Dimensionsreduktion mit PCA und EFA

Oliver Gansser

Dimensionsreduktion

Datensätze in den Sozialwissenschaften haben oft viele Variablen - oder auch Dimensionen - und es ist vorteilhaft, diese auf eine kleinere Anzahl von Variablen (oder Dimensionen) zu reduzieren. Zusammenhänge zwischen Konstrukten können so klarer identifiziert werden.

In diese Übung betrachten wir zwei gängige Methoden, um die Komplexität von multivarianten, metrischen Daten zu reduzieren, indem wir die Anzahl der Dimensionen in den Daten reduzieren.

- Die Hauptkomponentenanalyse (PCA) versucht, unkorrelierte Linearkombinationen zu finden, die die maximale Varianz in den Daten erfassen. Die Blickrichtung ist von den Daten zu den Komponenten.
- Die Exploratorische Faktorenanalyse (EFA) versucht die Varianz auf Basis einer kleinen Anzahl von Dimensionen zu modellieren, während sie gleichzeitig versucht, die Dimensionen in Bezug auf die ursprünglichen Variablen interpretierbar zu machen. Es wird davon ausgegangen, dass die Daten einem Faktoren Modell entsprechen. Die Blickrichtung ist von den Faktoren zu den Daten.

Gründe für die Notwendigkeit der Datenreduktion

- Im technischen Sinne der Dimensionsreduktion können wir statt Variablen-Sets die Faktorwerte verwenden.
- Wir können Unsicherheit verringern. Wenn wir glauben, dass ein Konstrukt nicht eindeutig messbar ist, dann kann mit einem Variablen-Set die unsicherheit reduziert werden.
- Wir können den Aufwand bei der Datenerfassung vereinfachen, indem wir uns auf Variablen konzentrieren, von denen bekannt ist, dass sie einen hohen Beitrag zum interessierenden Faktor leisten. Wenn wir feststellen, dass einige Variablen für einen Faktor nicht wichtig sind, können wir sie aus dem Datensatz eliminieren.

Benötigte Pakete

Pakete, die für die Datenanalyse benötigt werden, müssen vorher einmalig in R installiert werden.

```
# install.packages("corrplot")
# install.packages("gplots")
# install.packages("nFactors")
# install.packages("scatterplot3d")
```

Daten

Wir untersuchen die Dimensionalität mittels eines simulierten Datensatzes der typisch für die Wahrnehmung von Umfragen ist. Die Daten spiegeln Verbraucherbewertungen von Marken in Bezug auf Adjektive wieder, die in Umfragen in folgender Form abgefragt werden:

Auf einer Skala von 1 bis 10 (wobei 1 am wenigsten und 10 am meisten zutrifft)

wie... [ADJECTIV]... ist ... [Marke A]...?

Wir verwenden einen *simulierten* Datensatz aus *Chapman & Feit (2015): R for Marketing Research and Analytics* (http://r-marketing.r-forge.r-project.org). Die Daten umfassen simulierte Bewertungen von 10 Marken ("a" bis "j") mit 9 Adjektiven ("performance", "leader", "latest", "fun" usw.) für n = 100 simulierte Befragte.

Das Einlesen der Daten erfolgt direkt über das Internet.

```
brand.ratings <- read.csv("http://goo.gl/IQl8nc")
```

Wir überprüfen zuerst die Struktur des Datensatzes, die ersten 6 Zeilen und die Zusammenfassung

```
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 ...
```

```
head(brand.ratings)
```

```
##   perform leader latest fun serious bargain value trendy rebuy brand
## 1       2      4      8   8       2       9     7      4     6     a
## 2       1      1      4   7       1       1     1      2     2     a
## 3       2      3      5   9       2       9     5      1     6     a
## 4       1      6     10   8       3       4     5      2     1     a
## 5       1      1      5   8       1       9     9      1     1     a
## 6       2      8      9   5       3       8     7      1     2     a
```

```
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
```

Jeder der 100 simulierten Befragten beurteilt 10 Marken, das ergibt insgesamt 1000 Beobachtungen (Zeilen) im Datensatz.

Wir sehen in der `summary ()`, dass die Bereiche der Bewertungen für jedes Adjektiv 1-10 sind. In `str()` sehen wir, dass die Bewertungen als numerisch eingelesen wurden, während die Markennamen als Faktoren eingelesen wurden. Die Daten sind somit richtig formatiert.

Neuskalierung der Daten

In vielen Fällen ist es sinnvoll, Rohdaten neu zu skalieren. Dies wird üblicherweise als **Standardisierung**, **Normierung**, oder **Z Scoring/Transformation** bezeichnet. Als Ergebnis ist der Mittelwert aller Variablen über alle Beobachtungen dann 0. Da wir hier gleiche Skalenstufen haben, ist ein Skalieren nicht unbedingt notwendig, wir führen es aber trotzdem durch.

Ein einfacher Weg, alle Variablen im Datensatz auf einmal zu skalieren ist der Befehl `scale()`. Da wir die Rohdaten nie ändern wollen, weisen wir die Rohwerte zuerst einem neuen Dataframe `brand.sc` zu und scalieren anschließend die Daten. Wir skalieren in unserem Datensatz nur die ersten 9 Variablen, weil die 10. Variable der Faktor für die Markenamen ist.

```
brand.sc <- brand.ratings
brand.sc[, 1:9] <- scale(brand.ratings[, 1:9])
summary(brand.sc)
```

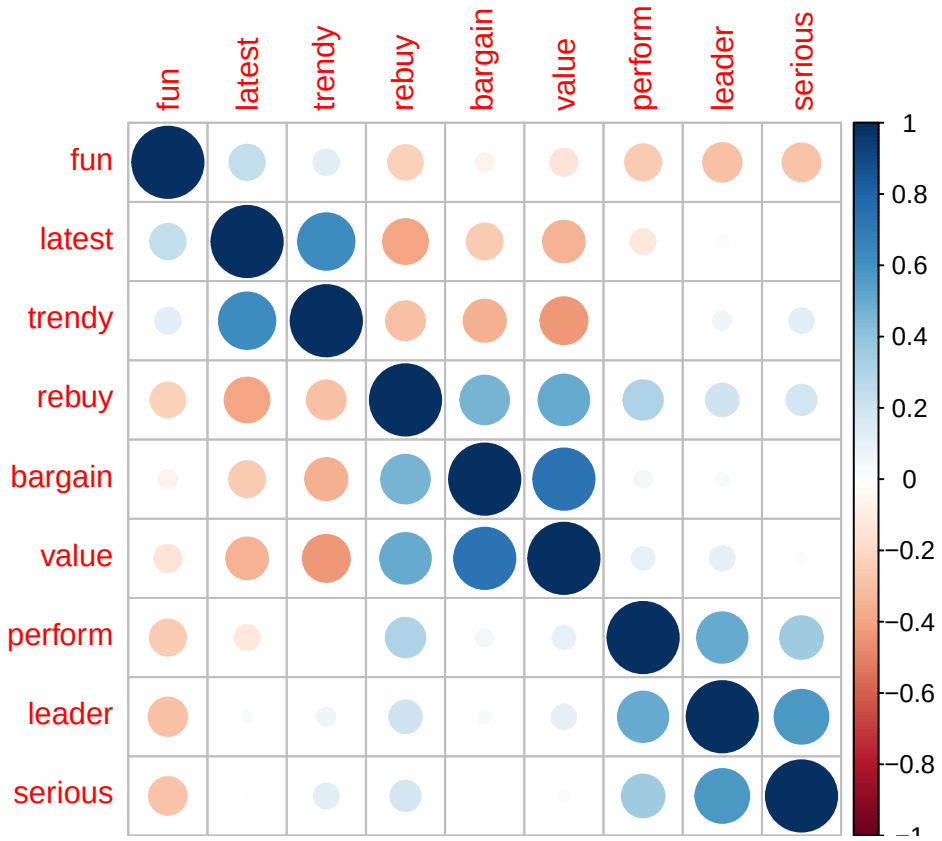
```
##     perform            leader            latest             fun          
##  Min.   :-1.0888   Min.   :-1.3100   Min.   :-1.6878   Min.   :-1.84677  
##  1st Qu.:-1.0888   1st Qu.:-0.9266   1st Qu.:-0.7131   1st Qu.:-0.75358  
##  Median :-0.1523   Median :-0.1599   Median : 0.2615   Median :-0.02478  
##  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
```

Die Daten wurden richtig skaliert, da der Mittelwert aller Variablen über alle Beobachtungen 0 ist.

