231 0.05631808 0.0150900513 0.1475928452
## Q3  0.18884342 0.09673342 0.1365365582 0.0004679743
## Q4  0.14368548 0.22309599 0.1237013940 0.0069167499
## Q5  0.33617747 0.06029780 0.0040436615 0.0753787019
## Q6  0.25503012 0.06893065 0.0004951881 0.0332574304
## Q7  0.03293105 0.15956359 0.2149654501 0.0405011948
## Q8  0.19659894 0.26253776 0.1456429798 0.0113995829
## Q9  0.23502023 0.01450960 0.0001056956 0.0860213884
## Q10 0.28558417 0.02672654 0.0583622301 0.2461000811
## Q11 0.04298891 0.08492543 0.2031127326 0.0655094020
## Q12 0.18789226 0.15168850 0.1477224158 0.0323115623
## Q13 0.31885026 0.07854023 0.0086220015 0.0760117628
## Q14 0.33720617 0.10567144 0.0165038392 0.1148393858
## Q15 0.11001049 0.13109731 0.2791536322 0.0064544782
## Q16 0.16558554 0.19349334 0.0401733860 0.0171477733
## Q17 0.39084539 0.07103259 0.0095626284 0.1532504575
## Q18 0.22858840 0.04182580 0.0504255208 0.2826058383
## Q19 0.26289547 0.03044277 0.0850669913 0.0037176325
## Q20 0.15268822 0.35821901 0.1012658418 0.0055884355
## Q21 0.32174097 0.09469174 0.0047015447 0.1344124851
## Q22 0.14475541 0.01799985 0.0002004244 0.0641238788
## Q23 0.09540756 0.03694500 0.1978921964 0.0054405091
## Q24 0.02502321 0.20650317 0.1200628061 0.0026726810
## Q25 0.30800382 0.11557126 0.0086841601 0.1119754922
## Q26 0.31923424 0.05446216 0.0369868149 0.1473748141
## Q27 0.06283097 0.09729141 0.1939000666 0.0176856369
## Q28 0.21716450 0.03070119 0.0010063002 0.0047548877
## Q29 0.17769165 0.08040662 0.0954166400 0.0719216657
## Q30 0.09276277 0.07877568 0.0021047113 0.1476187847
## Q31 0.09230027 0.08714298 0.3601834227 0.0049476626
## Q32 0.22252219 0.18649034 0.1340809568 0.0158276803
##
## $contrib
##         Dim.1      Dim.2        Dim.3       Dim.4
## Q1  4.9587314  1.9974527  0.158163113  5.41759565
## Q2  3.8629848  1.6711872  0.538891696  6.54208332
```

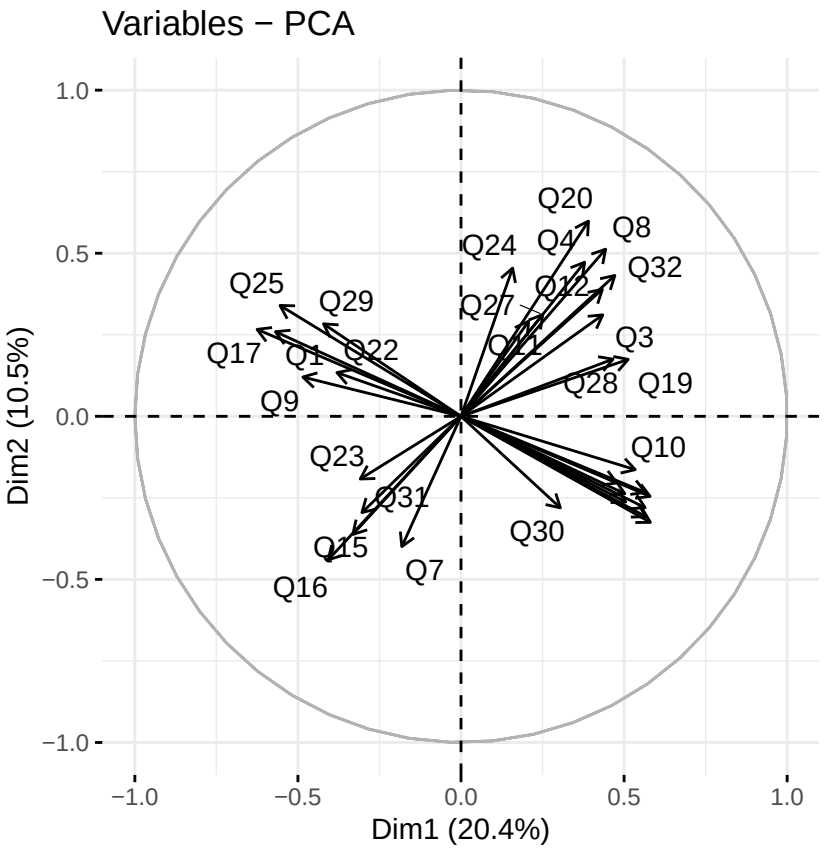
```
## Q3  2.8934339  2.8704752  4.875955405  0.02074306
## Q4  2.2015299  6.6201684  4.417589606  0.30658637
## Q5  5.1508667  1.7892817  0.144406109  3.34117652
## Q6  3.9075378  2.0454536  0.017684020  1.47414246
## Q7  0.5045652  4.7349029  7.676786071  1.79522382
## Q8  3.0122629  7.7905668  5.201161386  0.50528887
## Q9  3.6009489  0.4305590  0.003774572  3.81291579
## Q10 4.3756829  0.7930854  2.084215649 10.90843688
## Q11 0.6586704  2.5200841  7.253505134  2.90371776
## Q12 2.8788603  4.5012169  5.275421624  1.43221667
## Q13 4.8853816  2.3306092  0.307906509  3.36923707
## Q14 5.1666284  3.1357028  0.589380493  5.09027947
## Q15 1.6855662  3.8901922  9.969056487  0.28609608
## Q16 2.5370798  5.7417372  1.434660733  0.76007859
## Q17 5.9884814  2.1078270  0.341497912  6.79285815
## Q18 3.5024013  1.2411422  1.800782103 12.52656210
## Q19 4.0280496  0.9033613  3.037888616  0.16478483
## Q20 2.3394687 10.6298201  3.616377438  0.24770856
## Q21 4.9296726  2.8098905  0.167900250  5.95786114
## Q22 2.2179233  0.5341289  0.007157501  2.84230416
## Q23 1.4618221  1.0963089  7.067070807  0.24115169
## Q24 0.3834023  6.1277920  4.287649373  0.11846714
## Q25 4.7191938  3.4294709  0.310126298  4.96333680
## Q26 4.8912649  1.6161145  1.320862795  6.53241905
## Q27 0.9626878  2.8870333  6.924504981  0.78391951
## Q28 3.3273657  0.9110296  0.035936712  0.21076138
## Q29 2.7225680  2.3859925  3.407492377  3.18794267
## Q30 1.4212989  2.3375958  0.075162861  6.54323309
## Q31 1.4142126  2.5858878 12.862769718  0.21930617
## Q32 3.4094556  5.5339294  4.788261651  0.70156519
```

3.5.2 Visualizing results

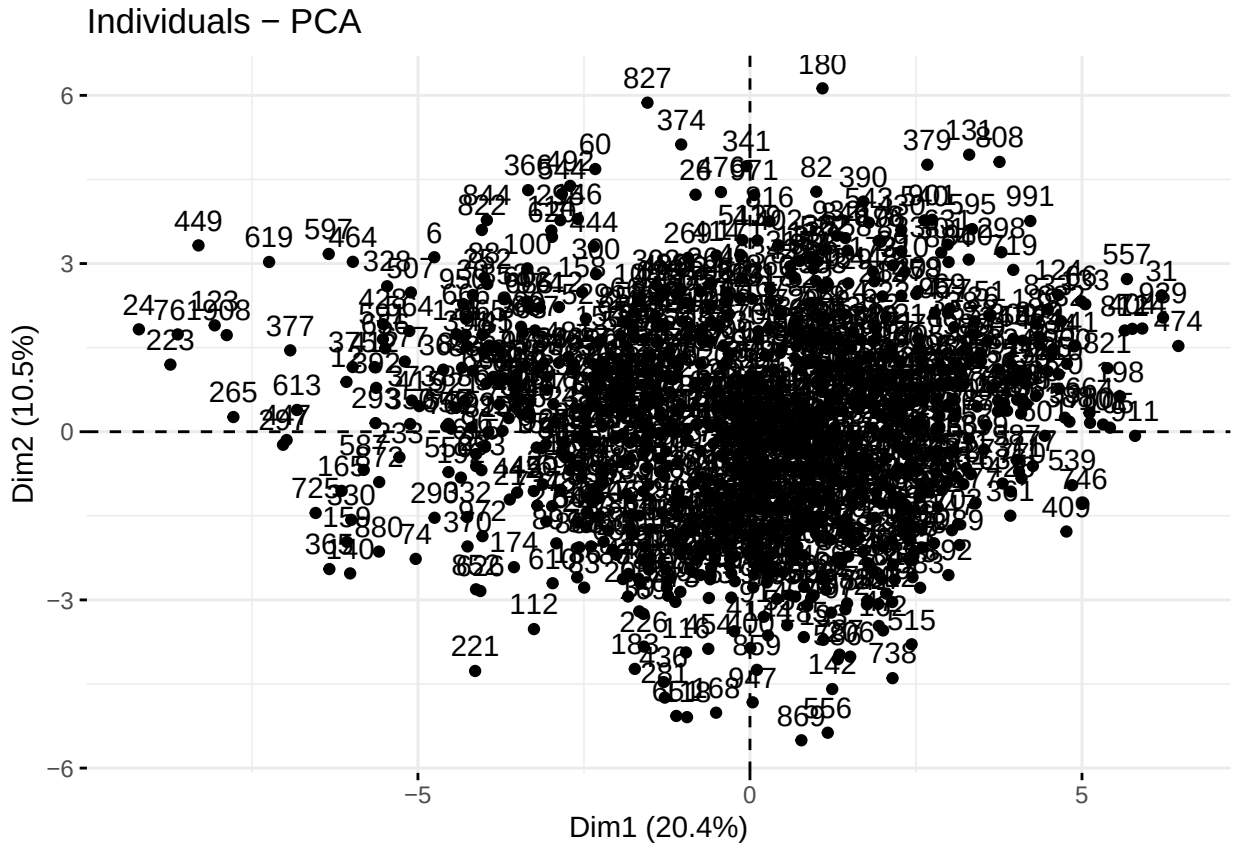
The visualization of the results can help in the analysis of the components. With the help of fviz_pca_var() and fviz_pca_ind() you can reproduce the PCA function with the original generated graphs. In addition, the two can be combined with the fviz_pca_biplot() function. This way we can see where the observations (the cars in this example) are located on the two most important dimensions. (The parameter repel = T is used to ensure that the labels are not overlapping instead they are dodged if they are too close to each other)

