R Workshop: Factor Analysis

Winter 2024

Contents

The content with * is optional.

This notebook shows how to perform Factor Analysis with R and how to use the output to construct a perceptual map. To perform this analysis, you need to install these R packages:

- corrplot
- gplots
- nFactors

1 Brand rating data

Consumers rate brands with regard to perceptual adjectives as expressed on survey items with the following form:

On a scale from 1 to 10 - where 1 is the least and 10 is the most - how is *Brand A* perceived? An observation is one respondent's rating of a brand on all adjectives. For instance:

1. How *trendy* is *Intelligentsia Coffee*?
2. How much of a *category leader* is *Blue Bottle Coffee*?

The data we used stimulated the rating of 10 brands ("a" to "j") on 9 adjectives ("performance", "leader", "latest", "fun", and so on), for $N = 100$ respondents. We start by loading and checking the data:

```
brand.ratings <- read.csv("Data_Factor_Analysis.csv", stringsAsFactors = TRUE)
head(brand.ratings)
```

Each of the 100 respondents has observations on each of the 10 brands, so there are 1,000 rows in total. We check the data quality and structure:

```
summary(brand.ratings)
```

```
##     perform          leader          latest           fun
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 4.000   1st Qu.: 4.000
##  Median : 4.000   Median : 4.000   Median : 7.000   Median : 6.000
##  Mean   : 4.488   Mean   : 4.417   Mean   : 6.195   Mean   : 6.068
##  3rd Qu.: 7.000   3rd Qu.: 6.000   3rd Qu.: 9.000   3rd Qu.: 8.000
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
##
##     serious          bargain          value           trendy
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.00
##  1st Qu.: 2.000   1st Qu.: 2.000   1st Qu.: 2.000   1st Qu.: 3.00
##  Median : 4.000   Median : 4.000   Median : 4.000   Median : 5.00
##  Mean   : 4.323   Mean   : 4.259   Mean   : 4.337   Mean   : 5.22
##  3rd Qu.: 6.000   3rd Qu.: 6.000   3rd Qu.: 6.000   3rd Qu.: 7.00
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.00
##
##      rebuy           brand
##  Min.   : 1.000   a      :100
##  1st Qu.: 1.000   b      :100
##  Median : 3.000   c      :100
##  Mean   : 3.727   d      :100
##  3rd Qu.: 5.000   e      :100
##  Max.   :10.000   f      :100
##                   (Other):400
```

It shows the range of the ratings for each adjective in the range from 1 to 10.

```
str(brand.ratings)
```

```
## 'data.frame':    1000 obs. of  10 variables:
##  $ perform: int  2 1 2 1 1 2 1 2 2 3 ...
##  $ leader : int  4 1 3 6 1 8 1 1 1 1 ...
##  $ latest : int  8 4 5 10 5 9 5 7 8 9 ...
##  $ fun    : int  8 7 9 8 8 5 7 5 10 8 ...
##  $ serious: int  2 1 2 3 1 3 1 2 1 1 ...
##  $ bargain: int  9 1 9 4 9 8 5 8 7 3 ...
##  $ value  : int  7 1 5 5 9 7 1 7 7 3 ...
##  $ trendy : int  4 2 1 2 1 1 1 7 5 4 ...
##  $ rebuy  : int  6 2 6 1 1 2 1 1 1 1 ...
##  $ brand  : Factor w/ 10 levels "a","b","c","d",..: 1 1 1 1 1 1 1 1 1 1 ...
```

It shows that ratings were read as numeric while brand labels were interpreted as a factor. In short, the data appear to be clean and formatted appropriately.

There are 9 perceptual adjectives in this data set.

Perceptual adjective (column name)	Example survey text
perform	*Brand* has a strong performance
leader	*Brand* is a leader in the field
latest	*Brand* has the latest products
fun	*Brand* is fun
serious	*Brand* is serious
bargain	*Brand* products are a bargain
value	*Brand* has a good value
trendy	*Brand* is trendy
rebuy	I would buy from *Brand* again

