Strukturgleichungsmodelle mit R und lavaan analysieren: Kurzeinführung

Christina Werner · Frühling 2015 · Universität Zürich

Diese Einführung bezieht sich auf die lavaan-Version 0.5-17 (Stand Oktober 2014). Ältere Versionen verwenden teils abweichende Anweisungen, zukünftige möglicherweise auch.

Ausführlich beschrieben ist die Benützung von lavaan im englischsprachigen Artikel des lavaan-Autors Yves Rosseel:

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software, 48* (2), 1–36. Available from: ▷ http://www.jstatsoft.org/v48/i02

1 R und lavaan installieren

Zur Analyse von Strukturgleichungsmodellen mit dem R-Zusatzpaket «lavaan» installieren Sie bitte zuerst die Statistik-Programmumgebung R, anschliessend das lavaan-Paket.

1. **R herunterladen:** R können Sie für Mac OS X, Windows und Linux als freie Software herunterladen von CRAN, dem Comprehensive R Archive Network:

   ▷ http://cran.r-project.org

   Am günstigsten nutzen Sie eine lokale Kopie von CRAN, z. B. bei der ETH Zürich:

   ▷ http://stat.ethz.ch/CRAN/

2. **R installieren:** Aus der heruntergeladenen exe-(Windows) bzw. pkg-Datei (Mac) können Sie R auf Ihrem System installieren. Falls Sie Linux benutzen, installieren Sie R am einfachsten direkt über die Paketverwaltung Ihrer Linux-Distribution.

3. **lavaan-Paket installieren:** Nach dem Start von R können Sie das lavaan-Paket mit folgender Anweisung installieren (Eingabe auf der R-Konsole, Internetverbindung vorausgesetzt):

   ```
   install.packages("lavaan")
   ```

   Alternativ können Sie unter einer grafischen R-Oberfläche Pakete auch über ein Menü installieren. Auf dem Mac: R-Menü *Packages and Data* → *Package Installer*; unter Windows: *Packages* → *Install Packages*

2 Ablauf einer R-Sitzung

1. **R starten**

2. **Verzeichnis wechseln:** In R sollten Sie in das Verzeichnis wechseln, in dem sich die Daten befinden (und allenfalls ein schon vorhandenes R-Skript mit Analyse-Anweisungen). Auf dem Mac: R-Menü *Misc* → *Change Working Directory*; unter Windows: *File* → *Change Directory*

   Alternativ können Sie dies in der R-Konsole mit der Funktion `setwd()` (set working directory) tun:

   ```
   setwd("langer/pfad/zum/verzeichnis/mit/den/daten")
   ```

   Bei Pfadangaben beachten Sie bitte, dass R zwischen Verzeichnisnamen Vorwärts-Schrägstriche / erwartet (nicht Windows-übliche Backslashes \).

   Empfehlung:

   Speichern Sie Datendateien und die dazugehörigen R-Skripte (Syntax) jeweils in das gleiche Verzeichnis und wechseln Sie mit R in dieses Verzeichnis. Dann benötigen Sie beim Einlesen der Daten keine weiteren Pfadangaben. Auch das Speichern von Ergebnissen oder des R-Workspace (siehe unten bei 5.) geschieht dann in diesem Verzeichnis.

3. **Skript mit Analyse-Anweisungen schreiben:** Falls noch kein R-Skript mit Analyse-Anweisungen existiert, öffnen Sie bitte ein neues, leeres Skript (eine Art Textdatei), in welches Sie ihre Anweisungen schreiben können. Mac: Menü *File* → *New Document*; Windows: *File* → *New Script*.

   Falls schon ein R-Skript mit Analyse-Anweisungen existiert, können Sie dieses öffnen: Menü *File* → *Open Document* bzw. *File* → *Open Script*

   Empfehlung:

   Es ist sehr sinnvoll, R-Anweisungen in ein Skript zu schreiben und nicht direkt in die R-Konsole zu tippen. Das Skript können Sie als Ganzes speichern. Damit haben Sie alle Anweisungen so aufgeschrieben, dass Sie sie später einfach erneut ausführen können. Im Skript können Sie ausserdem leicht Fehler korrigieren und Anweisungen so ändern, bis die Analyse wie gewünscht läuft.