```
fviz_pca_var(pca_mod3, repel = T)
```

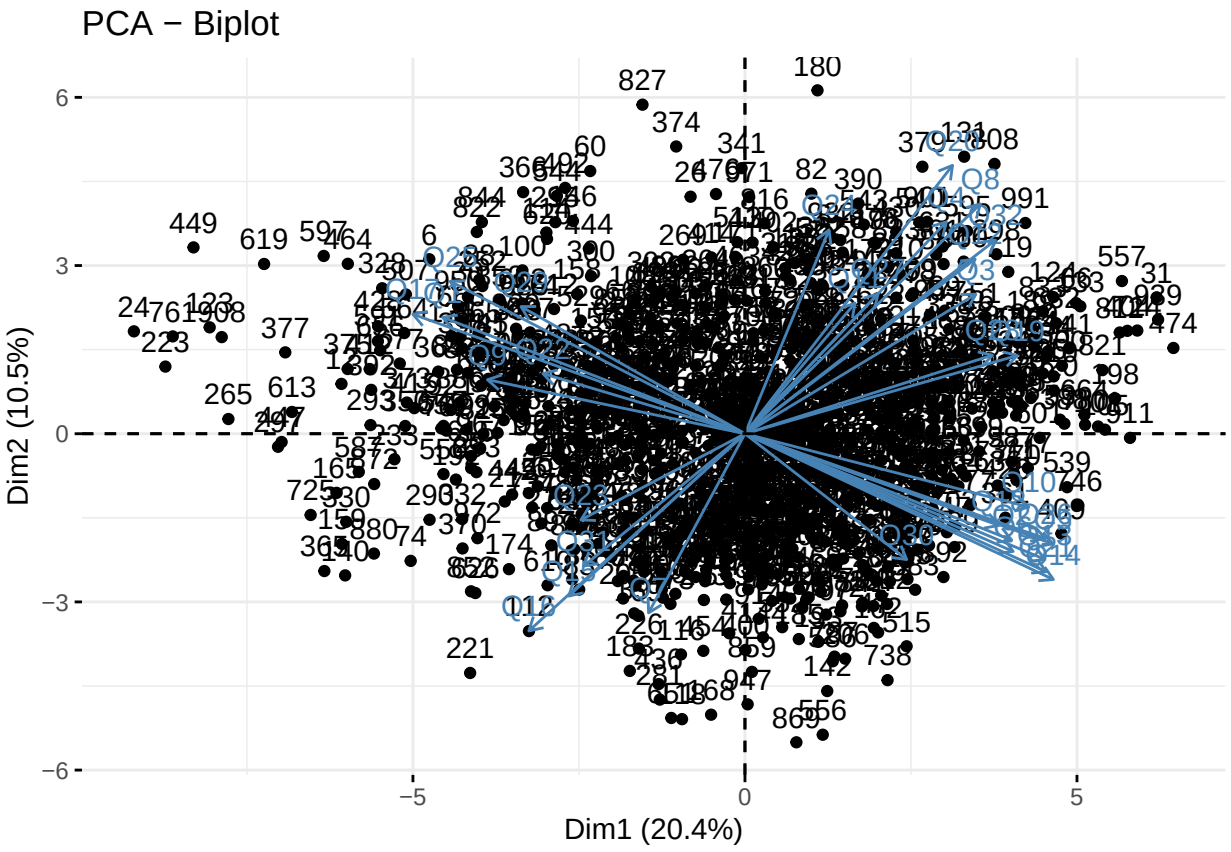
```
## Warning: ggrepel: 8 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```




```
fviz_pca_ind(pca_mod3)
```




```
fviz_pca_biplot(pca_mod3)
```

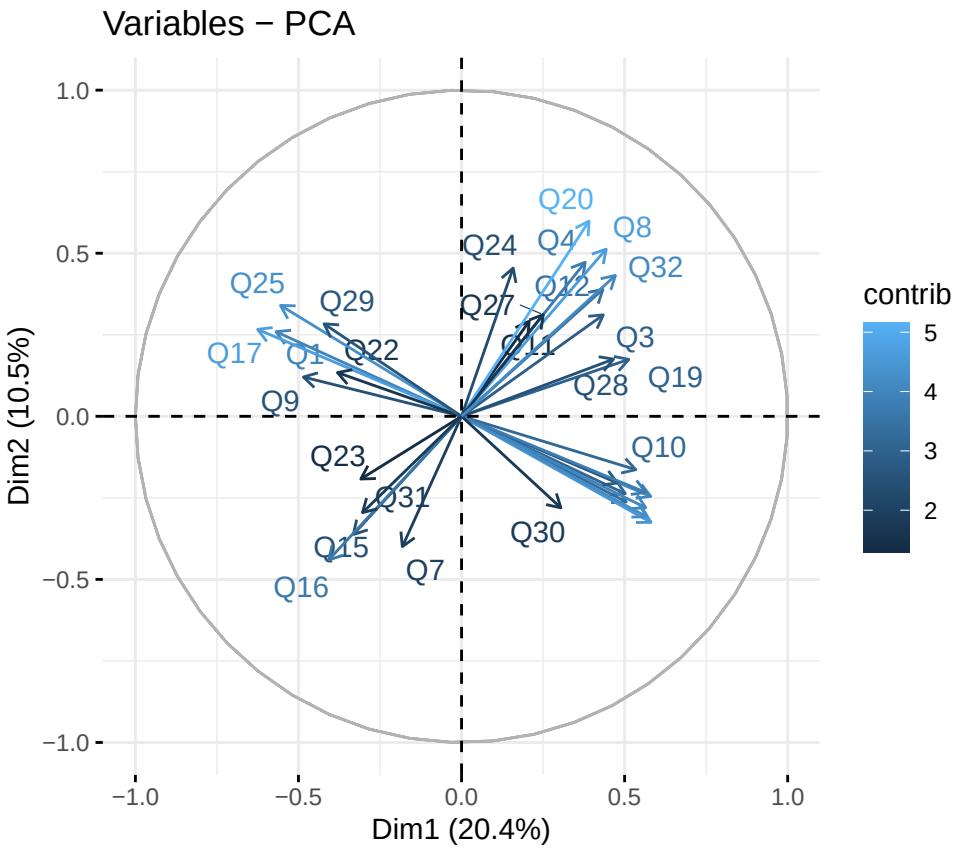


The figures can be further tuned by plotting the contribution of each variable or observation to the dimension via the parameters col.var = "contrib" and , col.ind = "contrib".

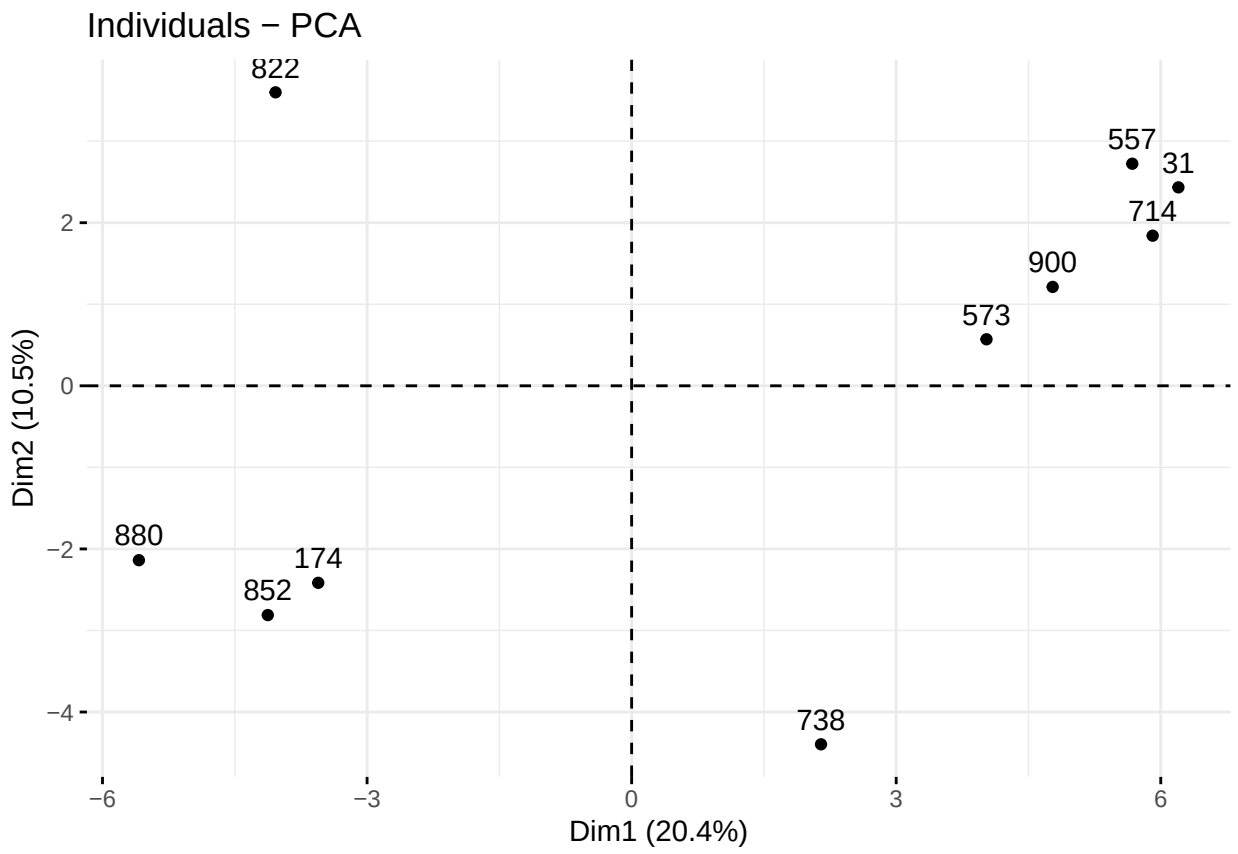
You can also use the select.ind = parameter to add only certain observations to the dimension. For example, cos^2 values show how well a given observation is represented by the given dimension. With the select.ind = list(cos2 = 10) parameter, you can specify that only the 10 observers with the highest cos^2 coefficient for the two dimensions are included. That is, the 10 most representative observations of the two dimensions described below.

```
fviz_pca_var(pca_mod3, repel = T, col.var = "contrib")
```