Zusammenhänge in den Daten

Wir verwenden den Befehl `corrplot()` für die Erstinspektion von bivariaten Beziehungen zwischen den Variablen. Das Argument `order = "hclust"` ordnet die Zeilen und Spalten entsprechend der Ähnlichkeit der Variablen in einer hierarchischen Cluster-Lösung der Variablen (mehr dazu im Teil *Clusteranalyse*) neu an.

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



Die Visualisierung der Korrelation der Adjektive scheint drei allgemeine Cluster zu zeigen:

- fun/latest/trendy
- rebuy/bargain/value
- perform/leader/serious

Daten mit fehlende Werten

Wenn in den Daten leere Zellen, also fehlende Werte, vorhanden sind, dann kann es bei bestimmten Rechenoperationen zu Fehlermeldungen kommen. Dies betrifft zum Beispiel Korrelationen, PCA und EFA. Der Ansatz besteht besteht deshalb darin, NA-Werte explizit zu entfernen. Dies kann am einfachsten mit dem Befehl `na.omit()` geschehen:

Beispiel:

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

Da wir in unserem Datensatz simulierte Daten verwenden, gibt es auch keine Leerzellen.

Hinweis: In vielen Funktionen gibt es auch die Option `na.rm = TRUE`, die fehlende Werte entfernt, z. B.:

```
var(brand.sc[, 1:9], na.rm = TRUE)
```

Aggregation der durchschnittlichen Bewertungen nach Marke

Um die Frage "Was ist die durchschnittliche (mittlere) Bewertung der Marke auf jedem Adjektiv?" zu beantworten, können wir den Befehl `aggregate()` verwenden. Dieser berechnet den Mittelwert jeder Variable nach Marke.

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

```
##    brand     perform     leader     latest        fun     serious
## 1      a -0.88591874 -0.5279035  0.4109732  0.6566458 -0.91894067
## 2      b  0.93087022  1.0707584  0.7261069 -0.9722147  1.18314061
## 3      c  0.64992347  1.1627677 -0.1023372 -0.8446753  1.22273461
## 4      d -0.67989112 -0.5930767  0.3524948  0.1865719 -0.69217505
## 5      e -0.56439079  0.1928362  0.4564564  0.2958914  0.04211361
## 6      f -0.05868665  0.2695106 -1.2621589 -0.2179102  0.58923066
## 7      g  0.91838369 -0.1675336 -1.2849005 -0.5167168 -0.53379906
## 8      h -0.01498383 -0.2978802  0.5019396  0.7149495 -0.14145855
## 9      i  0.33463879 -0.3208825  0.3557436  0.4124989 -0.14865746
## 10     j -0.62994504 -0.7885965 -0.1543180  0.2849595 -0.60218870
##        bargain       value      trendy       rebuy
## 1   0.21409609  0.18469264 -0.52514473 -0.59616642
## 2   0.04161938  0.15133957  0.74030819  0.23697320
## 3  -0.60704302 -0.44067747  0.02552787 -0.13243776
## 4  -0.88075605 -0.93263529  0.73666135 -0.49398892
## 5   0.55155051  0.41816415  0.13857986  0.03654811
## 6   0.87400696  1.02268859 -0.81324496  1.35699580
## 7   0.89650392  1.25616009 -1.27639344  1.36092571
## 8  -0.73827529 -0.78254646  0.86430070 -0.60402622
## 9  -0.25459062 -0.80339213  0.59078782 -0.20317603
## 10 -0.09711188 -0.07379367 -0.48138267 -0.96164748
```

Zusätzlich setzten wir die Markennamen als Fallbezeichnung in der Datenmatrix ein.

```
rownames(brand.mean) <- brand.mean[, 1] # Markenname als Fallbezeichnung setzen
brand.mean <- brand.mean[, -1]          # Variablenname brand entfernen
brand.mean
```

```
##       perform     leader     latest        fun     serious     bargain
## a -0.88591874 -0.5279035  0.4109732  0.6566458 -0.91894067  0.21409609
## b  0.93087022  1.0707584  0.7261069 -0.9722147  1.18314061  0.04161938
## c  0.64992347  1.1627677 -0.1023372 -0.8446753  1.22273461 -0.60704302
## d -0.67989112 -0.5930767  0.3524948  0.1865719 -0.69217505 -0.88075605
## e -0.56439079  0.1928362  0.4564564  0.2958914  0.04211361  0.55155051
## f -0.05868665  0.2695106 -1.2621589 -0.2179102  0.58923066  0.87400696
## g  0.91838369 -0.1675336 -1.2849005 -0.5167168 -0.53379906  0.89650392
## h -0.01498383 -0.2978802  0.5019396  0.7149495 -0.14145855 -0.73827529
## i  0.33463879 -0.3208825  0.3557436  0.4124989 -0.14865746 -0.25459062
## j -0.62994504 -0.7885965 -0.1543180  0.2849595 -0.60218870 -0.09711188
##         value      trendy       rebuy
## a  0.18469264 -0.52514473 -0.59616642
## b  0.15133957  0.74030819  0.23697320
## c -0.44067747  0.02552787 -0.13243776
## d -0.93263529  0.73666135 -0.49398892
## e  0.41816415  0.13857986  0.03654811
## f  1.02268859 -0.81324496  1.35699580
## g  1.25616009 -1.27639344  1.36092571
## h -0.78254646  0.86430070 -0.60402622
## i -0.80339213  0.59078782 -0.20317603
## j -0.07379367 -0.48138267 -0.96164748
```

