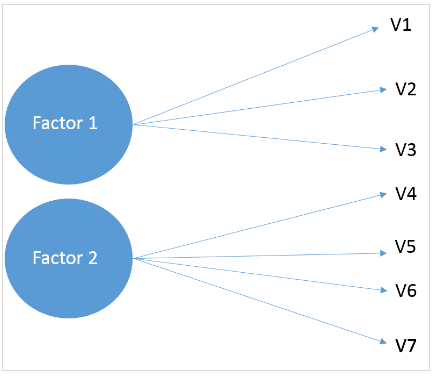
**Factor Analysis using R**

Exploratory Factor Analysis (EFA) or roughly known as factor analysis in R is a statistical technique that is used to identify the latent relational structure among a set of variables and narrow it down to a smaller number of variables. This essentially means that the variance of a large number of variables can be described by a few summary variables, i.e., factors. Here is an overview of exploratory factor analysis in R.



As the name suggests, EFA is exploratory in nature – we don’t really know the latent variables, and the steps are repeated until we arrive at a lower number of factors. In this tutorial, we’ll look at EFA using R. Now, let’s first get the basic idea of the dataset.

**1. The Data**

This dataset contains 90 responses for 14 different variables that customers consider while purchasing a car. The survey questions were framed using a 5-point Likert scale with 1 being very low and 5 being very high. The variables were the following:

* Price
* Safety
* Exterior looks
* Space and comfort
* Technology
* After-sales service
* Resale value
* Fuel type
* Fuel efficiency
* Color
* Maintenance
* Test drive
* Product reviews
* Testimonials

**2. Importing WebData**

Now we’ll read the dataset present in CSV format into R and store it as a variable.

**data <- read.csv(file.choose(),header=TRUE)**

It’ll open a window to choose the CSV file and the `header` option will make sure that the first row of the file is considered as the header. Enter the following to see the first several rows of the data frame and confirm that the data has been stored correctly.

**head(data)**

**3. Package Installation**

Now we’ll install the required packages to carry out further analysis. These packages are `psych` and `GPArotation`. In the code given below, we are calling `install.packages()` for installation.

**install.packages(‘psych’)**

**install.packages(‘GPArotation’)**

**4. Number of Factors**

Next, we’ll find out the number of factors that we’ll be selecting for factor analysis. This is evaluated via methods such as `Parallel Analysis` and `eigenvalue`, etc.

**Parallel Analysis**

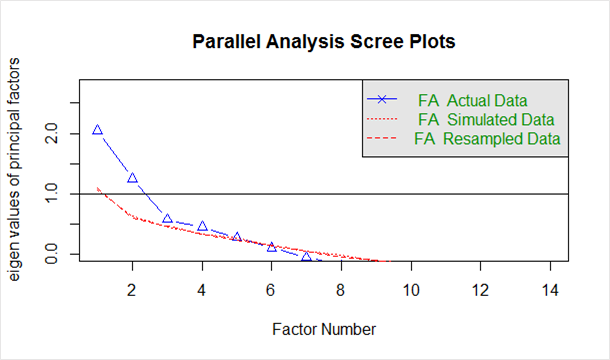
We’ll be using the `Psych` package’s `fa.parallel` function to execute the parallel analysis. Here we specify the data frame and factor method (`minres` in our case). Run the following to find an acceptable number of factors and generate the `scree plot`:

**Parallel <- fa.parallel(data, fm = ‘minres’, fa = ‘fa’)**

The console would show the maximum number of factors we can consider. Here is how it’d look.

*“Parallel analysis suggests that the number of factors = 5 and the number of components = NA“*

Given below in the `scree plot` generated from the above code:



The blue line shows eigenvalues of actual data and the two red lines (placed on top of each other) show simulated and resampled data. Here we look at the large drops in the actual data and spot the point where it levels off to the right. Also, we locate the point of inflection – the point where the gap between simulated data and actual data tends to be minimum.

Looking at this plot and parallel analysis, anywhere between 2 to 5 factors would be a good choice.

Factor Analysis

Now that we’ve arrived at a probable number of factors, let’s start off with 3 as the number of factors. In order to perform factor analysis, we’ll use the psych packages fa()function. Given below are the arguments we’ll supply:

* r – Raw data or correlation or covariance matrix
* nfactors – Number of factors to extract
* rotate – Although there are various types of rotations, `Varimax` and `Oblimin` are the most popular
* fm – One of the factor extraction techniques like `Minimum Residual (OLS)`, `Maximum Liklihood`, `Principal Axis` etc.