1.1 Rescaling the data

It is often a good practice to rescale raw data. This makes data more comparable across individuals and samples. A common procedure is to * centre* each variable by subtracting its mean from every observation and then *rescale* those centered values as units of standard deviation. This is commonly called *standardizing, normalizing, or Z-scoring* the data. We never want to alter raw data, so we create a new copy of the data and rescale it.

```
brand.sc <- brand.ratings
brand.sc[,1:9] <- scale (brand.ratings[,1:9])
#we select all rows and the first 9 columns as the 10th column is a factor variable.
summary(brand.sc)
```

```
##     perform            leader            latest             fun          
##  Min.   :-1.0888   Min.   :-1.3100   Min.   :-1.6878   Min.   :-1.84677  
##  1st Qu.:-1.0888   1st Qu.:-0.9266   1st Qu.:-0.7131   1st Qu.:-0.75358  
##  Median :-0.1523   Median :-0.1599   Median : 0.2615   Median :-0.02478  
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000  
##  3rd Qu.: 0.7842   3rd Qu.: 0.6069   3rd Qu.: 0.9113   3rd Qu.: 0.70402  
##  Max.   : 1.7206   Max.   : 2.1404   Max.   : 1.2362   Max.   : 1.43281  
##                                                                          
##     serious           bargain            value             trendy        
##  Min.   :-1.1961   Min.   :-1.22196   Min.   :-1.3912   Min.   :-1.53897  
##  1st Qu.:-0.8362   1st Qu.:-0.84701   1st Qu.:-0.9743   1st Qu.:-0.80960
```

```
##  Median :-0.1163   Median :-0.09711   Median :-0.1405   Median :-0.08023
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000
##  3rd Qu.: 0.6036   3rd Qu.: 0.65279   3rd Qu.: 0.6933   3rd Qu.: 0.64914
##  Max.   : 2.0434   Max.   : 2.15258   Max.   : 2.3610   Max.   : 1.74319
##
##      rebuy             brand
##  Min.   :-1.0717   a      :100
##  1st Qu.:-1.0717   b      :100
##  Median :-0.2857   c      :100
##  Mean   : 0.0000   d      :100
##  3rd Qu.: 0.5003   e      :100
##  Max.   : 2.4652   f      :100
##                    (Other):400
```

We can see the standardized variable has a mean of 0.00.

1.2 Correlation

Correlations among variables show some variables are highly related to each other and potentially could be combined.

```
cor(brand.sc[,1:9])
```

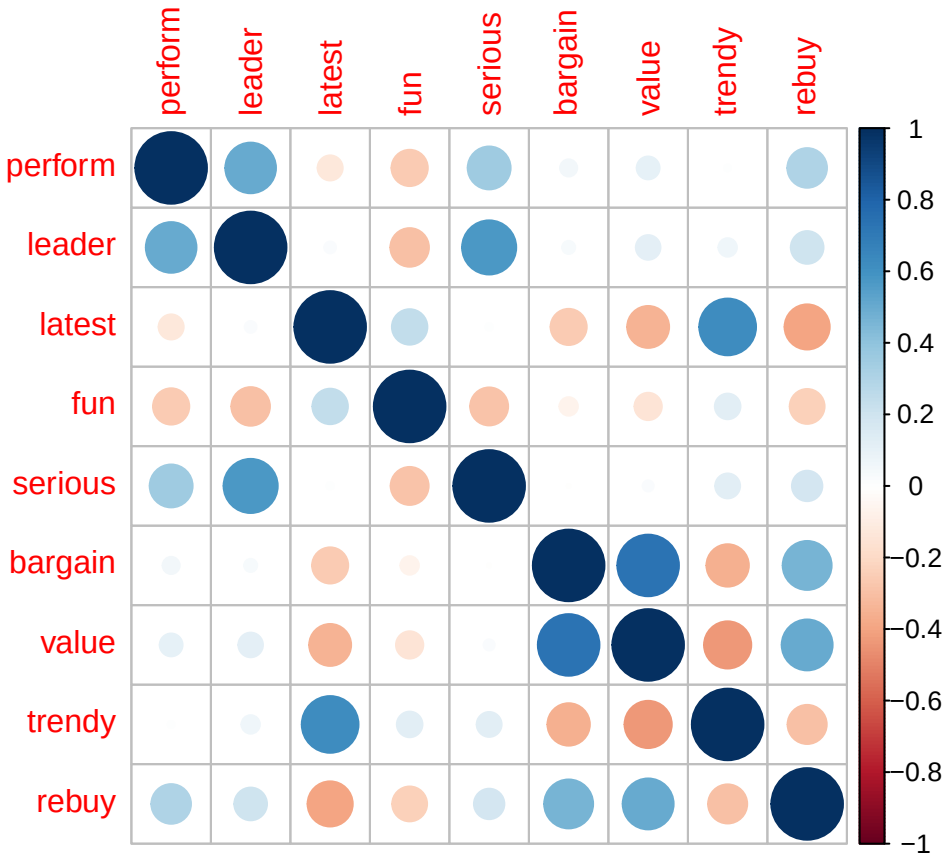
```
##              perform      leader       latest        fun      serious
## perform  1.000000000  0.50020206 -0.122445813 -0.2563323  0.359172206
## leader   0.500202058  1.00000000  0.026890447 -0.2903576  0.571215126
## latest  -0.122445813  0.02689045  1.000000000  0.2451545  0.009951527
## fun     -0.256332316 -0.29035764  0.245154457  1.0000000 -0.281097443
## serious  0.359172206  0.57121513  0.009951527 -0.2810974  1.000000000
## bargain  0.057129372  0.03309405 -0.254419016 -0.0665528 -0.002655590
## value    0.101946104  0.11831017 -0.342713717 -0.1452185  0.023756556
## trendy   0.008733494  0.06651244  0.627627667  0.1279736  0.121009377
## rebuy    0.306658801  0.20870036 -0.397180225 -0.2371607  0.180702720
##              bargain       value       trendy      rebuy
## perform  0.05712937  0.10194610  0.008733494  0.3066588
## leader   0.03309405  0.11831017  0.066512436  0.2087004
## latest  -0.25441902 -0.34271372  0.627627667 -0.3971802
## fun     -0.06655280 -0.14521849  0.127973639 -0.2371607
## serious -0.00265559  0.02375656  0.121009377  0.1807027
## bargain  1.00000000  0.73962672 -0.350533746  0.4673811
## value    0.73962672  1.00000000 -0.434534536  0.5059617
## trendy  -0.35053375 -0.43453454  1.000000000 -0.2982462
## rebuy    0.46738109  0.50596166 -0.298246195  1.0000000
```