Visualisierung der aggregierten Markenbewertungen

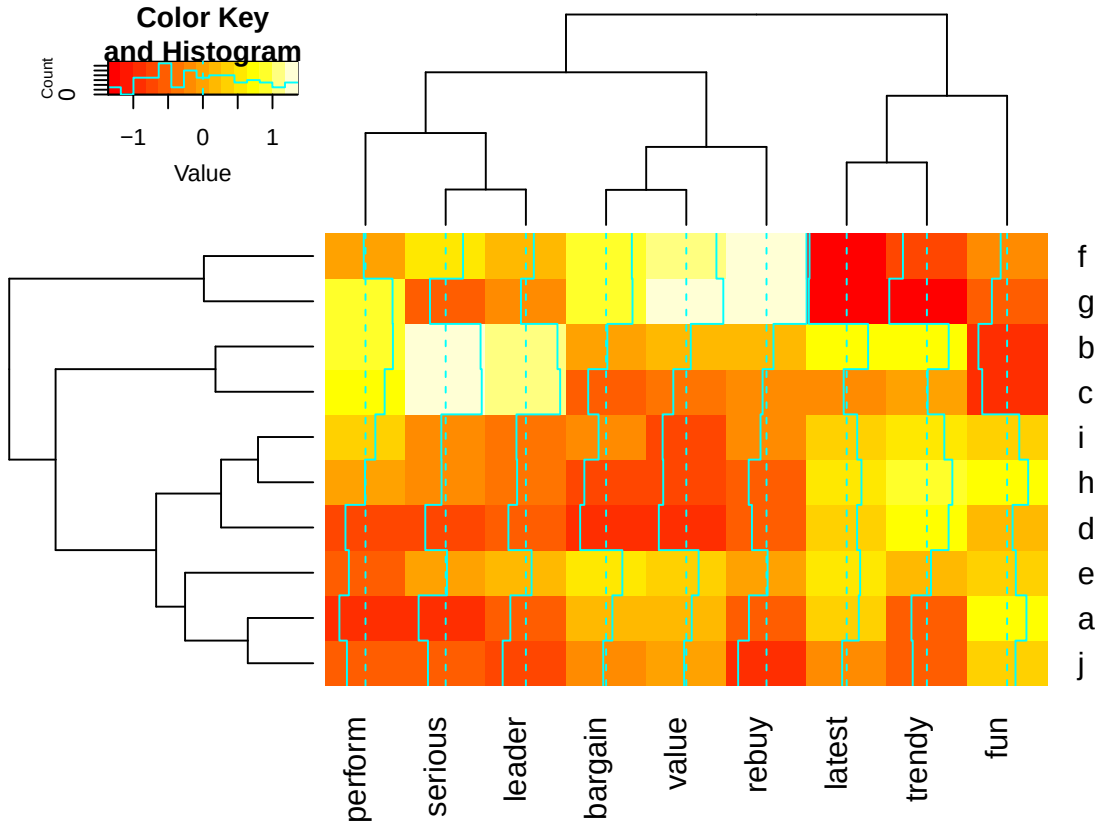
Eine **Heatmap** ist eine nützliche Darstellungsmöglichkeit, um solche Ergebnisse zu visualisieren und analysieren, da sie Datenpunkte durch die Intensitäten ihrer Werte färbt. Hierzu laden wir das Paket `gplots`.

```
library(gplots)
```

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

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

```
heatmap.2(as.matrix(brand.mean))
```



`heatmap.2()` sortiert die Spalten und Zeilen, um Ähnlichkeiten und Muster in den Daten hervorzuheben. Eine zusätzliche Analysehilfe ist das Spalten- und Zeilendendrogramm. Hier werden Beobachtungen die nahe beineinanderliegen in einem Baum abgebildet (näheres hierzu im Abschnitt *Clusteranalyse*.)

Auch hier sehen wir wieder die gleiche Zuordnung der Adjektive nach

- fun/latest/trendy
- rebuy/bargain/value
- perform/leader/serious

Zusätzlich können die Marken nach Ähnlichkeit bezüglich bestimmer Adjektive zugeordnet werden:

- f und g
- b und c
- i, h und d
- a und j

Hauptkomponentenanalyse (PCA)

Die PCA berechnet ein Variablenset (Komponenten) in Form von linearen Gleichungen, die die linearen Beziehungen in den Daten erfassen. Die erste Komponente erfasst so viel Streuung (Varianz) wie möglich von allen Variablen als eine einzige lineare Funktion. Die zweite Komponente erfasst unkorreliert zur

ersten Komponente so viel Streuung wie möglich, die nach der ersten Komponente verbleibt. Das geht so lange weiter, bis es so viele Komponenten gibt wie Variablen.

Bestimmung der Anzahl der Hauptkomponenten

Betrachten wir in einem ersten Schritt die wichtigsten Komponenten für die Brand-Rating-Daten. Wir finden die Komponenten mit prcomp(), wobei wir wieder nur die Bewertungsspalten 1-9 auswählen:

```
brand.pc <- prcomp(brand.sc[, 1:9])
summary(brand.pc)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     1.726 1.4479 1.0389 0.8528 0.79846 0.73133 0.62458
## Proportion of Variance 0.331 0.2329 0.1199 0.0808 0.07084 0.05943 0.04334
## Cumulative Proportion  0.331 0.5640 0.6839 0.7647 0.83554 0.89497 0.93831
##                           PC8     PC9
## Standard deviation     0.55861 0.49310
## Proportion of Variance 0.03467 0.02702
## Cumulative Proportion  0.97298 1.00000
```

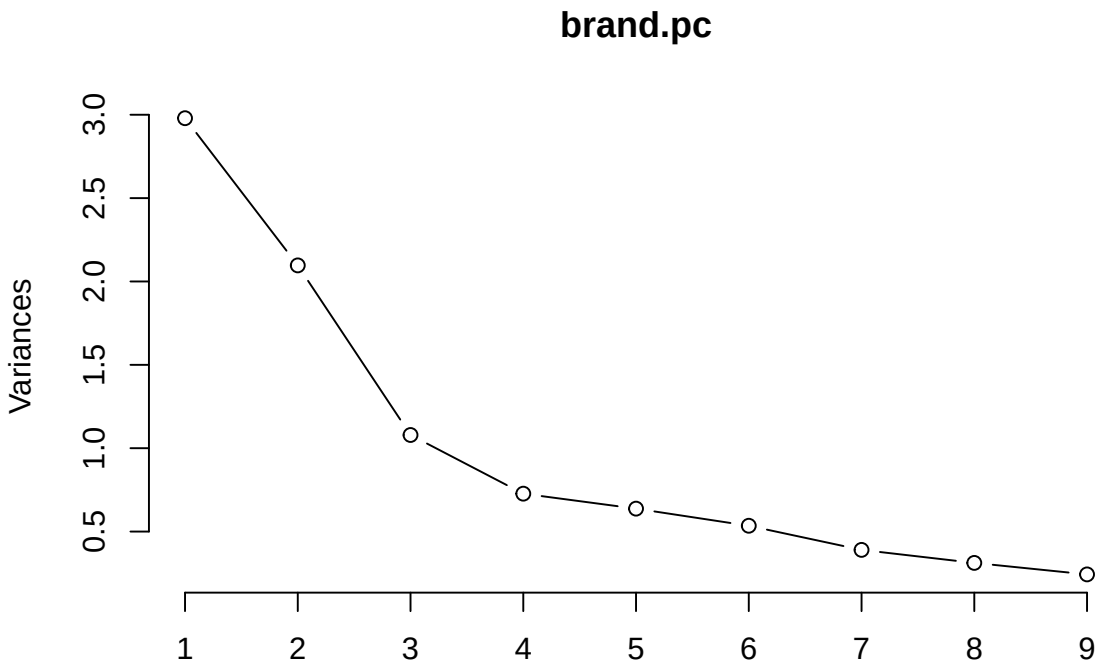
```
# Berechnung der Gesamtvarianz
Gesamtvarianz <- sum(brand.pc$sdev^2)

# Bei sum(brand.pc$sdev^2) wird berechnet:
# 1.726^2 + 1.4479^2 + 1.0389^2 + 0.8528^2 + 0.79846^2 + 0.73133^2 + 0.62458^2 + 0.55861^2 + 0.49310

# Varianzanteil der ersten Hauptkomponente
brand.pc$sdev[1]^2/Gesamtvarianz
```

```
## [1] 0.3310328
```

Scree-Plot

Der Standard-Plot `plot()` für die PCA ist ein **Scree-Plot**, Dieser zeigt uns in Reihenfolge der Hauptkomponenten jeweils die durch diese Hauptkomponente erfasste Streuung (Varianz). Wir plotten ein Liniendiagramm mit dem Argument `type = "l"` (`l` für Linie):

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



Wir sehen anhand des Scree-Plots, dass bei den Brand-Rating-Daten der Anteil der Streuung nach der dritten Komponente nicht mehr wesentlich abnimmt. Es soll die Stelle gefunden werden, ab der die Varianzen der Hauptkomponenten deutlich kleiner sind. Je kleiner die Varianzen, desto weniger Streuung erklärt diese Hauptkomponente.