4. **Skript ausführen:** In einem R-Skript-Fenster können Sie Anweisungen auf verschiedene Weise ausführen:

   - **Zeile:** Die Skript-Zeile, in der der Cursor gerade steht, können Sie ausführen über das Menü *Edit* → *Execute* (Mac) bzw. *Edit* → *Run line or selection* (Windows). Damit wird diese Zeile auf der R-Konsole ausgeführt und allfällige Ergebnisse oder Meldungen dort angezeigt. Alternativ geht dies mit dem Tastaturkürzel Cmd + Enter (Mac) bzw. Ctrl + R (Windows).

- **Abschnitt:** Sie können einen Abschnitt des Skripts markieren. Über das Menü *Edit* → *Execute* (Mac) bzw. *Edit* → *Run line or selection* (Windows) wird dann genau der markierte Abschnitt auf der R-Konsole ausgeführt. Auch dies geht mit dem Tastaturkürzel Cmd + Enter (Mac) bzw. Ctrl + R (Windows).
- **Alles:** Sie können das komplette Skript auf einmal ausführen: Auf dem Mac Tastaturkürzel Cmd + A (um alles zu markieren), gefolgt von Cmd + Enter (zum Ausführen), analog unter Windows Ctrl + A, gefolgt von Ctrl + R. Unter Windows gibt es alternativ auch das Menü *Edit* → *Run all*.

Empfehlung:

Wenn man ein neues Skript schreibt, ist es besser, die Anweisungen einzeln nacheinander auszuführen. Bei allfälligen Fehlermeldungen oder unerwarteten Ergebnissen sieht man so, wo genau das Problem entsteht.

5. **Speichern:** Spätestens am Ende der Sitzung sollten Sie ihr Skript speichern, am besten unter einem Dateinamen, welcher auf `.R` endet: Menü *File* → *Save* (oder Tastaturkürzel Cmd + S bzw. Ctrl + S)

   Darüber hinaus können Sie den gesamten R-Workspace speichern: Der Workspace enthält alle R-Objekte, also alle Daten, Modelle, aber auch Ergebnisse, denen Sie einen Namen gegeben haben. Auf dem Mac: Menü *Workspace* → *Save Default Workspace*; Windows (in der R-Konsole): *File* → *Save Workspace*

   Verwenden Sie dabei den voreingestellten Dateinamen `.Rdata`, so wird Ihr Workspace beim nächsten Start von R im gleichen Verzeichnis automatisch wieder geladen, so dass alle Objekte der vorigen Sitzung wieder verfügbar sind.

   Um R später gleich in diesem Verzeichnis zu starten (und nicht erst über das Menü dorthin zu wechseln), können Sie im Finder (Mac) bzw. Explorer (Windows) auf die R-Skript-Datei klicken (vorausgesetzt, `.R`-Dateien werden automatisch mit dem Programm R geöffnet; dies kann man z. B. im Mac-Finder über das Kontextmenü *Get Info* einstellen).

Die Linux-Version von R besitzt keine eigene graphische Oberfläche mit integriertem Skript-Editor. Der hier beschriebene Ablauf einer R-Sitzung ist daher nicht direkt übertragbar. Sehr hilfreich ist unter Linux z. B. der Editor Emacs, mit dem man ebenso Zeilen, Abschnitte oder ein komplettes R-Skript direkt aus dem Editor heraus ausführen kann (Emacs-Mode ESS: «Emacs Speaks Statistics»). Für aufwendige Analysen ist dies auch auf Macs und unter Windows eine empfehlenswerte Alternative zum R-eigenen Skript-Editor.

3 Analyse eines Strukturgleichungsmodells

Das Analysieren eines Strukturgleichungsmodells mit R und dem lavaan-Paket umfasst mehrere Schritte. Vorab muss das lavaan-Paket installiert sein (nur einmalig; z. B. mit `install.packages("lavaan")`, vgl. Abschnitt 1) und vor jeder Analyse mit der Anweisung `library()` geladen werden, am besten gleich zu Beginn:

```
library("lavaan")
```

Dann folgen die Schritte (1) Daten einlesen, (2) Modell definieren, (3) Modell analysieren und (4) Ergebnisse anschauen. Das Beispielskript in Abschnitt 4 dieser Einführung verdeutlicht dies.