We can also use *corrplot()* for visualizing the correlations among the variables:

```
library(corrplot)
```

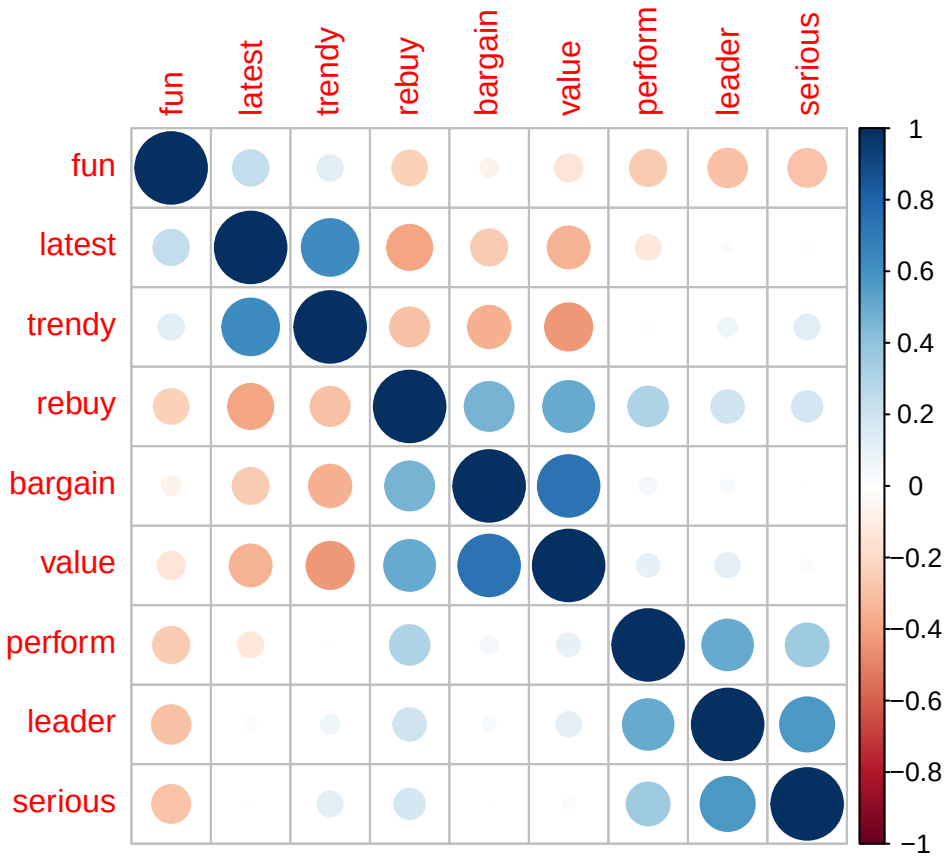
```
## corrplot 0.92 loaded
```

```
corrplot(cor(brand.sc[,1:9]))
```



We can use the optional argument (order = "hclust") to order variables for better visualization. Try to use "?corrplot" in your console for more information about the function and the available arguments.

```
corrplot(cor(brand.sc[,1:9]), order = "hclust")
```



The result shows that some of the ratings are highly related to each other so that they could be represented by common factors.

1.3 Mean rating by brand

1.3.1 aggregate()

The simplest business question regarding this data is perhaps: "What is the average (mean) position of the brand on each adjective?" We can use *aggregate()* to find the mean of each variable by brand and store the result in a new data frame:

```
brand.mean <- aggregate(. ~ brand, data=brand.sc, mean)
brand.mean
```

A *heatmap* is a useful way to visualize *brand.mean* matrix because it colors data points by the intensities of their values.