Elbow-Kriterium

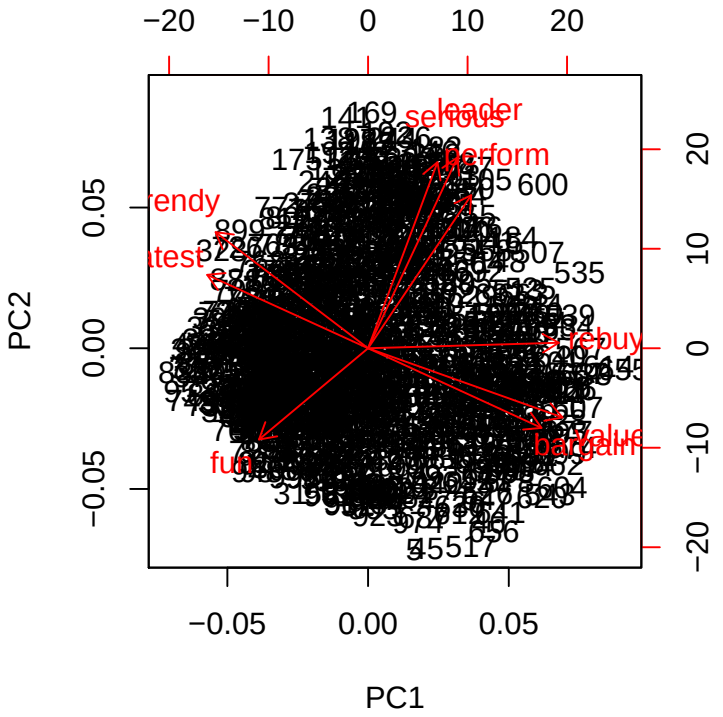
Nach diesem Kriterium werden alle Hauptkomponenten berücksichtigt, die links von der Knickstelle im Scree-Plot liegen. Gibt es mehrere Knicks, dann werden jene Hauptkomponenten ausgewählt, die links vom rechtesten Knick liegen. Gibt es keinen Knick, dann hilft der Scree-Plot nicht weiter. Bei den Brand-Rating-Daten tritt der Ellbogen, je nach Betrachtungsweise, entweder bei drei oder vier Komponenten auf. Dies deutet darauf hin, dass die ersten zwei oder drei Komponenten die meiste Streuung in den Brand-Rating-Daten erklären.

Biplot

Eine gute Möglichkeit die Ergebnisse der PCA zu analysieren, besteht darin, die ersten Komponenten zuzuordnen, die es uns ermöglichen, die Daten in einem niedrigdimensionalen Raum zu visualisieren. Eine gemeinsame Visualisierung ist ein Biplot. Dies ist ein zweidimensionales Diagramm von Datenpunkten in Bezug auf die ersten beiden Hauptkomponenten, die mit einer Projektion der Variablen auf die Komponenten überlagert wird.

Dazu verwenden wir `biplot()`:

```
biplot(brand.pc)
```



Die Adjektiv-Gruppierungen auf den Variablen sind als rote Ladungspfeile sichbar. ZUsätzlich erhalten wir einen Einblick in die Bewertungscluster (als dichte Bereiche von Beobachtungspunkten). Der Biplot ist hier durch die große Anzahl an Beobachtung recht unübersichtlich.

Deshalb führen wir die PCA mit den aggregierten Daten durch:

```
brand.mean
```

```
##       perform     leader     latest        fun     serious     bargain
## a -0.88591874 -0.5279035  0.4109732  0.6566458 -0.91894067  0.21409609
## b  0.93087022  1.0707584  0.7261069 -0.9722147  1.18314061  0.04161938
## c  0.64992347  1.1627677 -0.1023372 -0.8446753  1.22273461 -0.60704302
## d -0.67989112 -0.5930767  0.3524948  0.1865719 -0.69217505 -0.88075605
## e -0.56439079  0.1928362  0.4564564  0.2958914  0.04211361  0.55155051
## f -0.05868665  0.2695106 -1.2621589 -0.2179102  0.58923066  0.87400696
## g  0.91838369 -0.1675336 -1.2849005 -0.5167168 -0.53379906  0.89650392
## h -0.01498383 -0.2978802  0.5019396  0.7149495 -0.14145855 -0.73827529
## i  0.33463879 -0.3208825  0.3557436  0.4124989 -0.14865746 -0.25459062
## j -0.62994504 -0.7885965 -0.1543180  0.2849595 -0.60218870 -0.09711188
##         value      trendy       rebuy
## a  0.18469264 -0.52514473 -0.59616642
## b  0.15133957  0.74030819  0.23697320
## c -0.44067747  0.02552787 -0.13243776
## d -0.93263529  0.73666135 -0.49398892
## e  0.41816415  0.13857986  0.03654811
## f  1.02268859 -0.81324496  1.35699580
## g  1.25616009 -1.27639344  1.36092571
## h -0.78254646  0.86430070 -0.60402622
## i -0.80339213  0.59078782 -0.20317603
## j -0.07379367 -0.48138267 -0.96164748
```

```
brand.mu.pc<- prcomp(brand.mean, scale=TRUE)
summary(brand.mu.pc)
```

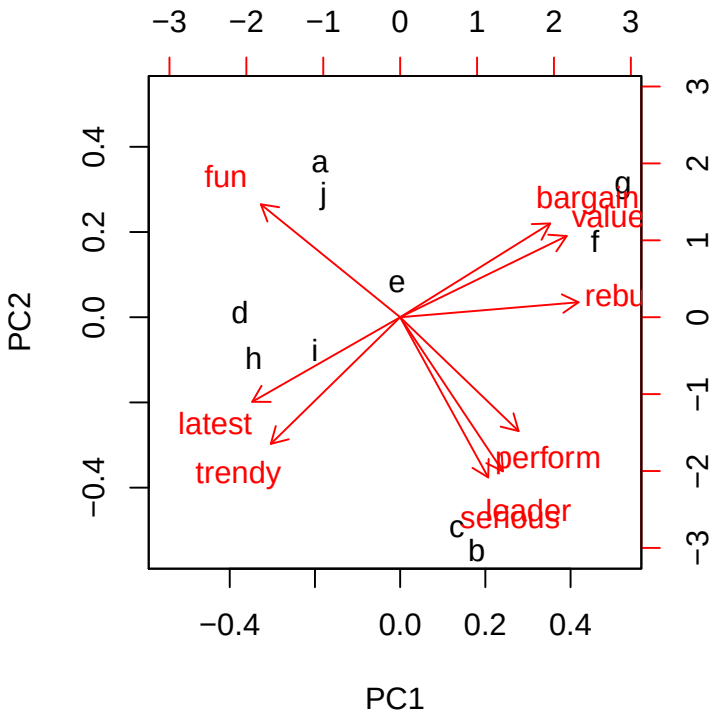
```
## Importance of components:
##                            PC1    PC2    PC3     PC4     PC5     PC6
## Standard deviation     2.1345 1.7349 0.7690 0.61498 0.50983 0.36662
## Proportion of Variance 0.5062 0.3345 0.0657 0.04202 0.02888 0.01493
## Cumulative Proportion  0.5062 0.8407 0.9064 0.94842 0.97730 0.99223
##                            PC7     PC8     PC9
## Standard deviation     0.21506 0.14588 0.04867
## Proportion of Variance 0.00514 0.00236 0.00026
## Cumulative Proportion  0.99737 0.99974 1.00000
```

Dem Befehl `prcomp()` wurde `Skalierung = TRUE` hinzugefügt, um die Daten neu zu skalieren. Obwohl die Rohdaten bereits skaliert waren, haben die aggregierten Daten eine etwas andere Skala als die standardisierten Rohdaten. Die Ergebnisse zeigen, dass die ersten beiden Komponenten für 84% der erklärbaren Streuung bei den aggregierten Daten verantwortlich sind.