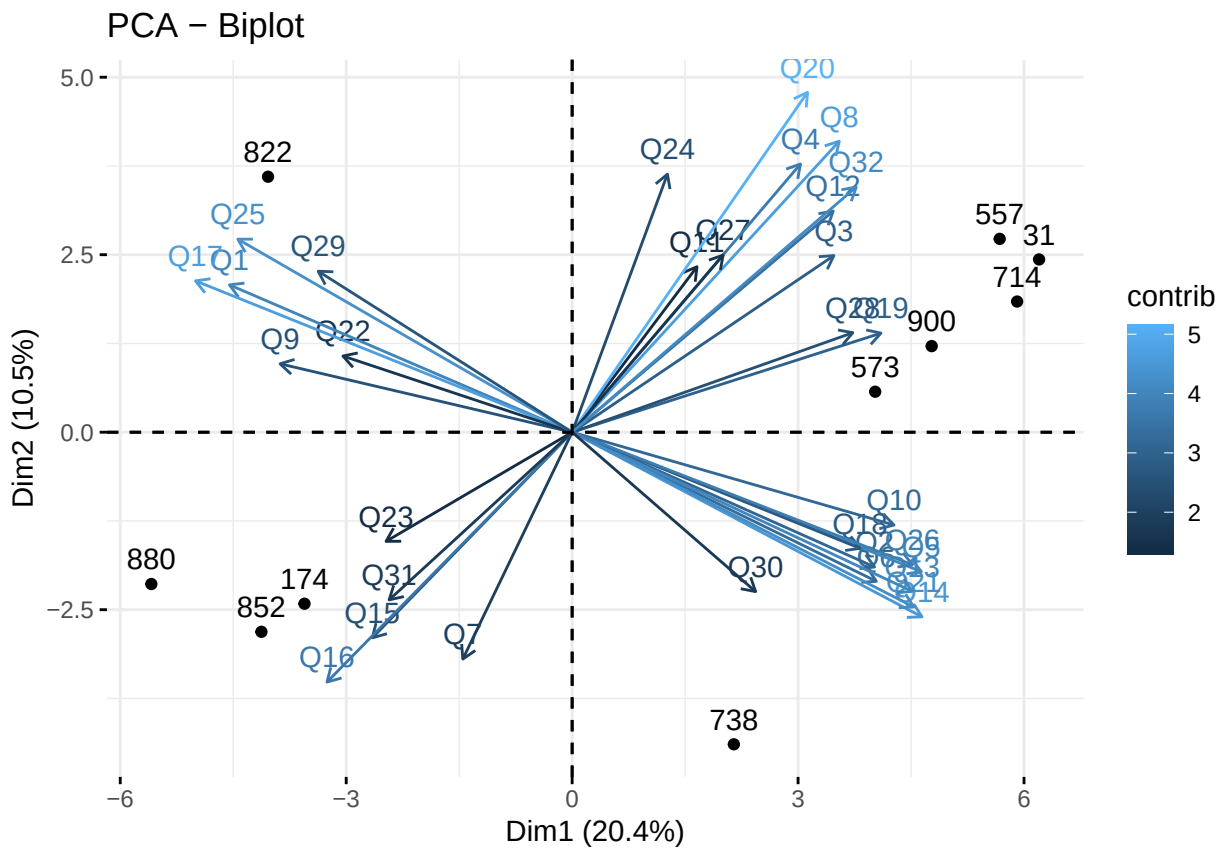
```
## Warning: ggrepel: 8 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```




```
fviz_pca_ind(pca_mod3, select.ind = list(cos2 = 10))
```

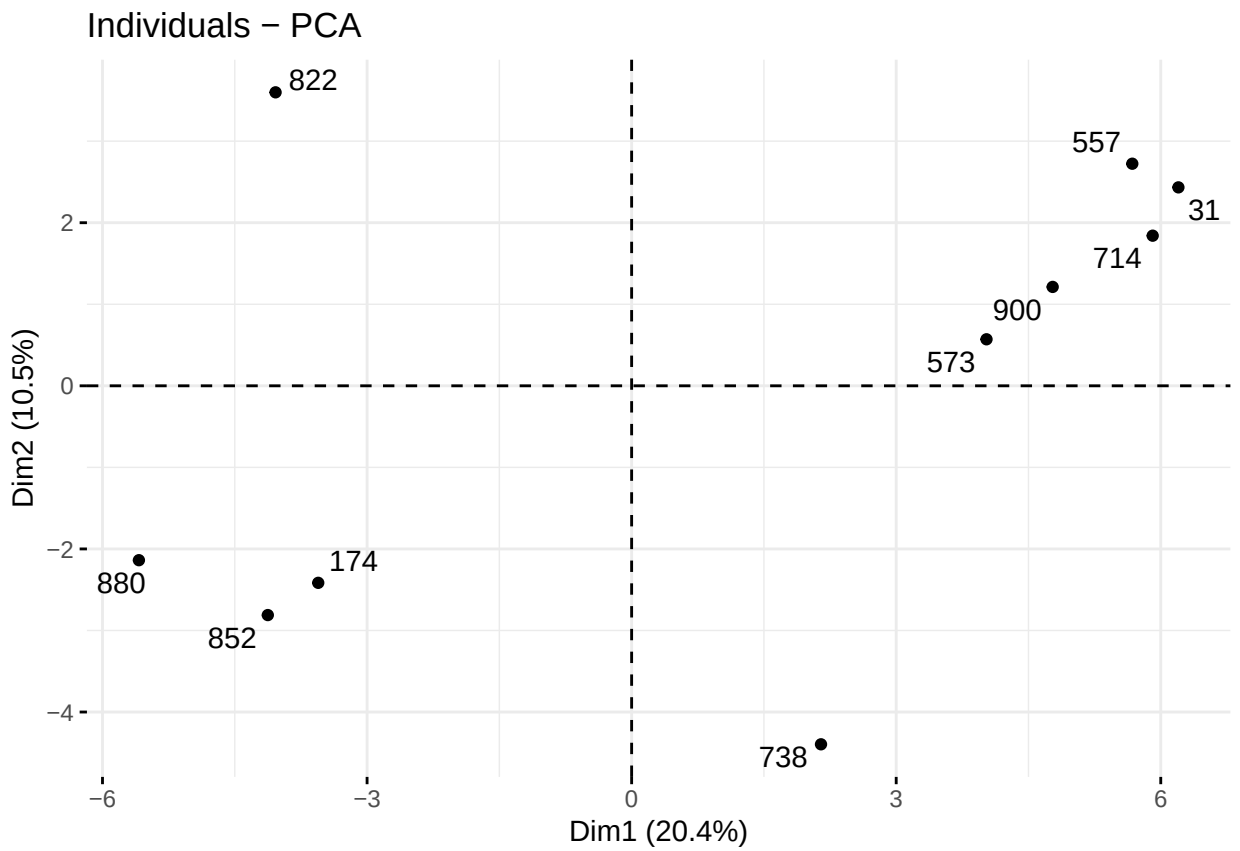



```
fviz_pca_biplot(pca_mod3, select.ind = list(cos2 = 10), col.var = "contrib")
```



This graph for example shows that the most typical members of high Dim.1, medium Dim.2 are cases 573 and 900, the most typical member of the low Dim.2, medium Dim.1 is probably case 738, and the most typical member of the low Dim.1 and high Dim.2 is case 822e. This may be important in the analysis of dimensions.