3.1 Schritt 1: Daten einlesen

Je nach Datenformat gibt es zum Einlesen verschiedene R-Funktionen, z. B.:

- `read.csv()` zum Einlesen von Comma-Separated Values (Werte durch Komma getrennt, Zahlen mit Dezimalpunkt). Das CSV-Format kann beispielsweise von Tabellenkalkulationsprogrammen wie Excel geschrieben und gelesen werden.
- `read.csv2()` zum Einlesen von Daten in der deutschsprachigen Variante von Comma-Separated Values (Werte durch Semikolon getrennt, Zahlen mit Dezimalkomma), welche z. B. von Excel auf deutschsprachigen Systemen verwendet wird.
- `read.table()` für Text-Datendateien mit beliebigen (angebbaren) Dezimal- und Trennzeichen (Voreinstellung: Dezimalpunkt, durch Leerzeichen getrennt; Details siehe `?read.table`)
- `read.spss()` aus dem R-Paket foreign (das Paket foreign ist bereits mit der R-Standardinstallation vorhanden, muss also nicht erst installiert, aber ggf. geladen werden).

Den eingelesenen Daten geben Sie einen Namen, damit sie als R-Objekt verfügbar werden (andernfalls würden die Daten einfach nur am Bildschirm ausgegeben). Beispiele:

```
meine.daten <- read.csv("datendatei.csv")

library("foreign")
meine.spss.daten <- read.spss("datendatei.sav", to.data.frame=TRUE)
```

Die eingelesenen Daten sind dann in R unter dem Namen `meine.daten` verfügbar (bzw. `meine.spss.daten`). Solche Namen sind R-intern, frei wählbar und können, aber müssen

nichts mit dem Namen einer Datei zu tun haben, aus der die Daten ursprünglich eingelesen wurden.

Mit der «Structure»-Funktion `str()` erhält man eine Übersicht, welche Variablen und wieviele Fälle in den Daten enthalten sind:

```
str(meine.daten)
```

3.2 Schritt 2: Modell definieren

Ein Strukturgleichungsmodell wird bei lavaan mit bestimmten Anweisungen definiert, die sich an allgemeine R-Anweisungen anlehnen, aber etwas davon unterscheiden. Allgemein steht in R die Formel-Schreibweise `y ~ x1 + x2` für «*y* wird vorhergesagt durch x_1 und x_2» (z. B. für das Festlegen einer Regressionsgleichung). In lavaan gibt es angelehnt daran die folgenden Festlegungen:

- `=~` bedeutet **wird gemessen durch**:

  ```
  xi1 =~ x1 + x2 + x3
  ```

  Auf der linken Seite von `=~` steht eine latente Variable, die durch die rechts davon stehenden beobachteten Variablen gemessen (operationalisiert) wird. Hier wird beispielsweise eine latente Variable namens `xi1` gemessen durch beobachtete Variablen namens `x1`, `x2` und `x3` (das Plus-Zeichen bedeutet nur, dass alle diese Variablen Messungen von `xi1` sein sollen, nicht etwa die Summe davon). In methodischen Texten zu Strukturgleichungsmodellen werden latente Variablen oft mit griechischen Buchstaben bezeichnet. In dem Fall steht für eine latente unabhängige (Prädiktor-)Variable der Buchstabe ξ (xi).

  In eigenen Daten sollte man natürlich inhaltliche Bezeichnungen der latenten Variablen verwenden, und statt der generischen *x*-Variablennamen stehen die Namen der zugehörigen beobachteten Variablen aus den eingelesenen Daten (so, wie sie bspw. in der ersten Zeile einer csv-Datendatei oder in einer SPSS-Datendatei enthalten sind). Inhaltliches Beispiel:

  ```
  burnout =~ erschoepfung + depersonalisation
  ```

  Hierfür müssten also Variablen namens `erschoepfung` und `depersonalisation` in den eingelesenen Daten enthalten sein.

- `~` bedeutet **wird vorhergesagt durch**:

  ```
  eta1 ~ xi1 + xi2
  ```

  Links von `~` steht eine latente Variable (die schon mit `=~` definiert wurde) oder eine beobachtete Variable aus den eingelesenen Daten, rechts können ebenfalls latente

oder beobachtete Variablen stehen. Hier wird also die Variable `eta1` vorhergesagt durch `xi1` und `xi2`. Inhaltliches Beispiel:

```
burnout ~ ueberstunden + kundenbeschwerden
```

Hier müssten `ueberstunden` und `kundenbeschwerden` entweder beobachtete Variablen aus den eingelesenen Daten sein oder latente Variablen, die schon mit `=~` definiert worden sind.