Wahrnehmungsraum

Wenn ein Biplot Marken in einem zweidimensionalen Raum abbildet, dann wird dieser Raum **zweidimensionaler Wahrnehmungsraum** bezeichnet.

```
biplot(brand.mu.pc)
```



Der Biplot der PCA-Lösung für die Mittelwerte gibt einen interpretierbaren Wahrnehmungsraum, der zeigt, wo die Marken in Bezug auf die ersten beiden Hauptkomponenten liegen. Die Variablen auf den beiden Komponenten sind mit der PCA auf den gesamten Datensatz konsistent. Wir sehen vier Bereiche (Positionen) mit gut differenzierten Adjektiven und Marken.

Exploratorische Faktorenanalyse (EFA)

EFA ist eine Methode, um die Beziehung von Konstrukten (Konzepten), d. h. Faktoren zu Variablen zu beurteilen. Dabei werden die Faktoren als **latente Variablen** betrachtet, die nicht direkt beobachtet

werden können. Stattdessen werden sie empirisch durch mehrere Variablen beobachtet, von denen jede ein Indikator der zugrundeliegenden Faktoren ist. Diese beobachteten Werte werden als **manifeste Variablen** bezeichnet und umfassen Indikatoren. Die EFA versucht den Grad zu bestimmen, in dem Faktoren die beobachtete Streuung der manifesten Variablen berücksichtigen.

Das Ergebnis der EFA ist ähnlich zur PCA: eine Matrix von Faktoren (ähnlich zu den PCA-Komponenten) und ihre Beziehung zu den ursprünglichen Variablen (Ladung der Faktoren auf die Variablen). Im Gegensatz zur PCA versucht die EFA, Lösungen zu finden, die in den **manifesten variablen maximal interpretierbar** sind. Im allgemeinen versucht sie, Lösungen zu finden, bei denen eine kleine Anzahl von Ladungen für jeden Faktor sehr hoch ist, während andere Ladungen für diesen Faktor gering sind. Wenn dies möglich ist, kann dieser Faktor mit diesem Variablen-Set interpretiert werden. Innerhalb einer PCA kann die Interpretierbarkeit über eine **Rotation** (z. B. `varimax()`) erhöht werden.

Finden einer EFA Lösung

Als erstes muss die Anzahl der zu schätzenden Faktoren bestimmt werden. Hierzu verwenden wir zwei gebräuchliche Methoden:

1. Das Elbow-Kriterium

Den Scree-plot haben wir bereits bei der PCA durchgeführt. Ein Knick konnten wir bei der dritte oder vierten Hauptkomponente feststellen. Somit zeigt der Skreeplot eine 2 oder 3 Faktorenlösung an.

Durch das Paket `nFactors` bekommen wir eine formalisierte Berechnung der Scree-Plot Lösung mit dem Befehl `nScree()`

```
library(nFactors)
```

```
## Loading required package: MASS
## Loading required package: psych
## Loading required package: boot
##
## Attaching package: 'boot'
## The following object is masked from 'package:psych':
##
##     logit
## Loading required package: lattice
##
## Attaching package: 'lattice'
## The following object is masked from 'package:boot':
##
##     melanoma
##
## Attaching package: 'nFactors'
## The following object is masked from 'package:lattice':
##
##     parallel
```

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

```
##   noc naf nparallel nkaiser
## 1   3   2         3       3
```

`nScree` gibt vier methodische Schätzungen für die Anzahl an Faktoren durch den Scree-Plot aus. Wir sehen, dass drei von vier Methoden drei Faktoren vorschlagen.

2. Das Eigenwert-Kriterium

Der Eigenwert ist eine Metrik für den Anteil der erklärten Varianz. Die Anzahl Eigenwerte können wir über den Befehl `eigen()` ausgeben.

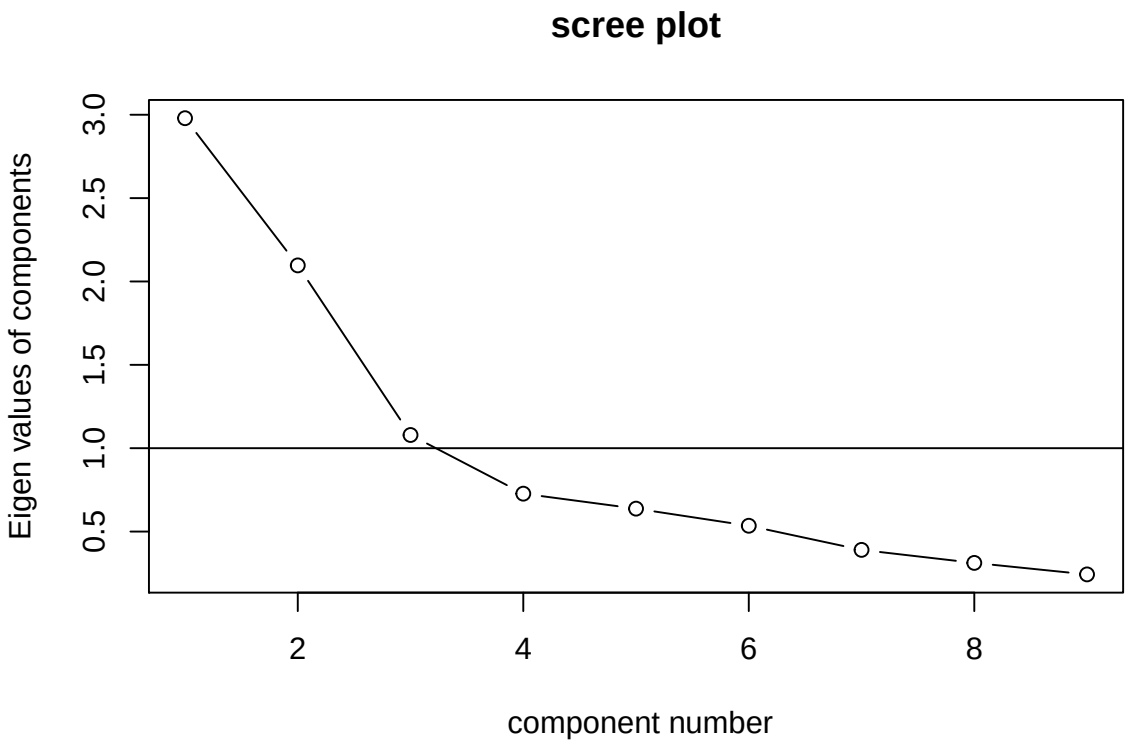
```
eigen(cor(brand.sc[, 1:9]))
```

```
## $values
## [1] 2.9792956 2.0965517 1.0792549 0.7272110 0.6375459 0.5348432 0.3901044
## [8] 0.3120464 0.2431469
##
## $vectors
##             [,1]        [,2]        [,3]        [,4]        [,5]
##  [1,] -0.2374679 -0.41991179  0.03854006  0.52630873  0.46793435
##  [2,] -0.2058257 -0.52381901 -0.09512739  0.08923461 -0.29452974
##  [3,]  0.3703806 -0.20145317 -0.53273054 -0.21410754  0.10586676
##  [4,]  0.2510601  0.25037973 -0.41781346  0.75063952 -0.33149429
##  [5,] -0.1597402 -0.51047254 -0.04067075 -0.09893394 -0.55515540
##  [6,] -0.3991731  0.21849698 -0.48989756 -0.16734345 -0.01257429
##  [7,] -0.4474562  0.18980822 -0.36924507 -0.15118500 -0.06327757
##  [8,]  0.3510292 -0.31849032 -0.37090530 -0.16764432  0.36649697
##  [9,] -0.4390184 -0.01509832 -0.12461593  0.13031231  0.35568769
##             [,6]         [,7]        [,8]        [,9]
##  [1,]  0.3370676  0.364179109 -0.14444718 -0.05223384
##  [2,]  0.2968860 -0.613674301  0.28766118  0.17889453
##  [3,]  0.1742059 -0.185480310 -0.64290436 -0.05750244
##  [4,] -0.1405367 -0.007114761  0.07461259 -0.03153306
##  [5,] -0.3924874  0.445302862 -0.18354764 -0.09072231
##  [6,]  0.1393966  0.288264900  0.05789194  0.64720849
##  [7,]  0.2195327  0.017163011  0.14829295 -0.72806108
##  [8,] -0.2658186  0.153572108  0.61450289 -0.05907022
##  [9,] -0.6751400 -0.388656160 -0.20210688  0.01720236
```