Before proceeding, we clean the new *brand.mean* object a bit. We want to use the brand variable to name the rows and remove the brand variable from the data.

```
rownames(brand.mean) <- brand.mean[, 1]
# Use brand for the row name
brand.mean <- brand.mean [, -1]
# Remove the brand name column by not selecting the first column
# Negative index is used to exclude the variable
```

The resulting matrix is now nicely formatted with brands by row and adjective means in columns:

```
brand.mean
```

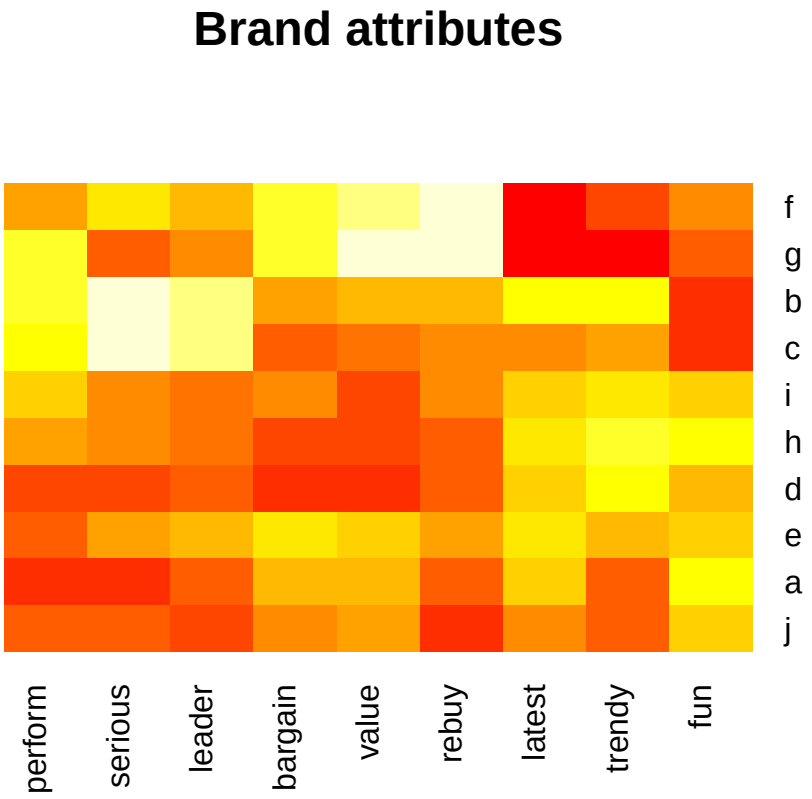
Then we can visualize this matrix by using a heat map :

```
library(gplots)
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
heatmap.2(as.matrix(brand.mean),main = "Brand attributes",
          trace = "none", key = FALSE, dend = "none"
          #turn off some options
          )
```



A lighter color indicates a high value and a darker color indicates a low value. We can see that the brands are clearly perceived differently, with some brands rated high on performance and leadership (brands b and c) and others rated high for value and rebuy intention (brands f and g).

From the correlation plot and heatmap plot, we could guess the groupings and relationships of adjectives and brands. For instance, there is a similarity in the color pattern across columns for the *bargain/value/rebuy*. We will formalize such insight next.

2 Principal component analysis (PCA) using *princomp()*

The *princomp()*function produces an unrotated principal component analysis.

You can use either the original individual ratings data *brand.sc* or the aggregated mean rating data *brand.mean*. The plot of individual respondents' rating will be too dense and it may not clearly tell us about the brand positions! A better solution is to perform PCA using *aggregated* ratings by the brand.