```
fviz_pca_ind(pca_mod3, select.ind = list(cos2 = 10), repel = T)
```



Another important tool for interpreting the dimensions is the bar chart** generated with **fviz_contrib(), which shows the relation of each variable to the dimensions.

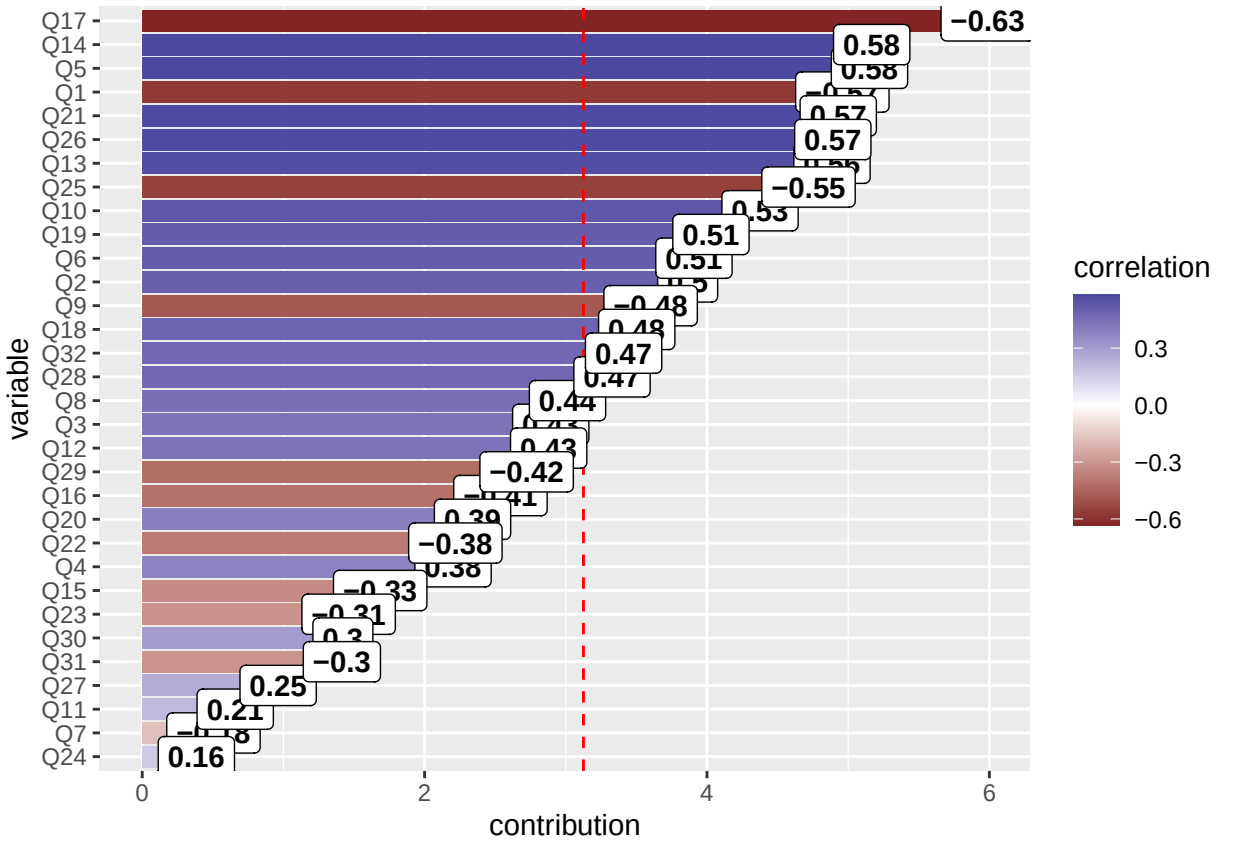
With the **axes = parameter you can specify which dimension** you are interested in. **The red dashed line** shows what would be the **expected contribution** if all the variables contributed to the dimension to the same degree.

This would only be truly informative if the we would also see the direction and size of the correlations as well. This is not included in the fviz_contrib() function itself, so we replace it with the **fviz_loadnings_with_cor() function** (our custom function), where the columns are sorted by correlation and the correlation is also included on the graph.

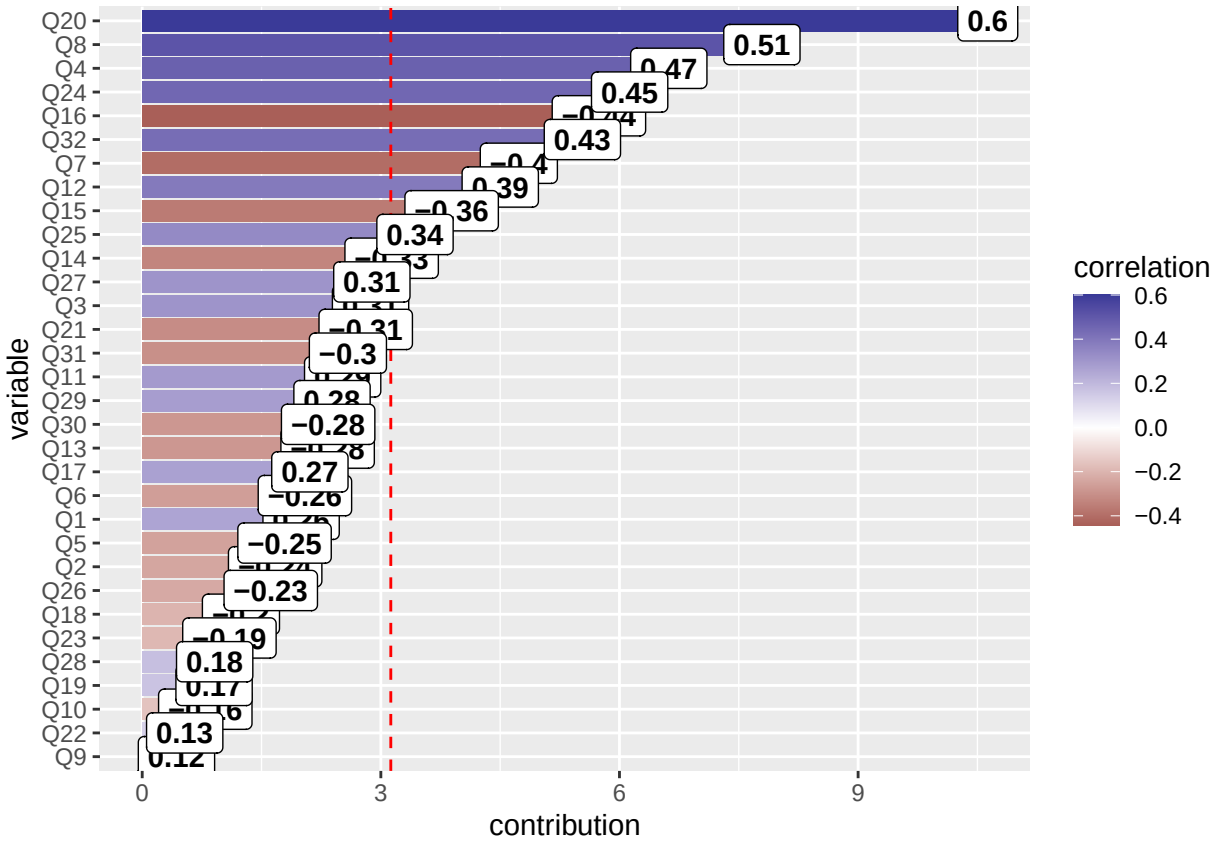
These graphs show that for Dim.1, Q14, Q17, Q5, and Q1 are the main contributing variables, for Dim.2, Q20, Q8, Q4 and Q24 are the main contributors, while for Dim.3, Q31, Q15, Q7 are the most important ones.

Based on these graphs, and based on the figure of representative cases, how do you think each dimension can be named? (You can find out exactly which questions are covered by each of the items numbered Q1 to Q32 in the data base presentation at the beginning of this document.)

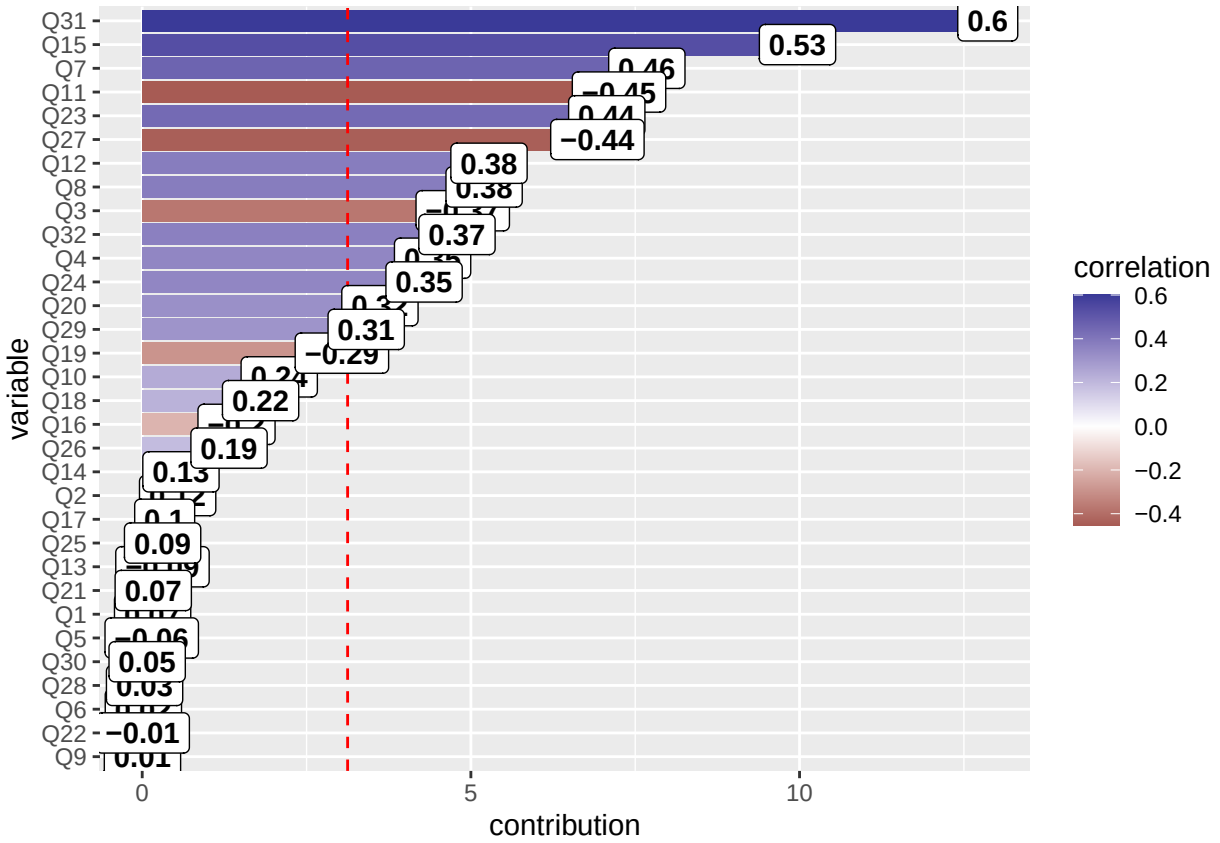
```
# original functions in factoextra fviz_contrib(pca_mod3,
# choice = 'var', axes = 1) fviz_contrib(pca_mod3, choice =
# 'var', axes = 2) fviz_contrib(pca_mod3, choice = 'var',
# axes = 3)

# using custom function for correlation color gradient
fviz_loadnings_with_cor(mod = pca_mod3, axes = 1)
```




```
fviz_loadnings_with_cor(mod = pca_mod3, axes = 2)
```



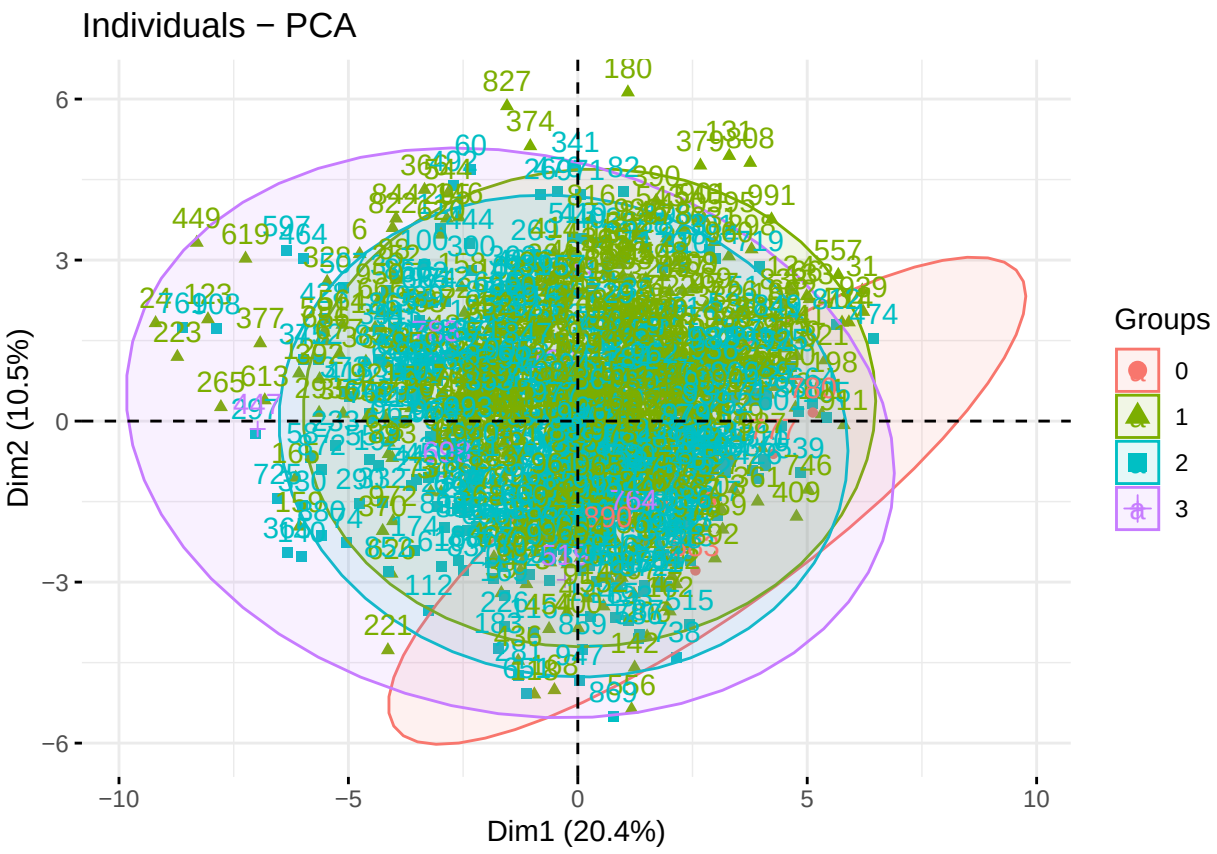

```
fviz_loadnings_with_cor(mod = pca_mod3, axes = 3)
```



Visualization can also be used to group observers according to their dimension values. This can be done with the parameter addEllipses = T.

This might help you to find subgroups of observations which are distinguished based on the values they take on the principal components.

```
fviz_pca_ind(pca_mod3, label = "ind", habillage = factor(hsq$gender),
    addEllipses = T)
```



4 Introduction to Exploratory Factor Analysis (EFA)

Exploratory Factor Analysis (EFA) is another dimension reduction technique that is similar to PCA.In PCA we assumed that there is only "common variance", so we wanted to maximize the extent to which we can cover/capture the variability in the original variables with no particular regard for making the components have a real-life relevance or interpretability. By contrast, in EFA we assume that the observed variables are manifestations of a smaller set of latent factors that are not directly observable, but the observed variables hold information about them. In this framework we do not assume that there is only "common variance".

Instead, in EFA we assume that any **observed variable consists of three components**: 1. Common variance (explained by the underlying factors), 2. Specific variance (variance that is specific to a particular item), 3. Error variance (for example due to measurement error). (2. and 3. together are also called uniqueness of an observed variable). While in PCA the extracted dimansions explain the same variance as the observed variables, so the dimensions explain all of the variance of the original variables, in EFA, not all of the variance of the original variables is explained by the factors. Because of this in EFA each observed variable has a number associated with it indicating the **proportion of variance** out of the total variance on this variable **explained by the factors**. This is called **communality**.

The most important steps in EFA:

- Assess factorability
- Factor extraction
- Selecting the ideal number of factors
- Factor rotation
- Interpretation of the factors

4.1 Factrability

Factorability of the data is one of the assumptions of exploratory factor analysis. When testing factorability, the question is whether there is enough correlation between the observed variables to allow EFA to be used. To test this we use two methods: either the **Bartlett sphericity test** or the **Kaiser-Meyer-Olkin test**.

First of all, however, we need **a correlation matrix of data** on which to run these tests. You could get this with the cor() function if you were working with continuous data, but in this dataset we are dealing with **ordinary data**, so we use another function called **mixedCor() from the psych package**. This function can be used to determine the **"Polychoric Correlation"** useful for ordinal data. The mixedCor() function is used to determine which are the continuous values and which are the ordinal values. Q1-Q32 are all ordinal, so only p = 1:32 is defined, and c= is set to NULL, because there are no continuous scalar continuous values.

It is important that the correlation matrix is stored in the $rho component of mixedCor() **, so we have to save it in a new data object. Save this to the object hsq_correl.