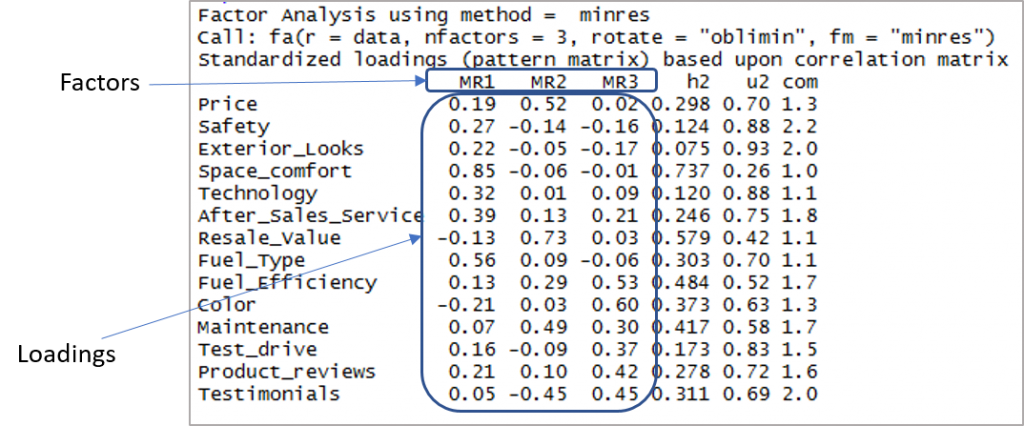
In this case, we will select oblique rotation (rotate = “oblimin”) as we believe that there is a correlation in the factors. Note that Varimax rotation is used under the assumption that the factors are completely uncorrelated. We will use `Ordinary Least Squared/Minres` factoring (fm = “minres”), as it is known to provide results similar to `Maximum Likelihood` without assuming a multivariate normal distribution and derives solutions through iterative eigen decomposition like a principal axis.

Run the following to start the analysis.

**Threefactor <- fa(data,nfactors = 3,rotate = “oblimin”,fm=”minres”)**

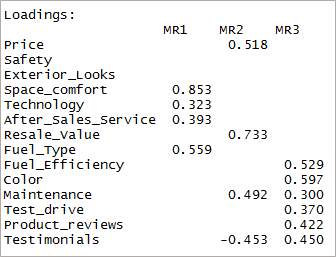
**print(Threefactor)**

Here is the output showing factors and loadings:



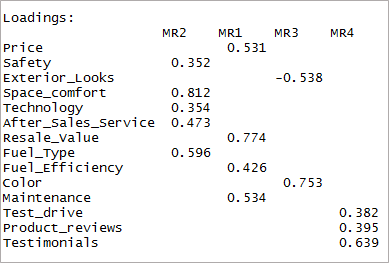
Now we need to consider the loadings of more than 0.3 and not loading on more than one factor. Note that negative values are acceptable here. So let’s first establish the cut-off to improve visibility.

print(Threefactor$loadings,cutoff = 0.3)



As you can see two variables have become insignificant and two others have double-loading. Next, we’ll consider the ‘4’ factors.

**fourfactor <- fa(data,nfactors = 4,rotate = “oblimin”,fm=”minres”) print(fourfactor$loadings,cutoff = 0.3)**



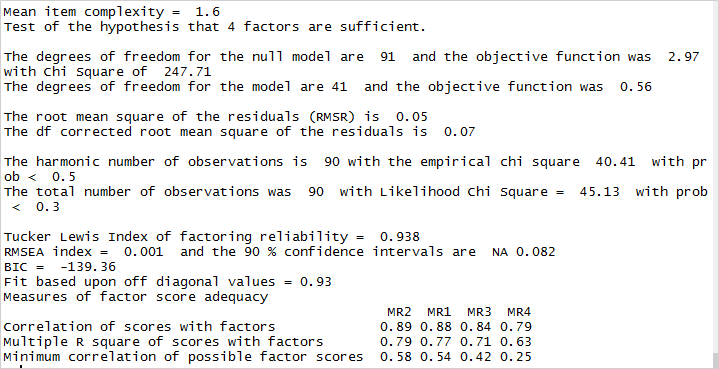
We can see that it results in only single-loading. This is known as the simple structure.

Hit the following to look at the factor mapping.

**fa.diagram(fourfactor)**

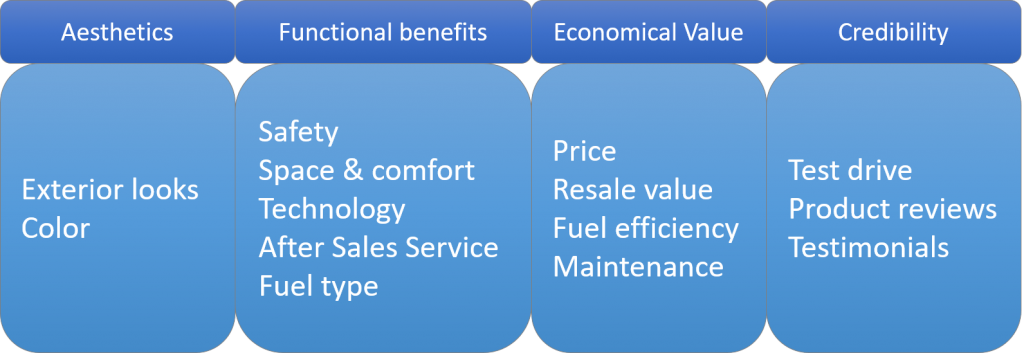
**Adequacy Test**

Now that we’ve achieved a simple structure it’s time for us to validate our model. Let’s look at the factor analysis output to proceed.



The root means the square of residuals (RMSR) is 0.05. This is acceptable as this value should be closer to 0. Next, we should check the RMSEA (root mean square error of approximation) index. Its value, 0.001 shows a good model fit as it is below 0.05. Finally, the Tucker-Lewis Index (TLI) is 0.93 – an acceptable value considering it’s over 0.9.

**Naming the Factors**



After establishing the adequacy of the factors, it’s time for us to name the factors. This is the theoretical side of the analysis where we form the factors depending on the variable loadings. In this case, here is how the factors can be created.