Der Eigenwert eines Faktors sagt aus, wie viel Varianz dieser Faktor an der Gesamtvarianz aufklärt. Laut dem Eigenwert-Kriterium sollen nur Faktoren mit einem Eigenwert größer 1 extrahiert werden. Dies sind bei den Brand-Rating Daten drei Faktoren, da drei Eigenwerte größer 1 sind.

Dies kann auch grafisch mit dem `VSS.Scree` geplotet werden.

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



Schätzung der EFA

Eine EFA wird geschätzt mit dem Befehl `factanal(x,factors=k)`, wobei `k` die Anzahl Faktoren angibt.

```
brand.fa<-factanal(brand.sc[, 1:9], factors=3)
brand.fa
```

```
##
## Call:
## factanal(x = brand.sc[, 1:9], factors = 3)
##
## Uniquenesses:
## perform  leader  latest     fun serious bargain   value  trendy   rebuy
##   0.624   0.327   0.005   0.794   0.530   0.302   0.202   0.524   0.575
##
## Loadings:
##         Factor1 Factor2 Factor3
## perform          0.607
## leader           0.810   0.106
## latest  -0.163           0.981
## fun             -0.398   0.205
## serious          0.682
## bargain  0.826          -0.122
## value    0.867          -0.198
## trendy  -0.356           0.586
## rebuy    0.499   0.296  -0.298
##
##                Factor1 Factor2 Factor3
## SS loadings      1.853   1.752   1.510
## Proportion Var   0.206   0.195   0.168
## Cumulative Var   0.206   0.401   0.568
##
## Test of the hypothesis that 3 factors are sufficient.
```

```
## The chi square statistic is 64.57 on 12 degrees of freedom.
## The p-value is 3.28e-09
```

Eine übersichtlichere Ausgabe bekommen wir mit dem `print` Befehl, in dem wir zusätzlich noch die Dezimalstellen kürzen mit `digits=2`, alle Ladungen kleiner als 0,5 ausblenden mit `cutoff=.5` und die Ladungen mit `sort=TRUE` so sortieren, dass die Ladungen, die auf einen Faktor laden, untereinander stehen.

```
print(brand.fa, digits=2, cutoff=.5, sort=TRUE)
```

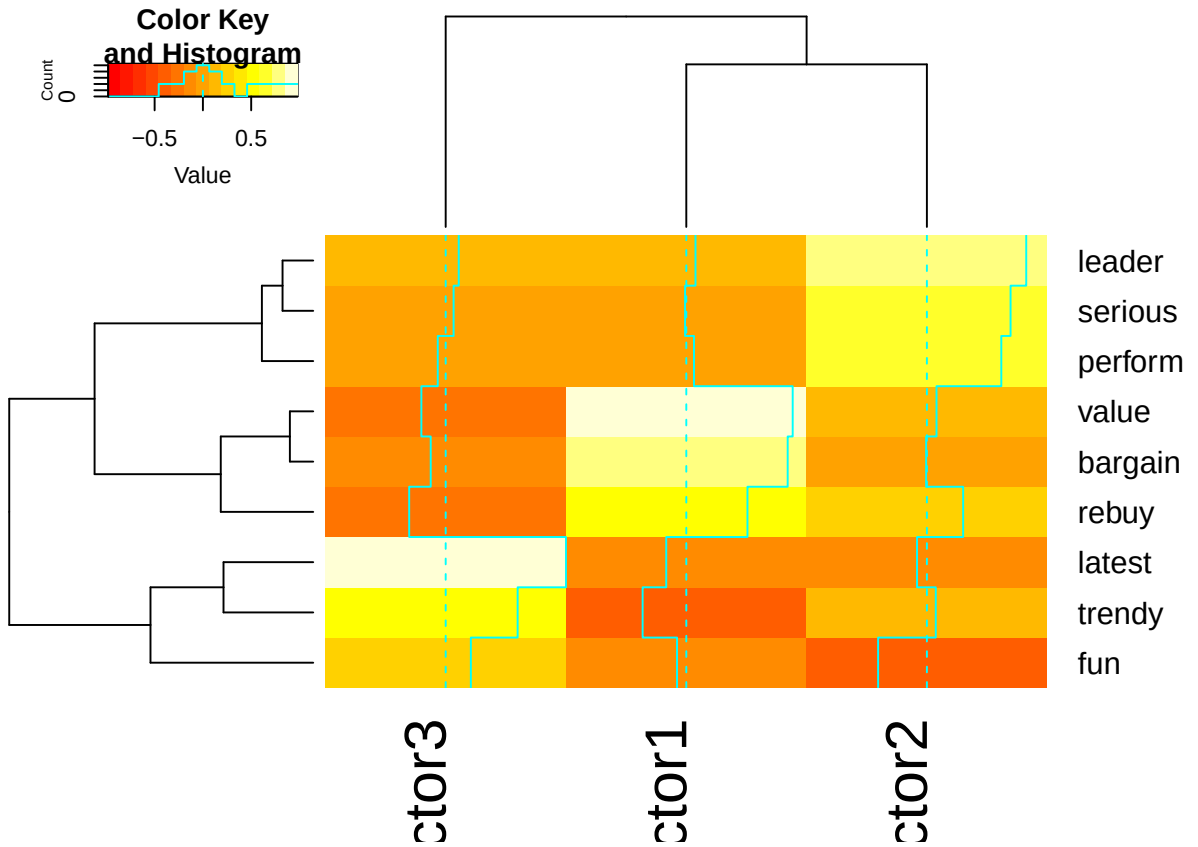
```
##
## Call:
## factanal(x = brand.sc[, 1:9], factors = 3)
##
## Uniquenesses:
## perform  leader  latest     fun serious bargain   value  trendy   rebuy
##    0.62    0.33    0.00    0.79    0.53    0.30    0.20    0.52    0.58
##
## Loadings:
##         Factor1 Factor2 Factor3
## bargain  0.83
## value    0.87
## perform          0.61
## leader           0.81
## serious          0.68
## latest                   0.98
## trendy                   0.59
## fun
## rebuy
##
##                Factor1 Factor2 Factor3
## SS loadings       1.85    1.75    1.51
## Proportion Var    0.21    0.19    0.17
## Cumulative Var    0.21    0.40    0.57
##
## Test of the hypothesis that 3 factors are sufficient.
## The chi square statistic is 64.57 on 12 degrees of freedom.
## The p-value is 3.28e-09
```

Standardmäßig wird bei `factanal()` eine Varimax-Rotation durchgeführt (das Koordinatensystem der Faktoren wird so rotiert, das eine optimale Zuordnung zu den Variablen erfolgt). Bei Varimax gibt es keine Korrelationen zwischen den Faktoren. Sollen Korrelationen zwischen den Faktoren zugelassen werden, empfielt sich die Oblimin-Rotation mit dem Argument `rotation="oblimin"` aus dem Paket `GPArotation`.