- `~~` bedeutet **kovariiert mit**:

  ```
  xi1 ~~ xi2
  ```

  Dies definiert einen ungerichteten Zusammenhang (Kovarianz/Korrelation) zwischen der linken und rechten Variablen, hier also zwischen `xi1` und `xi2`.

  Die Kovarianz einer Variablen mit sich selbst ist ihre Varianz. Daher kann man so auch die Varianzen (bzw. Residual- oder Messfehlervarianzen) von Variablen bezeichnen:

  ```
  xi1  ~~ xi1
  eta1 ~~ eta1
  x1   ~~ x1
  ```

  Für eine unabhängige (exogene) Variable `xi1` bezeichnet dies die Varianz von `xi1`. Für eine abhängige (endogene) Variable `eta1` bezeichnet dies die *Residual*varianz, also die Varianz von `eta1`, die nicht durch andere Variablen im Modell erklärt wird. Bei einer beobachteten Variable `x1`, die lediglich zur Messung einer latenten Variable dient (Indikatorvariable), wäre dies dann die *Messfehler*varianz von `x1`.

- `~ 1` steht für eine **Regressionskonstante** (ein Interzept) bzw. einen Mittelwert:

  ```
  eta1 ~ 1
  ```

  Als Sonderfall des Symbols `~` («wird vorhergesagt durch») bedeutet die Vorhersage durch die Konstante 1, dass im Modell auch eine Regressionskonstante mit geschätzt wird (bzw. für unabhängige/exogene Variablen ihr Mittelwert). Wenn also die Variable `eta1` (in griechischer Notation: η_1) hier eine abhängige/endogene Variable ist, die durch `xi1` und `xi2` vorhergesagt wird (ξ_1, ξ_2), so würde hier nicht nur diese Gleichung geschätzt (γ: Regressionsgewichte/Pfadkoeffizienten gamma; ζ: Vorhersagefehler zeta)

  $$\eta_1 = \quad \gamma_{11}\xi_1 + \gamma_{12}\xi_2 + \zeta_1,$$

  sondern

  $$\eta_1 = \alpha_1 + \gamma_{11}\xi_1 + \gamma_{12}\xi_2 + \zeta_1$$

mit einer Regressionskonstanten α_1 (alpha; relevant z. B. für längsschnittliche Analysen oder Gruppenvergleiche).

Mit all diesen Anweisungen zusammen definiert man das Modell, welches analysiert werden soll. Dabei steht jede Anweisung in einer eigenen Zeile, und die gesamte Modelldefinition als Text in Hochkommata `' '`. Das Modell erhält dann einen Namen, durch den es als R-Objekt ansprechbar ist. Beispiel:

```
mein.modell <- '
  # Messmodelle
  xi1  =~ x1 + x2 + x3
  xi2  =~ x4 + x5 + x6
  eta1 =~ y1 + y2

  # Strukturmodell
  eta1 ~  xi1 + xi2
'
```

Wie auch sonst in R-Skripten können Leerzeilen beliebig zur Gliederung verwendet werden. Gross- und Kleinschreibung werden unterschieden (`Mein.Modell` wäre also etwas anderes als `mein.modell`). Nach dem Raute-Zeichen # ist aller Text der Zeile ein Kommentar.

Nützliche Voreinstellungen

Bei der Analyse des Modells verwendet das lavaan-Paket – genauer gesagt: die Analyse-Funktionen `cfa()` und `sem()` – Voreinstellungen, welche die Modell-Definition sehr vereinfachen:

- **Varianzen:** Varianzen von unabhängigen Variablen, Residualvarianzen von abhängigen Variablen bzw. Messfehlervarianzen werden automatisch geschätzt. Wird z. B. die latente Variable `xi1` durch die beobachteten Variablen `x1`, `x2` und `x3` gemessen,

  ```
  xi1 =~ x1 + x2 + x3
  ```

  so muss man die Varianz der latenten Variable und die Messfehlervarianzen nicht extra aufführen (wie hier als Beispiel gemacht),

  ```
  xi1 ~~ xi1
  x1  ~~ x1
  x2  ~~ x2
  x3  ~~ x3
  ```

sondern diese Varianzen werden von den Analysefunktionen `cfa()` und `sem()` automatisch mit geschätzt.