```
hsq_mixedCor <- mixedCor(hsq, c = NULL, p = 1:32)
hsq_correl = hsq_mixedCor$rho
```

Bartlett sphericity test

The idea behind the Bartlett test is to copare the actual **observed correlation matrix** of the observed variables with **a hypothetical null-correlation matrix**, where every correlation is set to 0 (this is also called the identity matrix). We test the null hypothesis that the two correlation maricies don't differ from each other. So if the test is significant, we can say that the two matrices are significantly different from each other, that is, that the observed variables correlate with each other. This means that the observed variables are factorable.

However, there is a serious drawback to using the Bartlett's test: that with large enough samples this test almost always returns significant results. So even though people tend to report this test, they do not consider this as a definitive indicator of factorability. The only time we should take the result of this test seriously is when the ratio of the number of observatiosn and the number of observed variables is lower than 5. In our case this value is about $1071/32 = 33.5$, so the Bartlett's test is not reliable.

```
bfi_factorability <- cortest.bartlett(hsq_correl)
```

```
## Warning in cortest.bartlett(hsq_correl): n not specified, 100 used
```

```
bfi_factorability
```

```
## $chisq
## [1] 1280.585
##
## $p.value
## [1] 9.770759e-71
##
## $df
## [1] 496
```

Kaiser-Meyer-Olkin (KMO) test

The KMO test **compares the partial correlation matrix** with the **regular correlation matrix**. In the partial correlation matrix we calculate the correlation of every pair of observed variables if we take out the effect of every other observed variable from this correlation. The KMO value shows the difference between the partial and the regular correlation matrix. In cases **where the variables hold a lot of common variance**, (there is a large chance that they are governed by the same latent factors) **partial correlations are low, so the KMO index will be large**. In the KMO test, values close to 1 indicate good factorability. The KMO index needs to be at least 0.6 for the variables to have reasonable factorability.

In our example, the **KMO is higher than 0.6 in all cases, and the total KMO is also higher than 0.6**, so the data seems to be factorable.

```
KMO(hsq_correl)
```

```
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = hsq_correl)
## Overall MSA =  0.88
## MSA for each item =
##   Q1   Q2   Q3   Q4   Q5   Q6   Q7   Q8   Q9  Q10  Q11  Q12  Q13  Q14  Q15  Q16
## 0.94 0.93 0.91 0.90 0.91 0.86 0.83 0.85 0.95 0.86 0.79 0.90 0.87 0.92 0.83 0.88
##  Q17  Q18  Q19  Q20  Q21  Q22  Q23  Q24  Q25  Q26  Q27  Q28  Q29  Q30  Q31  Q32
## 0.90 0.82 0.90 0.83 0.89 0.87 0.83 0.83 0.86 0.90 0.84 0.94 0.88 0.79 0.82 0.92
```

4.2 Faktor extraction

Factors will be extracted with the function fa(). The most commonly used factor extraction method is the **Maximum Likelihood Estimation** (mle) where the observed variables meet the multivariate normality assumption, while **Principal Axis Factoring** (paf) is the preferred method when the observed variables do not show multivariate normal distribution.

The function mvn() from the MVN package and the functions mvnorm.kur.test() and mvnorm.skew.test() from the ICS package can help to determine whether the data show a multivariate normal distribution. If the **p-value of these tests is lower than 0.05, it indicates the violation of the multivariate normality** assumption.

```
result <- mvn(hsq[, 1:32], mvnTest = "hz")
result$multivariateNormality
```

```
##            Test       HZ p value MVN
## 1 Henze-Zirkler 1.001426       0  NO
```

```
mvnorm.kur.test(na.omit(hsq[, 1:32]))
```

```
## Warning in pchisqsum(n * W.stat, df = dfs, a = chi.fac, method = "integration"):
## Package 'CompQuadForm' not found, using saddlepoint approximation
```

```
##
##  Multivariate Normality Test Based on Kurtosis
##
## data:  na.omit(hsq[, 1:32])
## W = 1707.3, w1 = 0.12457, df1 = 527.00000, w2 = 0.23529, df2 = 1.00000,
## p-value < 2.2e-16
```

```
mvnorm.skew.test(na.omit(hsq[, 1:32]))
```

```
##
##  Multivariate Normality Test Based on Skewness
##
## data:  na.omit(hsq[, 1:32])
## U = 441.69, df = 32, p-value < 2.2e-16
```

Above you can see that both the Henze-Zirkler test and the multivariate skewedness and kurtosis tests indicate that the assumption of normality is violated. So we will use the **paf extraction method**.

For factor analysis, the psych package fa() function is used. Within this, we can specify the factor extraction method within the **fm =** parameter. Here we use fm = "pa", because we would like to use the paf method, but if the multivariate normality had not been violated, we would have used the "mle" method instead.

With the **nfactors =** parameter we can specify how many factors we want to extract. Let's set this to 5 first, and we'll see below how to choose the ideal number of factors.

In the $communality component of the model object, we find the communalities associated with the variables. This is sorted from highest to lowest in the list. As mentioned above, the communalities indicate the proportion of variance in the observed variable explained by the extracted factors. The output shows that Q17 "I usually don't like to tell jokes or amuse people." is the best represented item in the 5-factor structure, with 68% of its total variance explained by the new factors. In contrast, Q22 "When I am feeling sad or upset, I usually lose my sense of humor." is the least represented item, with only 25% of its variance explained by the current factor structure.

In order for the factor structure to work well, it is usually necessary to exclude the poorly represented items. This is especially important when the sample size is small. **If the number of observers is below 250, MacCallum et al. suggest that the average communality of the items should be at least 0.6.** In our case we can be more lenient, because our item size is larger, but in a more factor analysis case it may still be worth considering the exclusion of poorly represented (low communality) items.

MacCallum, R. C., Widaman, K. F., Zhang, S., & Hong, S. (1999). Sample size in factor analysis. Psychological methods, 4(1), 84.

```
EFA_mod1 <- fa(hsq_correl, nfactors = 5, fm = "pa")

# Sorted communality
EFA_mod1_common <- as.data.frame(sort(EFA_mod1$communality, decreasing = TRUE))
EFA_mod1_common
```

```
##     sort(EFA_mod1$communality, decreasing = TRUE)
## Q17                                     0.6754689
## Q20                                     0.6701555
## Q25                                     0.6528782
## Q8                                      0.6265007
## Q21                                     0.6240345
## Q10                                     0.6140024
## Q15                                     0.6047847
## Q18                                     0.5909926
## Q14                                     0.5796418
## Q13                                     0.5557827
## Q32                                     0.5458051
## Q26                                     0.5424384
## Q1                                      0.5311824
## Q31                                     0.5176279
## Q12                                     0.4858949
## Q19                                     0.4806134
## Q5                                      0.4741704
## Q4                                      0.4663425
## Q2                                      0.4442162
## Q16                                     0.4360995
## Q6                                      0.4257846
## Q29                                     0.4025312
## Q7                                      0.3978300
## Q11                                     0.3930955
## Q3                                      0.3908073
## Q24                                     0.3558342
## Q23                                     0.3419339
## Q27                                     0.3300415
## Q9                                      0.3241293
```

```
## Q30                                    0.2867962
## Q28                                    0.2663399
## Q22                                    0.2543910
```

```
mean(EFA_mod1$communality)
```

```
## [1] 0.4777546
```

4.3 Choosing the ideal number of factors

Similarly to PCA, we need to determine how many factors we want to extract from the data. As we have seen with the PCA, we can use the scree test, the Kaiser-Guttman criterion or the parallel test to decide. In addition, the psych package offers two other methods to solve the problem: the very simple structure (VSS) criterion, and Wayne Velicer's Minimum Average Partial (MAP) criterion (the vss() function in the psych package). An acceptable solution is to run all of these and consider all factor numbers offered by these methods, or the factor number that is suggested by the most methods.

In the example below, we use the fa.parallel function of the psych package and the nfactors function to decide, according to the key criteria, which factor retention would be ideal.

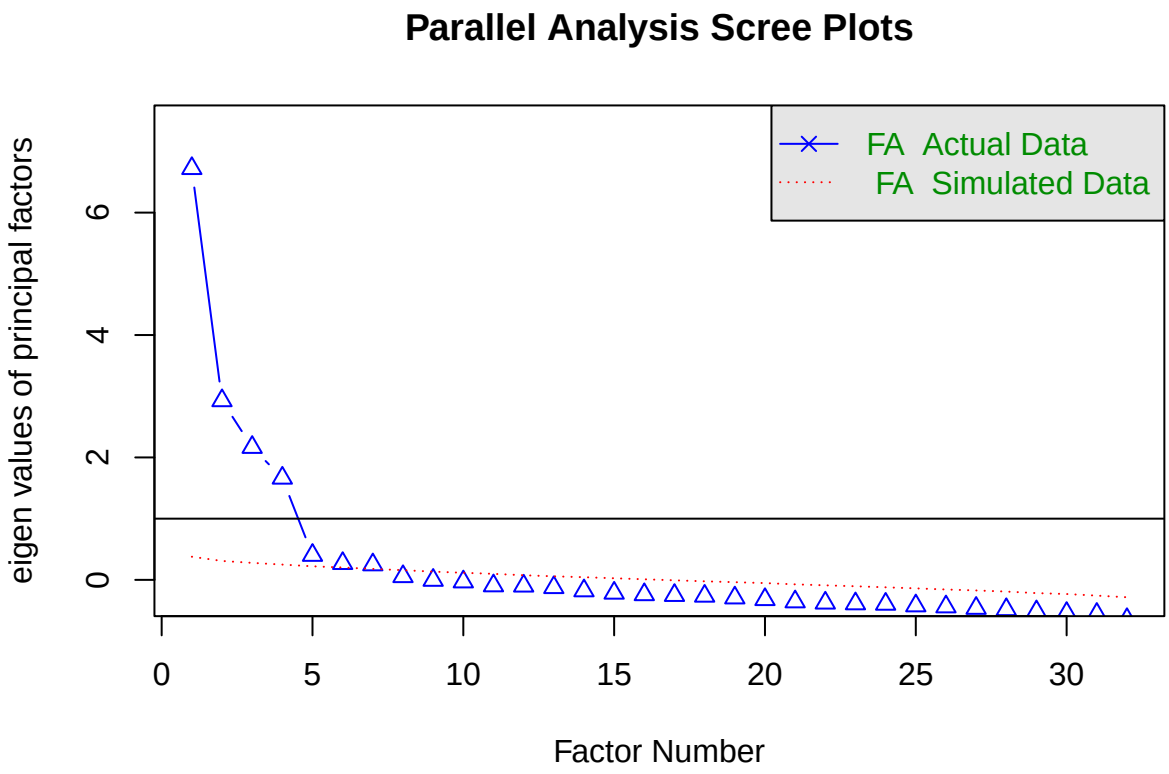
The ideal factor sets suggested by the different techniques are the following:

- scree test: 4
- Kaiser-Guttman criterion: 4
- Parallel test: 7
- VSS: 3-4
- MAP: 4

Based on these, it seems that the best technique is to use 4 latent factors to describe the variability of the data. Below, we will discuss this 4-factor model and examine the communication table.

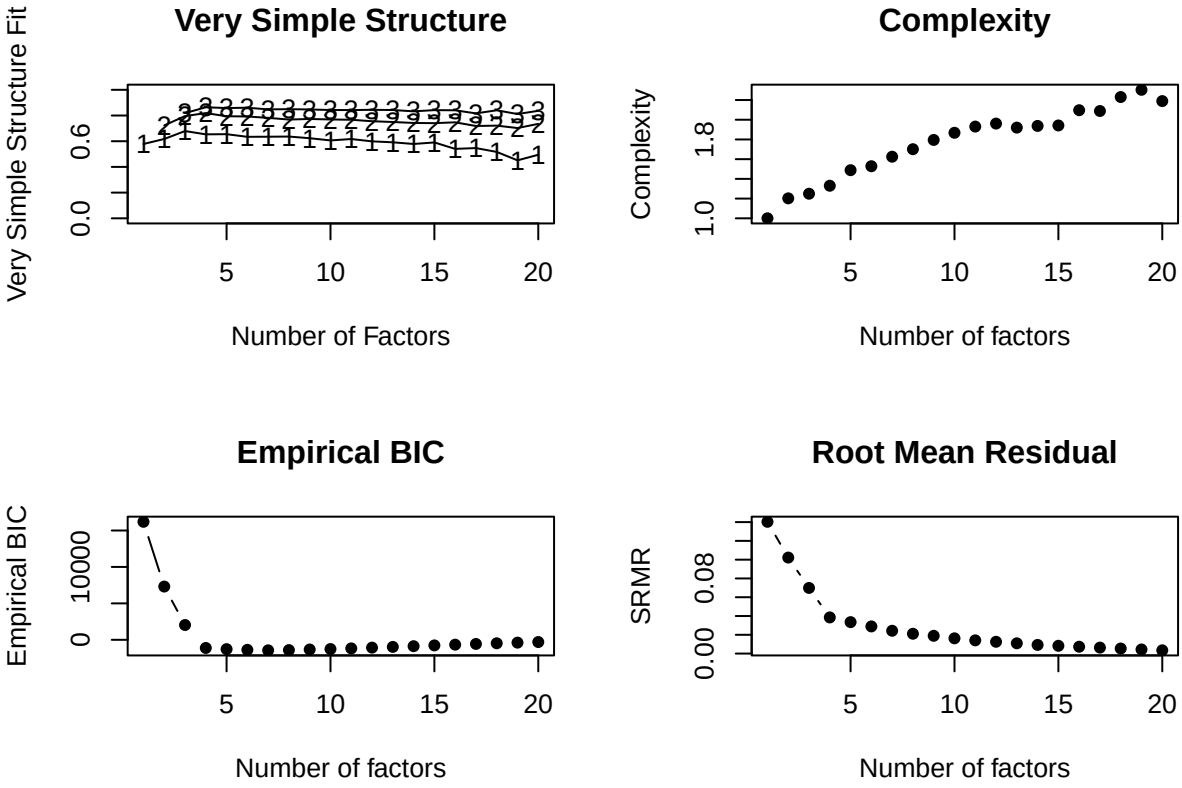
In the factor analysis, it is very common to repeat the process over and over again with different sets of observed variables, different factor numbers and rotational methods until the final accepteble factor structure is obtained. In EFA the final factor structure is usually chosen based on how well the factors are interpretable in the theoretical framework.