2.1 PCA

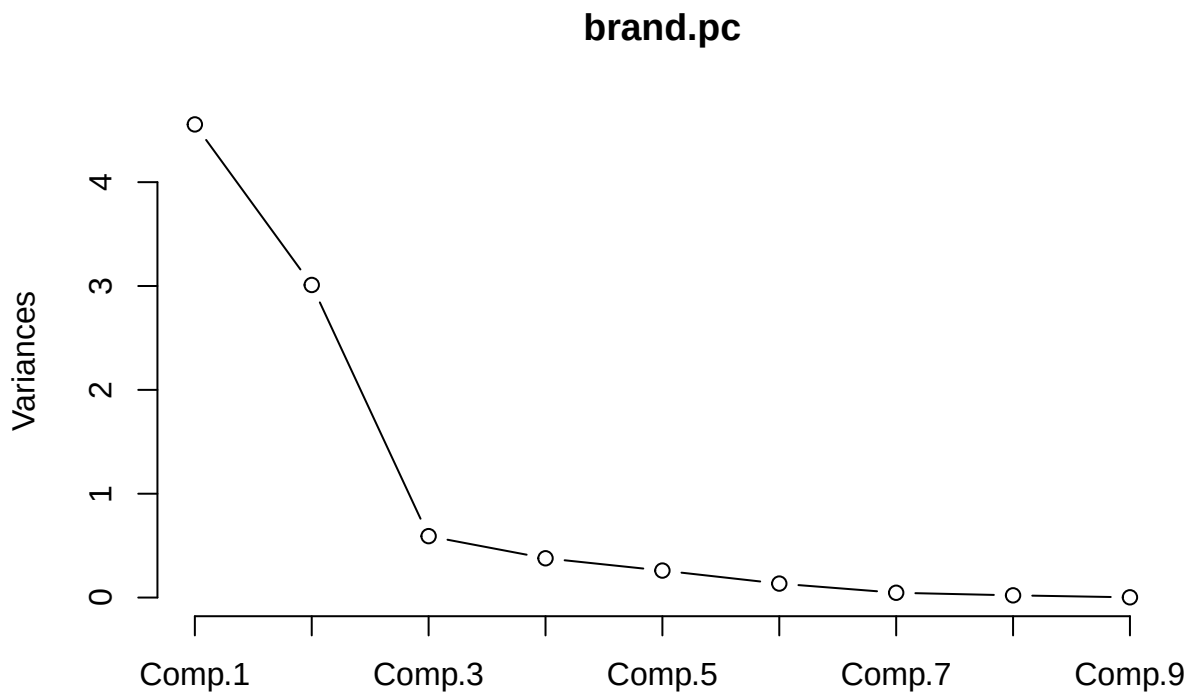
```
brand.pc<- princomp(brand.mean, cor = TRUE)
#We added "cor =TRUE" to use correlation-based one.
summary(brand.pc)
```

```
## Importance of components:
##                           Comp.1    Comp.2     Comp.3     Comp.4    Comp.5
## Standard deviation     2.134521 1.7349473 0.76898915 0.61498280 0.5098261
## Proportion of Variance 0.506242 0.3344491 0.06570492 0.04202265 0.0288803
## Cumulative Proportion  0.506242 0.8406911 0.90639603 0.94841868 0.9772990
##                             Comp.6      Comp.7      Comp.8       Comp.9
## Standard deviation     0.36661576 0.215062433 0.145882355 0.0486674686
## Proportion of Variance 0.01493412 0.005139094 0.002364629 0.0002631692
## Cumulative Proportion  0.99223311 0.997372202 0.999736831 1.0000000000
```

Note: If you are using the original *brand.sc* data, do not forget it still contains brand variables, and you need to select the rating columns 1-9 only, by using *princom(brand.sc[,1:9])*.

We use *plot()* for a *scree plot*, which is a plot of the eigenvalues of factors or principal components.

```
plot(brand.pc,type="l") # scree plot
```



A scree plot often helps us to determine where additional components are not worth; this occurs where the

line has an *elbow*. In the above figure, the elbow occurs at component three. This suggests that the first two components explain most of the variation in the observed brand rating.

```
loadings(brand.pc) # pc loadings
```

```
##
## Loadings:
##         Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
## perform  0.285  0.337  0.481  0.470  0.396  0.435
## leader   0.247  0.457 -0.317 -0.191         0.119 -0.610         0.451
## latest  -0.356  0.251 -0.496  0.275  0.461        -0.196 -0.119 -0.466
## fun     -0.336 -0.335 -0.152  0.324 -0.388  0.636 -0.246  0.179
## serious  0.212  0.475 -0.244 -0.212 -0.394  0.334  0.439        -0.407
## bargain  0.361 -0.278 -0.459  0.291  0.112  0.127  0.319 -0.513  0.321
## value    0.401 -0.241 -0.336         0.206                0.778
## trendy  -0.311  0.375         0.484 -0.273 -0.339  0.322  0.243  0.410
## rebuy    0.430                 0.442 -0.438 -0.368 -0.352 -0.142 -0.372
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.111  0.111  0.111  0.111  0.111  0.111  0.111  0.111  0.111
## Cumulative Var  0.111  0.222  0.333  0.444  0.556  0.667  0.778  0.889  1.000
```