- **Kovarianzen:** Kovarianzen von unabhängigen Variablen werden automatisch geschätzt. Für zwei unabhängige (exogene) Variablen `xi1` und `xi2` würde man normalerweise annehmen, dass diese korreliert sein können (wenn nicht aus inhaltlichen Gründen ausdrücklich Unkorreliertheit erwartet würde). Von Hand könnte man also festlegen:

  ```
  xi1 ~~ xi2
  ```

  Auch dies wird aber schon automatisch mit berücksichtigt.

- **Skalierung:** Latente Variablen erhalten automatisch eine Masseinheit (sie werden «skaliert»). Latente Variablen sind nicht direkt beobachbar und haben damit keine eindeutigen Masseinheiten (wie ein Intelligenztest in IQ-Punkten, Gewicht in kg, etc.). Um für eine latente Variable eine sinnvolle Skala zu erhalten (also festzulegen, was z. B. ein Unterschied von 1 auf dieser latenten Variable bedeutet), wird normalerweise die Ladung von genau einer ihrer beobachteten (Indikator-)Variablen auf 1 festgesetzt. Dadurch erhält die latente Variable die gleiche Skala wie diese beobachtete Variable: Bei einer Erhöhung der latenten Variable um 1 erhöht sich auch die beobachtete um 1. Von Hand könnte man entsprechend festlegen:

  ```
  xi1 =~ 1*x1 + x2 + x3
  ```

  Durch den vorgestellten Ausdruck `1*` würde die Faktorladung von `x1` auf der latenten Variable `xi1` auf 1 gesetzt. Auch dies geschieht aber bereits automatisch.

  Empfehlung:

  Um diese Voreinstellung auszunutzen, ist es sinnvoll, bei der Aufzählung von Indikatorvariablen in der Modelldefinition die vermutlich reliabelste und valideste Indikatorvariable zuerst zu nennen (hier `x1`), da die erstgenannte automatisch zum Skalieren der latenten Variable verwendet wird.

Die hier gezeigten Anweisungen zur Modelldefinition genügen für viele Modelle. Spezielle Anweisungen beispielsweise zum Gleichsetzen von Parametern und zum Festlegen von (Start-)Werten findet man im zu Anfang genannten Artikel von Rosseel (2012) sowie in der lavaan-Hilfe. Auf der R-Konsole (nach Laden des lavaan-Pakets): `?model.syntax`

3.3 Schritt 3: Modell analysieren

Nachdem das Modell definiert ist, können die Daten analysiert werden. Dies geschieht in der Regel mit einer dieser beiden lavaan-Funktionen:

- `cfa()` für konfirmatorische Faktorenanalysen (confirmatory factor analyses)
- `sem()` für Strukturgleichungsmodelle (structural equation models) generell

Die Funktionen haben viele Optionen (siehe die Hilfe auf der R-Konsole: `?cfa` oder `?sem`). Davon muss man aber meist nur wenige nennen. Auf jeden Fall braucht man Modell und Daten:

- **Modell** angeben: `model = mein.modell`
- **Daten** angeben: `data = meine.daten`

Sinnvoll ist oft auch das Festlegen der Schätzmethode und das Behandeln von fehlenden Werten:

- **Schätzmethode** festlegen: `estimator = ...`
  Üblicherweise wird für Strukturgleichungsmodelle mit quantitativen Variablen Maximum Likelihood-(ML-)Schätzung verwendet: `estimator = "ml"`. ML-Schätzung beruht auf der Voraussetzung multivariat normalverteilter Variablen. Bei nichtnormalen Daten gibt es Varianten der ML-Schätzung, die die Standardfehlerschätzungen und die χ^2-Teststatistik entsprechend anpassen, z. B. die Versionen `estimator = "mlm"` oder `"mlr"` mit robuster Standardfehlerschätzung und Satorra-Bentler bzw. Yuan-Bentler scaled χ^2-Teststatistik. Weitere Möglichkeiten sind z. B. `estimator = "wlsmv"` (Alternative für grobstufige / ordinalskalierte Variablen) oder `estimator = "uls"` (ungewichtete Kleinstquadrateschätzung, ohne Verteilungsvoraussetzung).
- **Fehlende Werte** behandeln: `missing = ...`
  Üblicherweise werden Fälle mit fehlenden Werten bei der Analyse ausgeschlossen. Bei Maximum Likelihood-(ML-)Schätzung ist es aber möglich, unter bestimmten Voraussetzungen auch Fälle mit unvollständigen Daten einzubeziehen und so mehr Information zu nutzen (wenn Werte «missing at random», MAR, oder «missing completely at random», MCAR, sind). Dies kann mit der Option `missing = "fiml"` («full information maximum likelihood») festgelegt werden.