```
fa.parallel(hsq_correl, n.obs = nrow(hsq), fa = "fa", fm = "pa")
```




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

```
nfactors(hsq_correl, n.obs = nrow(hsq))
```




```
##
## Number of factors
## Call: vss(x = x, n = n, rotate = rotate, diagonal = diagonal, fm = fm,
##     n.obs = n.obs, plot = FALSE, title = title, use = use, cor = cor)
## VSS complexity 1 achieves a maximimum of 0.68  with  3  factors
## VSS complexity 2 achieves a maximimum of 0.82  with  4  factors
## The Velicer MAP achieves a minimum of 0.01  with  4  factors
## Empirical BIC achieves a minimum of  -1440.89  with  7  factors
## Sample Size adjusted BIC achieves a minimum of  -204.78  with  14  factors
##
## Statistics by number of factors
##    vss1 vss2   map dof chisq     prob sqresid  fit  RMSEA  BIC SABIC complex
## 1  0.58 0.00 0.034 464  8804  0.0e+00    40.0 0.58 0.1345 5602  7075     1.0
## 2  0.62 0.72 0.027 433  6069  0.0e+00    26.3 0.72 0.1145 3081  4457     1.2
## 3  0.68 0.80 0.019 403  4167  0.0e+00    17.2 0.82 0.0970 1386  2666     1.2
## 4  0.65 0.82 0.011 374  2283 1.5e-270    11.5 0.88 0.0717 -298   890     1.3
## 5  0.65 0.79 0.012 346  1980 9.3e-227    10.3 0.89 0.0690 -407   692     1.5
## 6  0.63 0.79 0.013 319  1632 8.3e-175     9.3 0.90 0.0644 -570   443     1.5
## 7  0.63 0.78 0.014 293  1297 3.6e-126     8.4 0.91 0.0587 -725   206     1.6
## 8  0.64 0.77 0.016 268  1099 4.6e-101     7.8 0.92 0.0559 -750   101     1.7
## 9  0.62 0.77 0.017 244   946  3.2e-83     7.3 0.92 0.0538 -738    37     1.8
## 10 0.61 0.77 0.020 221   708  2.5e-52     6.9 0.93 0.0471 -817  -115     1.9
## 11 0.62 0.77 0.022 199   584  1.5e-39     6.4 0.93 0.0441 -789  -157     1.9
## 12 0.60 0.76 0.025 178   500  2.7e-32     6.1 0.94 0.0426 -729  -163     2.0
## 13 0.59 0.75 0.028 158   399  7.8e-23     5.8 0.94 0.0392 -691  -189     1.9
## 14 0.58 0.74 0.032 139   313  2.0e-15     5.5 0.94 0.0355 -646  -205     1.9
```

```
## 15 0.59 0.74 0.036 121   269  3.4e-13     5.2 0.94 0.0350 -566  -182     1.9
## 16 0.54 0.75 0.040 104   223  1.3e-10     4.9 0.95 0.0339 -495  -165     2.1
## 17 0.55 0.72 0.046  88   170  4.1e-07     4.8 0.95 0.0305 -438  -158     2.1
## 18 0.52 0.72 0.052  73   129  6.4e-05     4.5 0.95 0.0277 -375  -143     2.2
## 19 0.45 0.70 0.058  59    73  1.0e-01     4.4 0.95 0.0155 -334  -147     2.3
## 20 0.49 0.74 0.066  46    49  3.6e-01     4.2 0.96 0.0076 -269  -123     2.2
##    eChisq   SRMR eCRMS  eBIC
## 1   19394 0.1403 0.145 16192
## 2   10295 0.1022 0.109  7307
## 3    4812 0.0699 0.078  2031
## 4    1460 0.0385 0.044 -1120
## 5    1105 0.0335 0.040 -1282
## 6     824 0.0289 0.036 -1377
## 7     581 0.0243 0.032 -1441
## 8     439 0.0211 0.029 -1410
## 9     353 0.0189 0.027 -1331
## 10    260 0.0162 0.024 -1265
## 11    196 0.0141 0.022 -1177
## 12    156 0.0126 0.021 -1073
## 13    118 0.0109 0.019  -972
## 14     84 0.0092 0.017  -875
## 15     68 0.0083 0.017  -767
## 16     52 0.0073 0.016  -665
## 17     40 0.0064 0.015  -567
## 18     29 0.0054 0.014  -475
## 19     19 0.0044 0.013  -388
## 20     12 0.0035 0.011  -305
```

```
EFA_mod2 <- fa(hsq_correl, nfactors = 4, fm = "pa")

EFA_mod2_common <- as.data.frame(sort(EFA_mod2$communality, decreasing = TRUE))
EFA_mod2_common
```

```
##     sort(EFA_mod2$communality, decreasing = TRUE)
## Q20                                     0.6700371
## Q17                                     0.6697004
## Q25                                     0.6530751
## Q8                                      0.6301917
## Q21                                     0.6195670
## Q10                                     0.6145714
## Q18                                     0.5907715
## Q14                                     0.5678837
## Q32                                     0.5457790
## Q26                                     0.5449964
## Q13                                     0.5427269
## Q1                                      0.5316347
## Q31                                     0.5224554
## Q15                                     0.5181019
## Q12                                     0.4849208
## Q5                                      0.4602084
## Q4                                      0.4523891
## Q2                                      0.4241464
## Q29                                     0.4033447
## Q7                                      0.3973521
## Q3                                      0.3950130
```

```
## Q16                                    0.3775542
## Q19                                    0.3656149
## Q6                                     0.3630614
## Q11                                    0.3313680
## Q27                                    0.3311449
## Q9                                     0.3143381
## Q24                                    0.2969414
## Q23                                    0.2792367
## Q30                                    0.2700595
## Q28                                    0.2369152
## Q22                                    0.1856682
```

```
mean(EFA_mod2$communality)
```

```
## [1] 0.4559615
```

4.4 Factor rotation

The aim of factor rotation is to improve the interpretability of the factors. This way we can avoid that the whole factor structure consists of 1 or 2 very dominant factors, which are very strongly loaded by the items (observed variables), while the other factors are less relevant. In the factor rotation, the original items remain in the "factor space", but the dimensional axes of the factors are rotated to better fit each item group.

There are many known methods of factor rotation, but they can be grouped into two categories: orthogonal and oblique. With orthogonal methods (such as Quartimax, Equimax, or the **Varimax** method most commonly used in psychology), the factor dimensions remain perendicular (orthogonal), this means that the different factors will not correlate with each other, and their explained variance will not overlap. In the case of oblique methods (such as **Direct Oblimin** or Promax), however, it is allowed that the factors correlate with each other. In the exploratory factor analysis series, several rotation methods can be tested, but here the theoretical justification is important. Is it theoretically plausible that the latent factors correlate with each other? If so, then oblique methods should be used. (Uncorrelated factors are easier to interpret most of the times).

Orthogonal rotation options:

- **Varimax** rotation is the most commonly used. It minimizes the number of items with extreme loadings. This often produces factors that are easier to interpret compared with the other rotations.
- **Quantimax** is less often used. This rotation tries to minimize the number of factors needed to explain all the items well. It often creates a "general factor" on which all items have moderate to high loading, which is often hard to interpret.

Oblique rotation options:

- **Promax** rotation is the most commonly used because it is computationally faster and handles large datasets well compared to the alternatives.
- **Direct** oblimin is also an alternative, less commonly used, but still accepted in the literature.

The default method of factor rotation is Direct Oblimin ("oblimin") in the fa() function. Try also the Promax ("promax") and Varimax ("varimax") methods. It is worthwhile to try several rotation methods in the exploratory factor analysis and choose the one that produces the best factor structure in our theoretical framework.