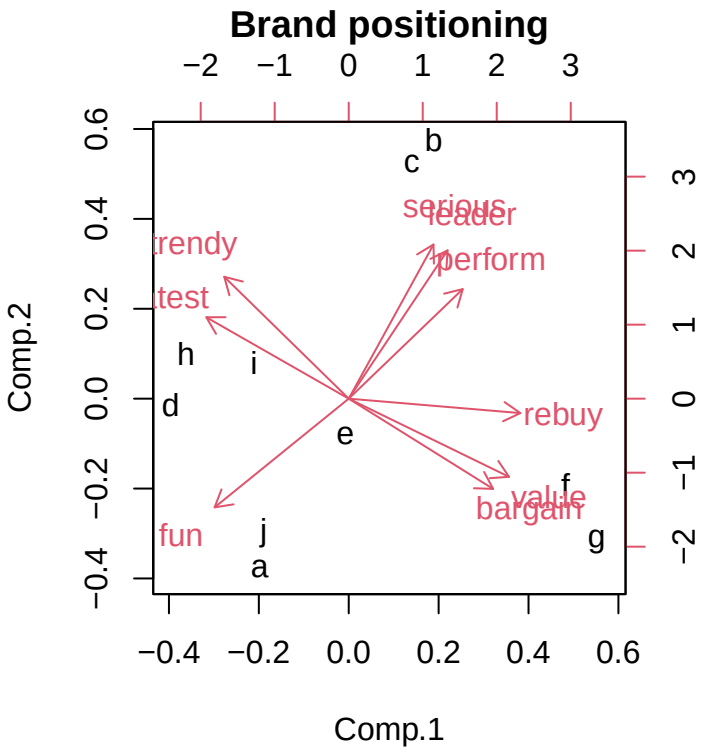
```
brand.pc$scores # the principal components
```

```
##        Comp.1      Comp.2     Comp.3     Comp.4      Comp.5      Comp.6
## a -1.33844390 -2.07516736 -0.8122064 -0.2718479  0.48312804  0.13000857
## b  1.27882437  3.16566948 -0.6728669  0.4260610  0.59604422 -0.27621388
## c  0.94758158  2.87102460  0.4373410 -1.1352295 -0.04911469  0.26686063
## d -2.67305251 -0.07560147  0.6699120 -0.1392928 -0.29772003 -0.90897740
## e -0.05065207 -0.45435975 -1.5626455  0.2818699 -0.13169989 -0.06627978
## f  3.25569134 -1.04004170 -0.2793130 -0.1952913 -1.10420368 -0.01568321
## g  3.72365199 -1.74788465  1.0669089  0.5065070  0.57881864 -0.09100917
## h -2.44257414  0.54678181  0.3765861  0.5529998 -0.39200093  0.41844244
## i -1.42198785  0.44723594  0.5608676  0.8640900 -0.10997727  0.36645365
## j -1.27903882 -1.63765690  0.2154161 -0.8898662  0.42672558  0.17639814
##        Comp.7       Comp.8       Comp.9
## a -0.31644487 -0.041531674 -0.078403652
## b  0.20827319  0.058302325 -0.049489688
## c -0.24761926 -0.054420917  0.027937805
## d -0.06870184 -0.023255194  0.003026500
## e -0.05612634 -0.018226700  0.102182048
## f  0.12754579 -0.009395687 -0.049461109
## g -0.13166370  0.063897903  0.031587970
## h -0.05665576  0.311997827 -0.003602992
## i  0.08841666 -0.318403585 -0.003490899
## j  0.45297613  0.031035701  0.019714016
```

2.2 Visualising PCA (perceptual map)

biplot() is a two-dimensional plot of data points with respect to the first two PCA components. It reveals how the rating adjectives are associated. A biplot of the PCA solution for the mean rating gives an interpretable *perceptual map*, showing where the brands are placed with respect to the first two principal components.

```
biplot(brand.pc, main = "Brand positioning")
```



The result shows the adjective map in four regions:

- category leadership: “serious”, “leader”, and “perform”
- category value: “rebuy”, “value”, and “bargain”
- category trendiness: “trendy” and “latest”
- category fun: “fun”

What does the map tell us? First, we interpret the adjective clusters and relationships and see four areas with well-differentiated sets of adjectives and brands that are positioned in proximity. Brands f and g are high on “value”, while brands a and j are relatively high on *fun*, which is opposite in direction from leadership adjectives (“leader” and “serious”).

With such a map, one might form questions and then refer to the underlying data to answer them.

Suppose that you are the brand manager for brand *e*, what does the map tell you?

Your brand is in the center and thus appears not to be well-differentiated on any of the dimensions. It could be good or bad, depending on your strategic goal.

What should you do about the position of your brand *e*?

Again, it depends on the strategic goal. If you wish to increase differentiation, one possibility would be to take action to shift your brand in some direction on the map.

Suppose you wanted to move in the direction of brand c, you could look at the specific differences from c in the data:

```
brand.mean["c",] - brand.mean["e",]
```

This shows you that e is relatively stronger than c on “value” and “fun”, and weaker than c on “perform” and “serious”. Those could be the aspects of the product or message for e to work on.

Another option could be to aim for a differentiated space where no brand is positioned. In the figure, there is a large gap between the lower right and the upper right. This area could be described as the "value leader" area or similar. How do we find out how to position there? Let us assume the gap reflects approximately the average of those four brands (*g,f, c, and b*). We can find the average using *colMeans()* on the brands' rows, and then take the difference of *e* from that average:

```
colMeans(brand.mean[c("b","c","f","g"),]) - brand.mean["e",]
```

This suggests that brand *e* could target the gap by increasing its emphasis on performance while reducing the emphasis on "latest" and "fun".

3 Factor analysis using *factanal()* *

The *factanal()* function produces maximum likelihood factor analysis.