Heatmap mit Ladungen

In der obigen Ausgabe werden die Item-to-Faktor-Ladungen angezeigt. Im zurückgegebenen Objekt `brand.fa` sind diese als `$loadings` vorhanden. Wir können die Item-Faktor-Beziehungen mit einer Heatmap von `$loadings` visualisieren:

```
heatmap.2(brand.fa$loadings)
```



Das Ergebnis aus der Heatmap zeigt eine deutliche Trennung der Items in 3 Faktoren, die grob interpretierbar sind als **value**, **leader** und **latest**.

Berechnung der Faktor-Scores

Zusätzlich zur Schätzung der Faktorstruktur kann die EFA auch die latenten Faktorwerte für jede Beobachtung schätzen. Die gängige Extraktionsmethode ist die Bartlett-Methode.

```
brand.fa.ob <- factanal(brand.sc[, 1:9], factors=3, scores="Bartlett")
brand.scores <- data.frame(brand.fa.ob$scores)
head(brand.scores)
```

```
##      Factor1    Factor2    Factor3
## 1  1.9097351 -0.8668153  0.8448345
## 2 -1.5787358 -1.5067438 -1.1191067
## 3  1.1724921 -1.1298348 -0.2958264
## 4  0.4960526 -0.4295001  1.3020740
## 5  2.1137739 -2.0471144 -0.2104525
## 6  1.6906002  0.1545123  1.2186914
```

Wir können damit die Faktor-Scores verwenden, um die Positionen der Marken auf den Faktoren zu bestimmen.

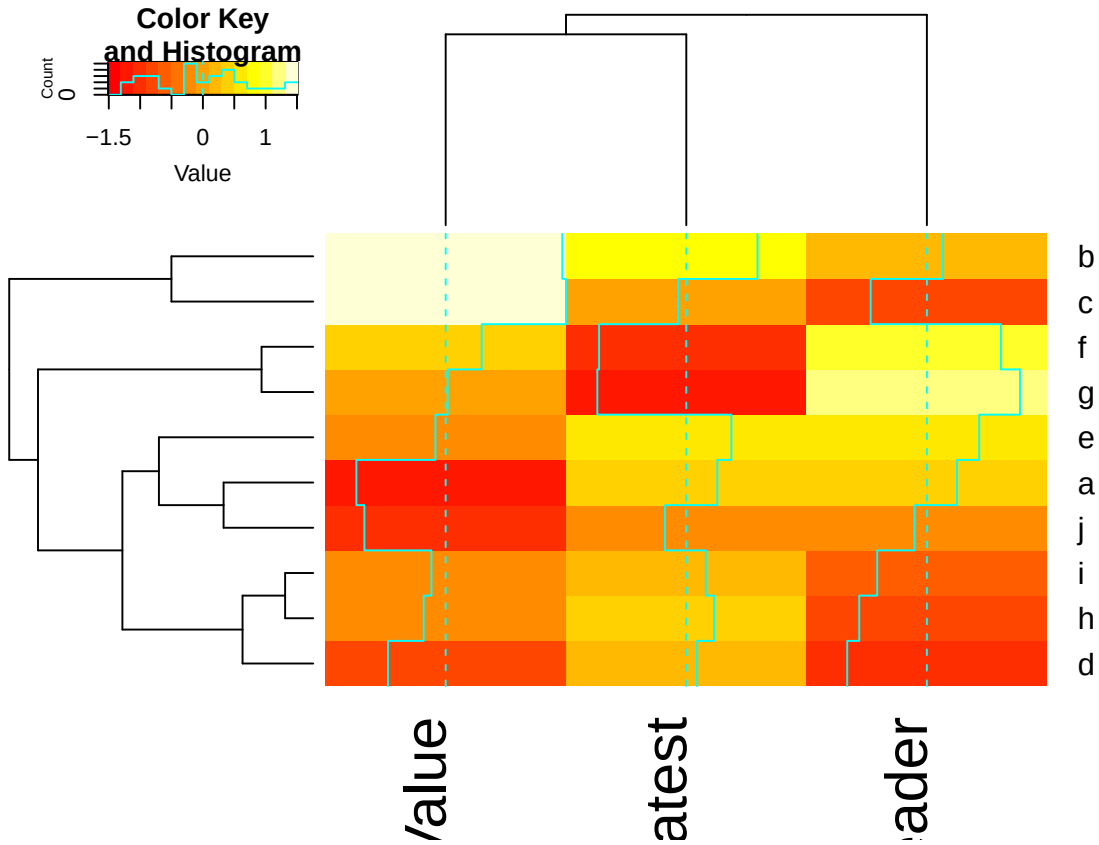
```
brand.scores$brand <- brand.sc$brand # Zuweisung der Markennamen zur Scores-Matrix
brand.fa.mean <- aggregate(. ~ brand, data=brand.scores, mean) # Aggregation Marken
rownames(brand.fa.mean) <- brand.fa.mean[, 1] # Fallbezeichnung mit Markennamen setzen
brand.fa.mean <- brand.fa.mean[, -1] # Erste Spalte löschen
names(brand.fa.mean) <- c("Leader", "Value", "Latest") # Spaltennamen neu zuweisen
brand.fa.mean
```

```
##       Leader       Value      Latest
## a  0.3778894 -1.12059561  0.38878416
## b  0.2021913  1.46306266  0.89377982
```

```
## c -0.7056202  1.50959052 -0.09947966
## d -0.9997980 -0.72387907  0.13460869
## e  0.6564191 -0.12905899  0.56566118
## f  0.9285971  0.45161454 -1.09464854
## g  1.1688809  0.02452425 -1.11657333
## h -0.8494187 -0.27625354  0.34995876
## i -0.6240513 -0.17881277  0.24499689
## j -0.1550895 -1.02019199 -0.26708797
```

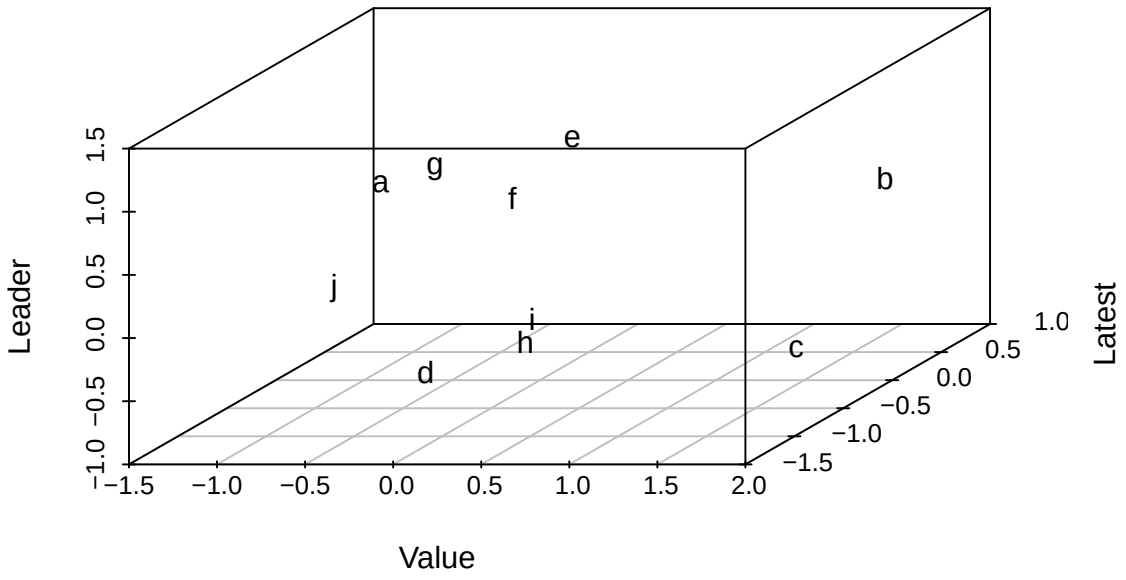
Mittels Heatmap kann dann sehr schnell analysiert werden, welche Marke auf welcher Dimension gute oder schlechte Ausprägungen hat.

```
heatmap.2(as.matrix(brand.fa.mean))
```