```
EFA_mod2$rotation
```

```
## [1] "oblimin"
```

```
EFA_mod_promax <- fa(hsq_correl, nfactors = 4, fm = "pa", rotate = "promax")

EFA_mod_varimax <- fa(hsq_correl, nfactors = 4, fm = "pa", rotate = "varimax")
```

4.5 Interpreting factors

The interpretation of the factors in not a trivial task. It often requires a lot of domain-specific knowledge.

As stated earlier, EFA is an iterative process. It often takes many runs and changing the settings of the factor procedure to get to the final model and until we have arrived at the final factor structure we cannot finalize our interpretation of the factors. Nevertheless, trying to interpret the meaning of the factors, that is, trying to figure out what latent constructs they could reflect, is important in fine-tuning the EFA.

This is an exploratory process so we shouldn't be afraid of exploring different options, parameter settings, extraction and rotation methods, item configurations. However, it is useful in the long run to keep a record of what were the different settings/options that were tried during this process until we get to the final factor structure. This could improve the transparency and reproducibility of our research, and it could also help us save time if we know what we have already tried before.

The interpretation of the factors is usually done by using one of the correlation matrixes. If we did not use factor rotation this should be the Factor Matrix, if we used rotation this should be either the Rotated factor matrix (in the case of orthogonal rotations), or the Pattern Matrix (in the case of oblique rotations).

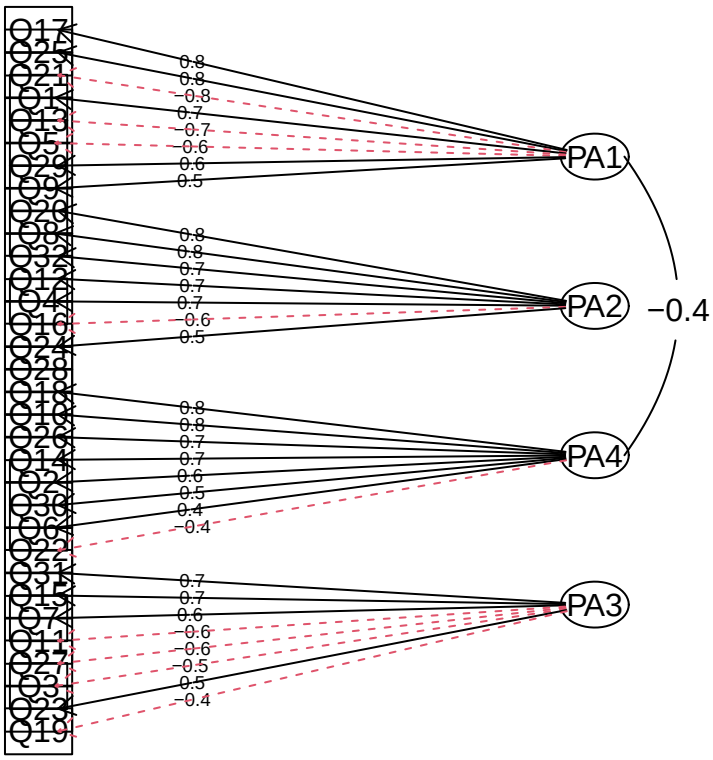
When we use oblique rotations (like promax or direct oblimin) in some outputs we also see the structure matrix. The difference between the pattern matrix and the structure matrix is that the pattern matrix includes regression coefficients as loadings, while the Structure matrix includes correlation coefficients. The regression coefficients in the Pattern matrix is similar to what we would get if we built a linear regression model to predict the observed variable by the factor scores. If no rotation is used or in the case of orthogonal rotation the factors are uncorrelated, so there is no shared variance explained by them so this has the same meaning as a simple correlation coefficient in those cases. However, in oblique rotations, the factors are allowed to correlate, which results in some overlap in the portion of variance they explain of the variability of each observed variable. The pattern matrix accounts for this shared variance by using a regression coefficient instead of the correlation coefficient, this way, the meaning of the loadings in that table is the portion of the variance of an observed variable explained uniquely by the given factor, while taking into account the variance explained by all other factors. While the structure matrix contains simple correlation coefficients, which does not take into account the variance explained by the other factors.

The interpretation of the factors is usually done by looking at which items have high loadings on a given factor, and figuring out from this information what could that factor mean.

For example the first component seems to hold information about generally how often and how readily does the person joke/use humor, while the second component seems to be mostly related to questions related to self-harm by humor, etc.

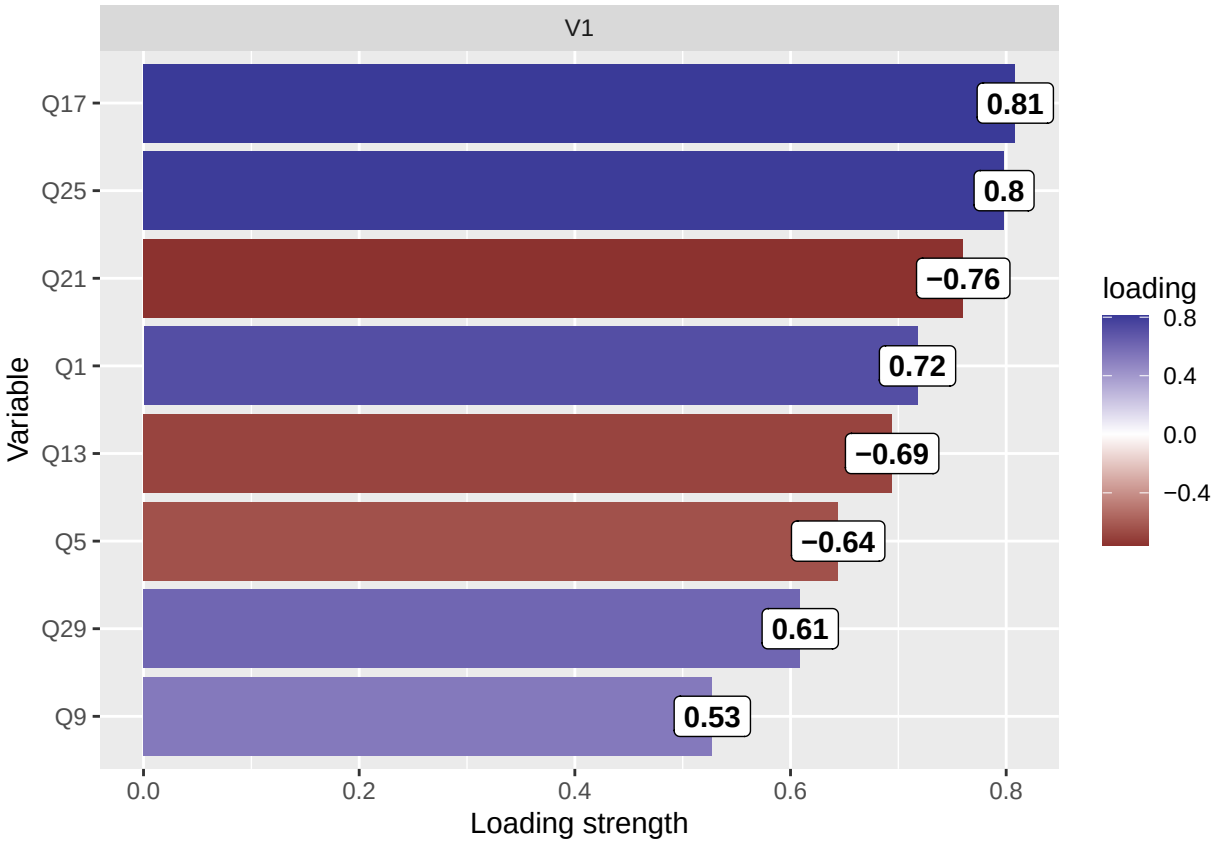
The sign of the loading is important here as well in the interpretation. For example Q17. "I usually don't like to tell jokes or amuse people". Has a positive loading on factor 1, while Q21. "I enjoy making people laugh" has a negatie loading. This confirms that the first factor is about general attitude about joking and humor.