Insgesamt kann eine einfache Analyse-Anweisung also z. B. so aussehen (das Ergebnis der Analyse ist selbst ein R-Objekt, hier mit dem Namen `ergebnis` versehen):

```
ergebnis <- sem(model=mein.modell, data=meine.daten, estimator="mlr")
```

3.4 Schritt 4: Ergebnisse anschauen

Zum Betrachten der Ergebnisse kann man die Funktion `summary()` mit verschiedenen Optionen verwenden:

- `fit.measures = TRUE` bewirkt zusätzlich zum χ^2-Test die Ausgabe weiterer Modellgütekriterien.

- `standardized = TRUE` gibt zur leichteren Interpretation eine standardisierte Lösung aus, d. h. Parameterschätzungen für den Fall, dass alle latenten und beobachteten Variablen im Modell standardisiert sind (Standardabweichung = 1).

  Unstandardisierte Parameterschätzungen können durch unterschiedliche Masseinheiten (Varianzen) der Variablen beliebige Werte aufweisen und sind daher nicht immer einfach interpretierbar.

  Bei standardisierten Schätzungen sind dagegen Parameter für Beziehungen zwischen Variablen vom Wertebereich wie Korrelationen oder standardisierte Regressionskoeffizienten interpretierbar (normalerweise zwischen −1 und +1). Bei Varianzen sind die Schätzungen interpretierbar als Varianzanteile zwischen 0 und 1, d. h. als unerklärter Varianzanteil bei abhängigen Variablen bzw. als Messfehlervarianz bei Indikatorvariablen. Durch den begrenzten Wertebereich ist damit die inhaltliche Bedeutsamkeit der Werte meist einfacher abzuschätzen. Wenn aber Varianzunterschiede zwischen Variablen wichtig sind (z. B. beim Vergleich zwischen Gruppen oder Messzeitpunkten), können unstandardisierte Schätzungen aussagekräftiger sein.

- `rsquare = TRUE` liefert R^2-Werte (erklärte Varianzanteile) für alle abhängigen Variablen im Modell, d. h. für beobachtete und latente Variablen, die durch andere Variablen erklärt werden.

- `modindices = TRUE` liefert Modifikationsindices: Für alle *nicht* im Modell enthaltenen Parameter gibt der Modifikationsindex eine Abschätzung, um wieviel sich der χ^2-Wert des Modells voraussichtlich verringert (also sich die Modellanpassung verbessert), wenn man diesen Parameter zusätzlich ins Modell aufnehmen würde. Dies liefert Hinweise auf mögliche Veränderungen (Modifikationen) an Modellen, die nicht gut zu den Daten passen. Als rein statistisches Kriterium sagen Modifikationsindices naturgemäss nichts darüber aus, ob der entsprechende Parameter dann auch sinnvoll interpretierbar und mit theoretischen Annahmen zu vereinbaren wäre. Ein explorativ modifiziertes Modell sollte idealerweise auch an einem neuen Datensatz kreuzvalidiert werden.

Standardeinstellung für diese vier Optionen ist `FALSE`, d. h. nur wenn man alle diese Optionen bei `summary()` erwähnt und auf `TRUE` setzt, enthält man die zusätzlichen Ergebnisse. Sinnvoll ist oft zumindest:

```
summary(ergebnis, fit.measures=TRUE, standardized=TRUE)
```