Drei Dimensionen lassen sich in einem dreidimensionalen Raum darstellen:

```
library(scatterplot3d)
attach(brand.fa.mean) # Datensatz zum Suchpfad hinzufügen
scatterplot3d(Leader~Value+Latest, pch=row.names(brand.fa.mean))
```




```
detach(brand.fa.mean) # Datensatz vom Suchpfad entfernen
```

Interne Konsistenz der Skalen

Das einfachste Maß für die **interne Konsistenz** ist die **Split-Half-Reliabilität**. Die Items werden in zwei Hälften unterteilt und die resultierenden Scores sollten in ihren Kenngrößen ähnlich sein. Hohe Korrelationen zwischen den Hälften deuten auf eine hohe interne Konsistenz hin. Das Problem ist, dass die Ergebnisse davon abhängen, wie die Items aufgeteilt werden. Ein üblicher Ansatz zur Lösung dieses Problems besteht darin, den Koeffizienten **Alpha (Cronbachs Alpha)** zu verwenden.

Der Koeffizient **Alpha** ist der Mittelwert aller möglichen Split-Half-Koeffizienten, die sich aus verschiedenen Arten der Aufteilung der Items ergeben. Dieser Koeffizient variiert von 0 bis 1. Formal ist es ein korrigierter durschnittlicher Korrelationskoeffizient.

Faustreglen für die Bewertung von Cronbachs Alpha:

Alpha	Bedeutung
größer 0,9	excellent
größer 0,8	gut
größer 0,7	akzeptabel
größer 0,6	fragwürdig
größer 0,5	schlecht

Wir bewerten nun die interne Konsistent der Itmes für die Konstrukte `Leader`, `Value` und `Latest`.

```
alpha(brand.sc[, c("leader","serious","perform")], check.keys=TRUE)
```

```
##
## Reliability analysis
## Call: alpha(x = brand.sc[, c("leader", "serious", "perform")], check.keys = TRUE)
##
```

```
##   raw_alpha std.alpha G6(smc) average_r S/N   ase     mean   sd
##       0.73      0.73    0.66      0.48 2.7 0.015 -7.5e-17 0.81
##
##  lower alpha upper     95% confidence boundaries
## 0.7 0.73 0.76
##
##  Reliability if an item is dropped:
##         raw_alpha std.alpha G6(smc) average_r S/N alpha se
## leader       0.53      0.53    0.36      0.36 1.1    0.030
## serious      0.67      0.67    0.50      0.50 2.0    0.021
## perform      0.73      0.73    0.57      0.57 2.7    0.017
##
##  Item statistics
##            n raw.r std.r r.cor r.drop     mean sd
## leader  1000  0.86  0.86  0.76   0.65  7.0e-17  1
## serious 1000  0.80  0.80  0.64   0.54 -1.5e-16  1
## perform 1000  0.77  0.77  0.57   0.48 -1.6e-16  1
```

```
alpha(brand.sc[, c("value", "bargain", "rebuy")], check.keys=TRUE)
```

```
##
## Reliability analysis
## Call: alpha(x = brand.sc[, c("value", "bargain", "rebuy")], check.keys = TRUE)
##
##   raw_alpha std.alpha G6(smc) average_r S/N   ase    mean   sd
##       0.8       0.8    0.75      0.57   4 0.011 9.7e-18 0.84
##
##  lower alpha upper     95% confidence boundaries
## 0.78 0.8 0.82
##
##  Reliability if an item is dropped:
##         raw_alpha std.alpha G6(smc) average_r S/N alpha se
## value        0.64      0.64    0.47      0.47 1.8   0.0230
## bargain      0.67      0.67    0.51      0.51 2.0   0.0207
## rebuy        0.85      0.85    0.74      0.74 5.7   0.0095
##
##  Item statistics
##            n raw.r std.r r.cor r.drop     mean sd
## value   1000  0.89  0.89  0.83   0.73  1.2e-16  1
## bargain 1000  0.87  0.87  0.80   0.70 -1.2e-16  1
## rebuy   1000  0.78  0.78  0.57   0.52  5.2e-17  1
```

```
alpha(brand.sc[, c("latest","trendy","fun")], check.keys=TRUE)
```

```
##
## Reliability analysis
## Call: alpha(x = brand.sc[, c("latest", "trendy", "fun")], check.keys = TRUE)
##
##   raw_alpha std.alpha G6(smc) average_r S/N   ase    mean   sd
##       0.6       0.6    0.58      0.33 1.5 0.022 4.4e-17 0.75
##
##  lower alpha upper     95% confidence boundaries
## 0.56 0.6 0.64
##
##  Reliability if an item is dropped:
##         raw_alpha std.alpha G6(smc) average_r  S/N alpha se
## latest       0.23      0.23    0.13      0.13 0.29    0.049
## trendy       0.39      0.39    0.25      0.25 0.65    0.038
## fun          0.77      0.77    0.63      0.63 3.37    0.014
##
```

```
##  Item statistics
##           n raw.r std.r r.cor r.drop     mean sd
## latest 1000  0.84  0.84  0.76   0.58 -9.7e-17  1
## trendy 1000  0.79  0.79  0.68   0.48  8.8e-17  1
## fun    1000  0.61  0.61  0.26   0.21  1.5e-16  1
```

Bis auf `Latest` sind alle Konstrukte bezüglich ihrer internen Konsistenz akzeptabel. Bei dem Konstrukt `Latest` können wir durch Elimination von `fun` das Cronbachs Alpha von einem fragwürdigen Wert auf einen akteptablen Wert von 0,77 erhöhen.

Das Argument `check.keys=TRUE` gibt uns eine Warung aus, sollte die Ladung eines oder mehrerer Items negativ sein. Dies ist hier nicht der Fall, somit müssen auch keine Items recodiert werden.

Übung

Führen Sie eine Dimensionsreduktion mit den nichtskalierten original Daten durch. Berechenn Sie zur Interpretaion keine Faktor-Scores, sondern berechnen Sie stattdessen den Mittelwert der Variablen, die hoch (mindestens 0,5) auf einen Faktor laden. Für die Berechnung verwenden Sie

```
Datensatz$Neue_Variable <- apply(Datensatz[,c("Variable1","Variable2", "etc..")], 1, mean, na.rm=TRU
```

Literatur

- Chris Chapman, Elea McDonnell Feit (2015): *R for Marketing Research and Analytics*, Kapitel 8.1-8.3
- Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani (2013): *An Introduction to Statistical Learning – with Applications in R*, http://www-bcf.usc.edu/~gareth/ISL/, Kapitel 10.2, 10.4
- Reinhold Hatzinger, Kurt Hornik, Herbert Nagel (2011): *R – Einführung durch angewandte Statistik.* Kapitel 11
- Maike Luhmann (2015): R für Einsteiger, Kapitel 19

Versionshinweise:

- Datum erstellt: 2017-02-18
- R Version: 3.3.2