```
fa.diagram(EFA_mod2)
```

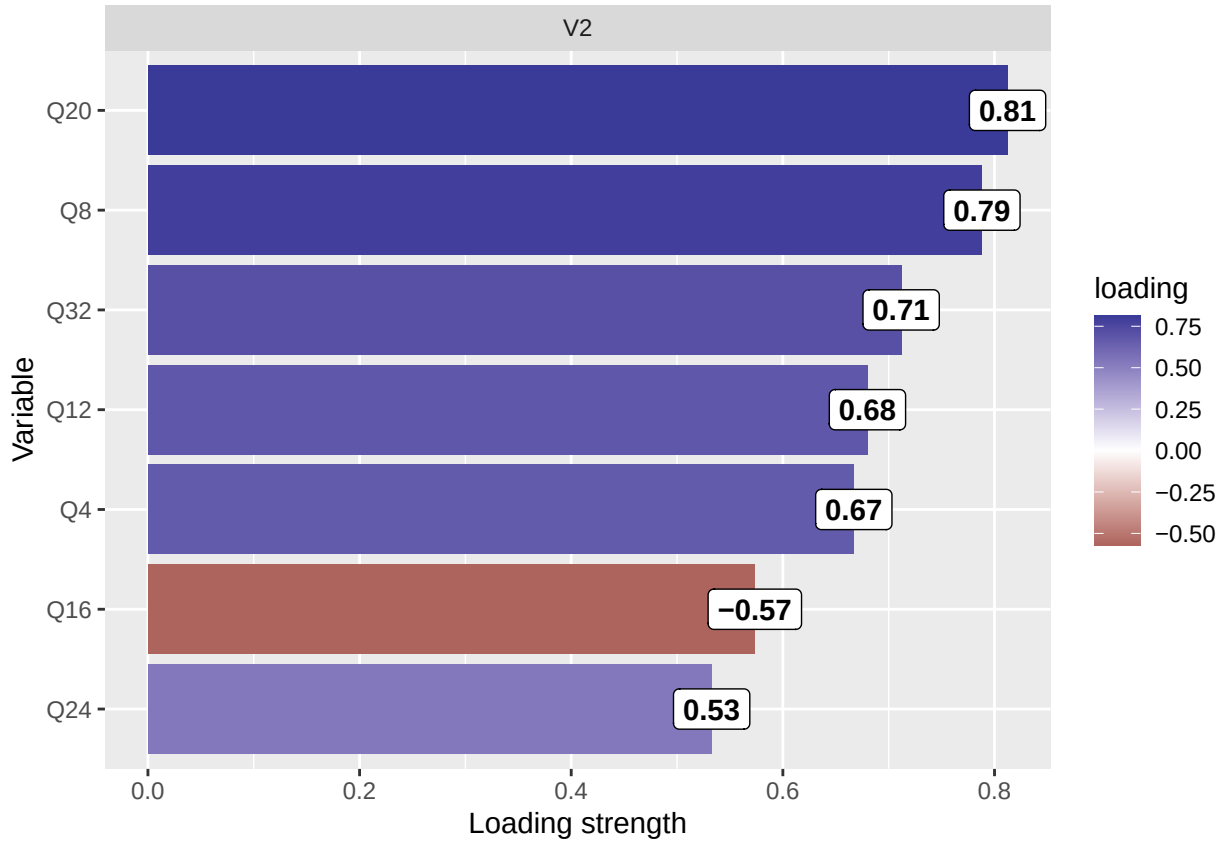



```
fviz_loadnings_with_cor(EFA_mod2, axes = 1, loadings_above = 0.4)
```

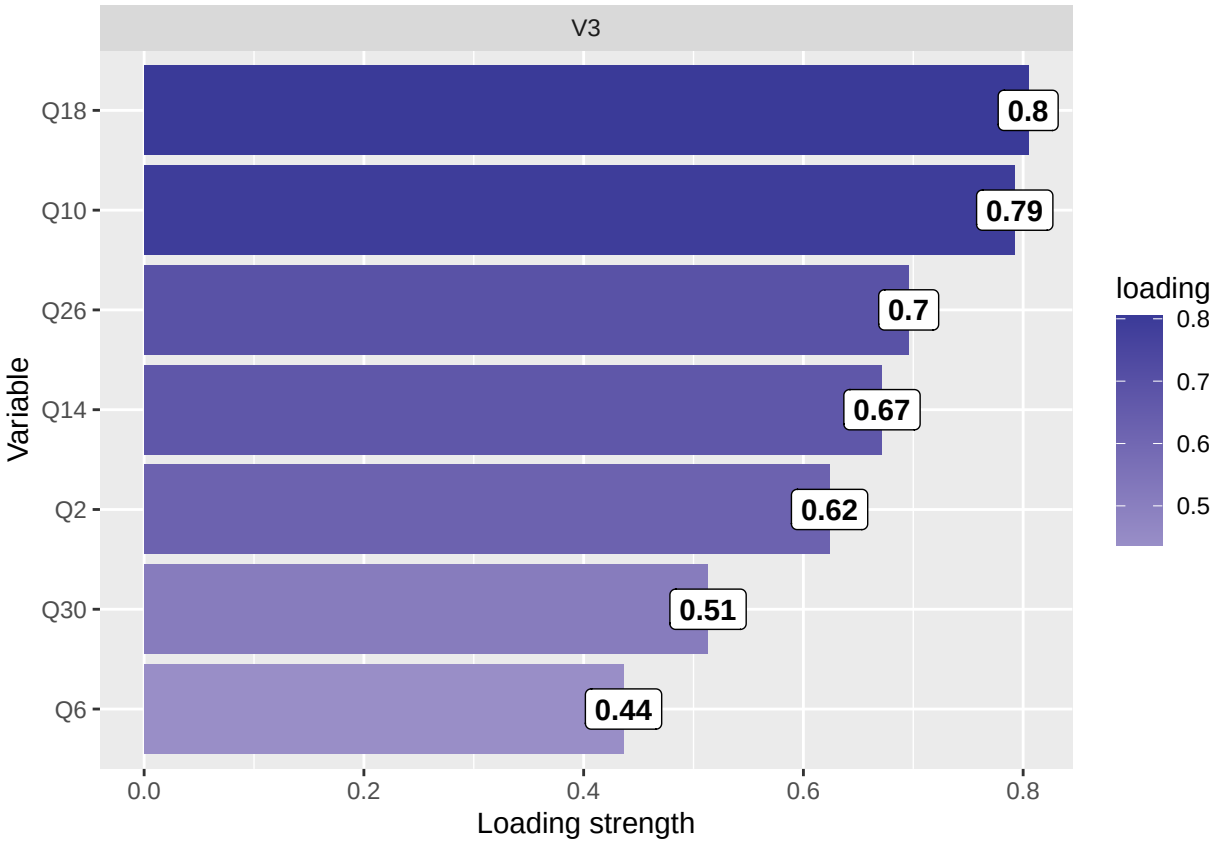
```
## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if `.name_repair` is
## Using compatibility `.name_repair`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```



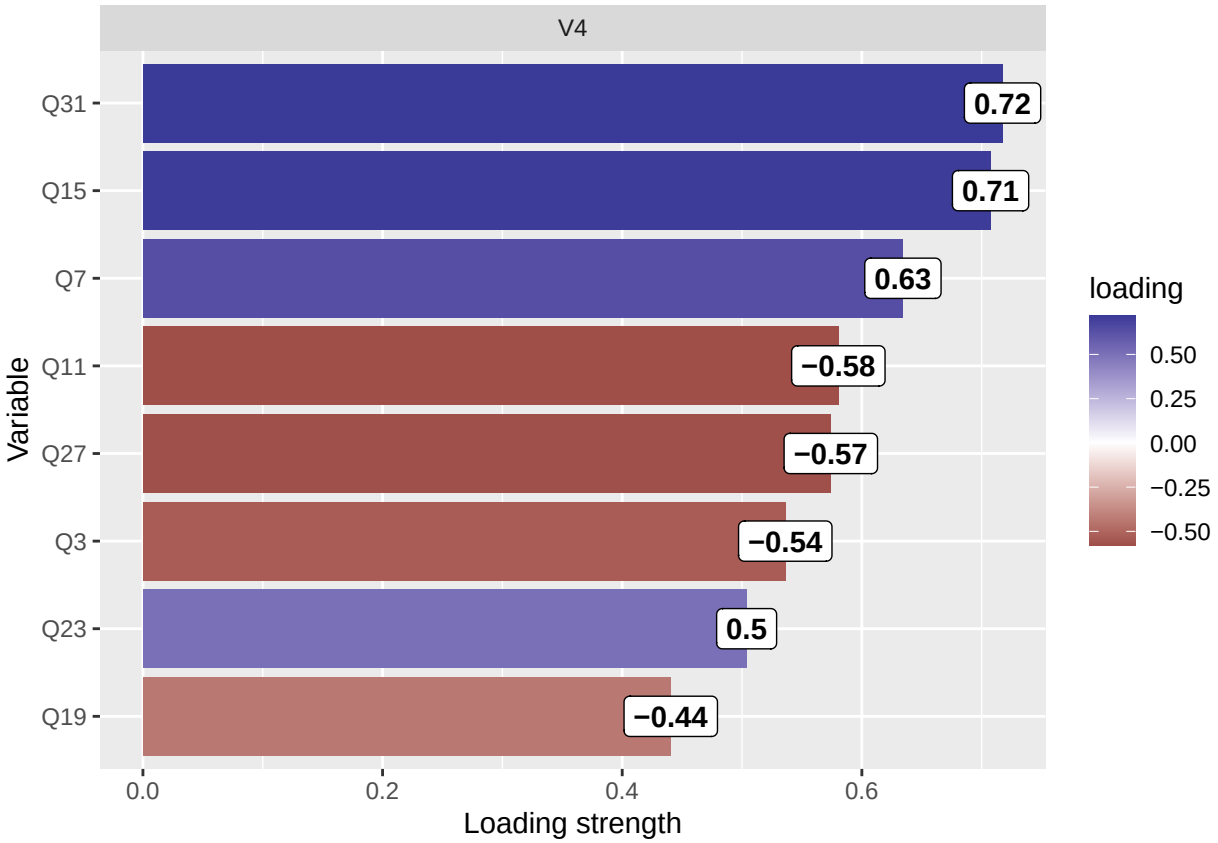

```
fviz_loadnings_with_cor(EFA_mod2, axes = 2, loadings_above = 0.4)
```




```
fviz_loadnings_with_cor(EFA_mod2, axes = 3, loadings_above = 0.4)
```




```
fviz_loadnings_with_cor(EFA_mod2, axes = 4, loadings_above = 0.4)
```



4.6 Which items to keep in the analysis?

PCA and EFA tries to account for the variance within all variables that are entered into the analysis. According to the communalities table this is achieved better for some variables (the ones with higher post-extraction communalities) than for others. It is possible however that the factors underlying most observed variables are not related to some of the observed variables in the dataset or that the common variance component of some observed variables is not high enough, and so they seem to be loading on all factors to a little bit.

This makes it hard to interpret the factors, and it makes the factors less clear. Thus, sometimes it is a good idea to exclude some of the items from the analysis, to get more clear-cut factors and aid interpretability. This can be especially important if the sample size is small (<250) to create a good factor solution.

As mentioned above, a rule of thumb by MacCallum et al. is that if the total sample size is smaller than 250, we should strive to achieve an average post-extraction communality of 0.6. In our example we have 948 valid cases, so we allow for lower average communality and still hope for a good factor solution.

Another indicator that we can use to decide whether to keep or exclude an item are the item loadings. We need to look at the Rotated factor matrix in the case of Orthogonal rotations or the Pattern matrix in the case of Oblique rotations for the item loadings. If there are items that load very poorly on all factors, we may consider dropping that item from factor analysis.

Also, usually our goal is to get well interpretable factors and factors that are well distinguished from each other, so we might even consider dropping items that have a moderate but very similar loading on multiple factors. This could improve the distinguishability of the factors from each other and may make the factors easier to interpret, but at the cost of excluding the variance of that variable from the factor model.

General guidelines when considering excluding items (from https://en.wikiversity.org/wiki/Survey_research_and_design_in_psychology/Lectures/Exploratory_factor_analysis/Notes#Assumption_testing):

1. Communalities (each ideally > .5)
2. Size of main loading (bare min > |.4|, preferably > |.5|, ideally > |.6|)
3. Meaning of item (face validity)
4. Contribution it makes to the factor (i.e., is a better measure of the latent factor achieved by including or not including this item?)
5. Number of items already in the factor (i.e., if there are already many items (e.g., > 6) in the factor, then the researcher can be more selective about which ones to include and which ones to drop)
6. Eliminate 1 variable at a time, then re-run, before deciding which/if any items to eliminate next
7. (Size of cross loadings max ~ |.3|)

4.7 Naming the factors

In the end of the iterative process, we arrive at the final factor solution that we are satisfied with. This can take many tries. When we are done, the final task is to name the factors, that is, to give them a name or label that corresponds to their interpretation.

The names should be informative and should capture the main aspect in which the items loading onto a given factor are similar to each other. For example, the original authors devising the questionnaire found 4 factor, and named them: Affiliative humor, Self-enhancing humor, Aggressive humor, Self-defeating humor.

4.8 Saving factor scores

You can use your factor model to produce factor scores for observations in your original dataset or in a new dataset. The factor score will tell you how high or low value a certain observation in your dataset takes on the factors, based on their responses to the individual items.

You can calculate the factor scores by using the factor.scores() function, by providing the data object name and the factor model you want to use to calculate the factor scores by. The data object has to have a value for all items in the final factor solutions, and only those variables.

In our example, you can compute the factor scores for each observation in our original dataset from the factor object EFA_mod2 by using following code. Note that I am specifying hsq[,1:32] as the dataframe containing the values on the items entered into the factor analysis. In this code we also add the factor scores to our data object as new variables so that we can work with the factor scores later:

```
factorscores = factor.scores(hsq[, 1:32], EFA_mod2)$scores

hsq_with_factorscroes = cbind(hsq, factorscores)
```

Practice

You can practice the techniques you have learned above on the Big Five Inventory (bfi) database. This is a dataset included in the psych package, which contains the responses of 2800 individuals to the Big Five personality questionnaire. The first 25 columns contain the responses to the BFI questions, and the last few columns (gender, education, and age) contain demographic data. You can read the answers to the questions for each item and how responses are coded by running the ?bfi command.

You can load the database with the following commands.

```
data(bfi)
my_data_bfi = bfi[, 1:25]
```

In this exercise, use only the first 25 columns of the original frame. First, start with visualizing the correlation between variables. Use different methods such as ggcorr(), ggcorrplot() combined with hc.order=TRUE, or network_plot(). Then perform an exploratory factor analysis, and based on this, determine how many factors are ideal to keep, which items load highest on which factors, and based on this, how to name the factors. Which items are most and least represented in the factor structure?