3.1 Determine the factor number

The first thing is to determine the number of factors to estimate. There are various ways to do this, and two traditional methods are to use a scree plot, and to retain factors where the eigenvalue (a metric for proportion of variance explained) is greater than 1.0. An eigenvalue of 1.0 corresponds to the amount of variance that might be attribute to a single independent variable; a factor that captures less variance than such an item may be considered relatively uninteresting.

The scree plot and the eigenvalues could be formally obtained from the analysis with *nScree()* from the *nFactors*.

```
library(nFactors)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'nFactors'
```

```
## The following object is masked from 'package:lattice':
##
##     parallel
```

```
nScree(brand.mean)
```

```
##   noc naf nparallel nkaiser
## 1   2   2         2       2
```

```
eigen(cor(brand.mean))
```

```
## eigen() decomposition
## $values
## [1] 4.556177847 3.010042148 0.591344314 0.378203844 0.259922697 0.134407115
## [7] 0.046251850 0.021281662 0.002368523
##
## $vectors
##             [,1]        [,2]        [,3]        [,4]        [,5]        [,6]
##  [1,] -0.2852486 -0.33729698  0.48121446  0.46995620  0.39623804  0.43471514
##  [2,] -0.2473668 -0.45654557 -0.31711577 -0.19084693 -0.06130157  0.11868317
##  [3,]  0.3562989 -0.25056983 -0.49589600  0.27477470  0.46061874 -0.08173299
##  [4,]  0.3355152  0.33455495 -0.15167546  0.32394053 -0.38757837  0.63609709
##  [5,] -0.2121240 -0.47463096 -0.24371327 -0.21229430 -0.39428137  0.33437227
##  [6,] -0.3613409  0.27776101 -0.45940272  0.29120398  0.11248446  0.12716342
##  [7,] -0.4010778  0.24062869 -0.33576144  0.05052374  0.20581208 -0.08329187
```

```
##  [8,]  0.3114405 -0.37521575 -0.08724910  0.48392969 -0.27261916 -0.33925412
##  [9,] -0.4295359  0.04438337  0.09031492  0.44234693 -0.43824713 -0.36828116
##              [,7]         [,8]         [,9]
##  [1,] -0.02784431  0.074243080  0.012984626
##  [2,]  0.60997229  0.021119910  0.450594077
##  [3,]  0.19587019 -0.119316063 -0.466262266
##  [4,]  0.24602385  0.179248006 -0.008094488
##  [5,] -0.43881277  0.005157446 -0.406716076
##  [6,] -0.31905166 -0.512721569  0.320827507
##  [7,] -0.08325891  0.778125659 -0.065102236
##  [8,] -0.32150758  0.243224760  0.410460300
##  [9,]  0.35159046 -0.141872872 -0.371841553
```

nScree() applies several methods to estimate the number of factors from scree tests.

The final choice of a model depends on whether it is useful. A best practice is to check a few factor solutions, including the ones suggested by the scree and eigenvalues results.

```
brand.fa <- factanal(brand.mean, factors = 2, rotation = "varimax", scores = "regression")
```

A factor analysis can be rotated to have new loading that accounts for the same proportion of variance. The default in *factanal()* is to find factors that have zero correlation among factors (using a *varimax* rotation).

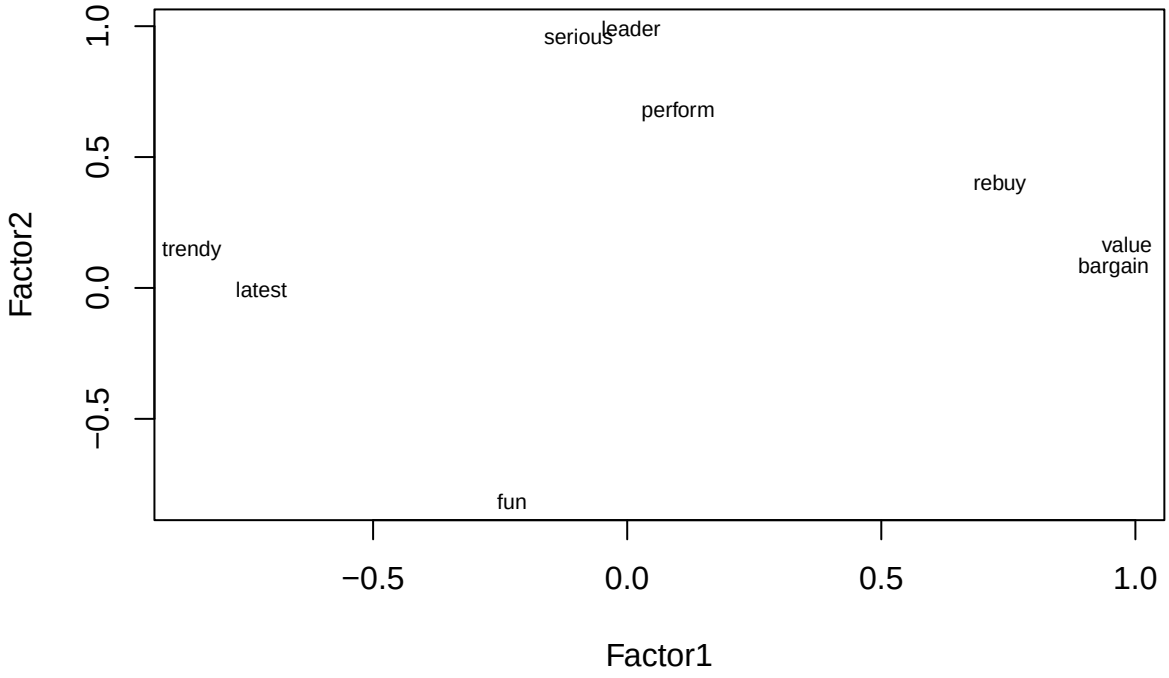
3.2 Factor loadings

Factor loading is obtained from the *loading* value.

```
brand.fl<- brand.fa$loadings[, 1:2]
```

You can simply use the factor loading to plot the variables on the two factors. For instance:

```
plot(brand.fl,type="n") # set up plot
text(brand.fl,labels=names(brand.mean),cex=.7)
```



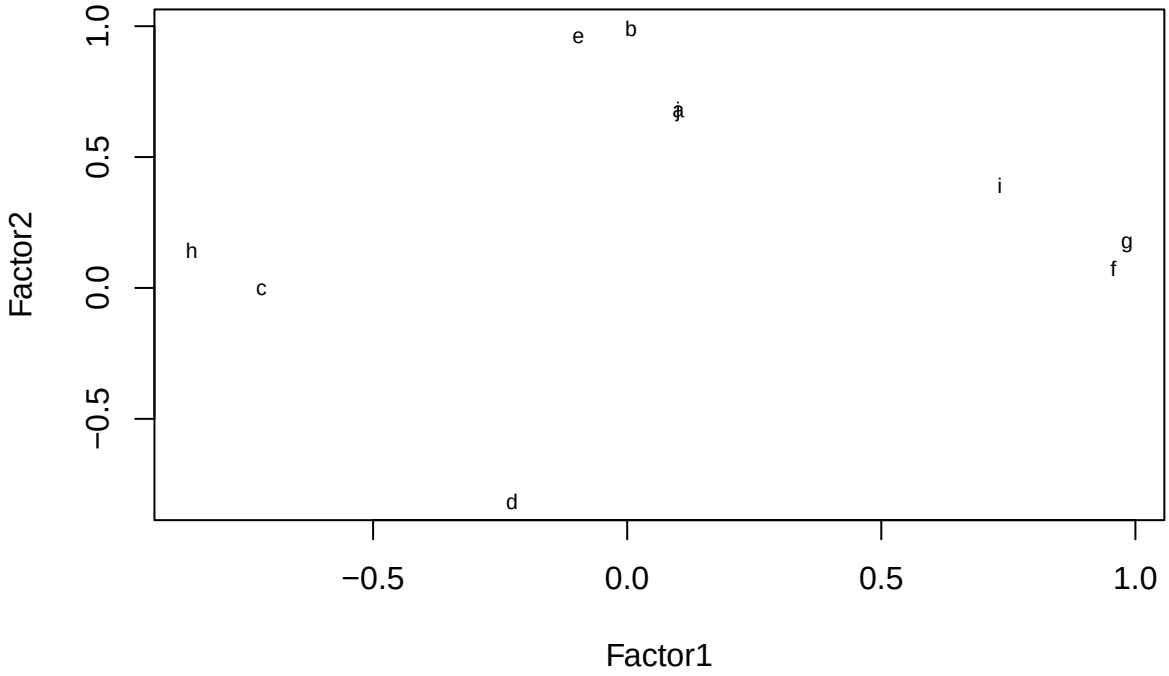
3.3 Factor scores

Factor scores are requested from *factanal()* by adding the *scores = . . . " argument. We can then extract them from the resulting object to form a new data frame.

```
brand.fs <- brand.fa$scores
```

We can simply use the factor score to plot the brands on the two factors. For instance:

```
plot(brand.fl,type="n") # set up plot
text(brand.fl,labels=rownames(brand.mean),cex=.7)
```



4 Recap

To summarize, when you wish to compare several brands across many dimensions, it can be helpful to focus on just the first two or three principal components that explain variation in the data. You can select how many components to focus on using the scree plot, which shows how much variation in the data is explained by each principal component. A perceptual map plots the brands on the first two principal components, revealing how the observations relate to the underlying dimensions (the components).

PCA may be performed using brand survey ratings (as we have done here) or with objective data such as price and physical measurements, or with a combination of the two. In any case, when you are confronted with multidimensional data on brands or products, PCA visualisation is a useful tool for understanding differences in the market.