4 Einfaches lavaan-Beispielskript mit Kommentaren

```
# Beispiel-Syntax für die Analyse eines Strukturgleichungsmodells
# mit der freien Statistik-Umgebung R (http://www.r-project.org/)
# und dem lavaan-Paket für Strukturgleichungsmodelle (http://lavaan.org/)

# Mit # beginnende Zeilen enthalten Kommentare

# R-Paket "lavaan" für Strukturgleichungsmodelle installieren
# (nur einmalig; sobald das Paket vorhanden ist, die
# install.packages()-Anweisung mit # vor der Zeile auskommentieren)

install.packages("lavaan")

# lavaan-Paket für Strukturgleichungsmodelle laden (bei jeder Analyse)

library("lavaan")

# (1) Daten aus Datei einlesen

meine.daten <- read.csv("meine_datendatei.csv")

# Struktur der Daten anschauen: Anzahl Fälle, Anzahl und Namen der Variablen

str(meine.daten)

# (2) Modell definieren anhand der Variablen im Datensatz
# (hier als Beispiel mit generischen Namen):
# zwei unabhängige/exogene latente Variablen (Prädiktor-Konstrukte) xi1 und xi2,
# eine abhängige/endogene latente Variable eta1.
# Die gesamte Modelldefinition steht in einfachen Anführungszeichen ' '.
# Das Symbol =~ bedeutet "wird gemessen durch",
# das Symbol ~ bedeutet "wird vorhergesagt durch".

mein.modell <- '
 # Messmodelle fuer die drei Konstrukte mit 3, 3 und 2 Indikatorvariablen
  xi1  =~ x1 + x2 + x3
  xi2  =~ x4 + x5 + x6
  eta1 =~ y1 + y2

 # Strukturmodell zur Vorhersage von eta1 durch xi1 und xi2
  eta1 ~  xi1 + xi2
'

# (3) Modell analysieren mit der Funktion sem()

ergebnis <- sem(model=mein.modell, data=meine.daten, estimator="mlr")

# (4) Ergebnisse ausgeben einschliesslich
# Gütekriterien und standardisierter Parameterschätzungen

summary(ergebnis, fit.measures=TRUE, standardized=TRUE)
```

5 Darstellung der Ergebnisse in lavaan

Am Schluss der Analyse eines Strukturgleichungsmodells fordert man mit `summary()` die Ergebnisse an. Diese enthalten zuerst Angaben zum Ablauf der Analyse und zu Modellgütekriterien. Dann folgen die Parameter- und Standardfehlerschätzungen des Modells.

```
lavaan (0.5-17) converged normally after  39 iterations

  Number of observations                           212

  Estimator                                         ML      Robust
  Minimum Function Test Statistic               16.292      15.667
  Degrees of freedom                                17          17
  P-value (Chi-square)                           0.503       0.548
  Scaling correction factor                                  1.040
    for the Yuan-Bentler correction
```

Die «Minimum Function Test Statistic» ist die χ^2-Teststatistik des analysierten Modells. Hat man einen «robusten» Schätzer ausgewählt (wie im Beispiel mit `estimator="mlr"`), so werden für die Modellgütekriterien zwei Spalten ausgegeben.

- Die linke Spalte «`ML`» enthält die Ergebnisse für gewöhnliche Schätzung, hier also den unkorrigierten Maximum Likelihood-χ^2-Wert.
- Die rechte Spalte «`Robust`» nennt die Ergebnisse unter Berücksichtigung der Korrektur für nicht normalverteilte Daten.

Geht man davon aus, dass die eigenen Daten nicht multivariat normalverteilt sind, wären also die Werte in der Spalte «`Robust`» zu interpretieren. In diesem Beispiel ist der Yuan-Bentler-korrigierte χ^2-Wert rechts nur geringfügig kleiner aus als der unkorrigierte links. In jedem Fall gibt es hier keine signifikante Abweichung zwischen Modell und Daten (`P-value` > .05).

Im Fall von `fit.measures = TRUE` folgt analog die χ^2-Teststatistik für das Unabhängigkeitsmodell (`baseline model`), welche in die Gütekriterien Comparative Fit Index (CFI) und Tucker-Lewis Index eingeht (TLI, alternativ auch als Non-normed Fit Index NNFI bezeichnet).

```
Model test baseline model:

  Minimum Function Test Statistic             308.701     270.371
  Degrees of freedom                               28          28
  P-value                                       0.000       0.000
```

```
User model versus baseline model:

  Comparative Fit Index (CFI)                    1.000       1.000
  Tucker-Lewis Index (TLI)                       1.004       1.009
```

Wie in der Regel zu erwarten, passt das Unabhängigkeitsmodell nicht zu den Daten (sofern zwischen den Variablen im Modell irgendwo Zusammenhänge bestehen; `P-value` $< .05$). CFI und TLI zeigen an, um wieviel das analysierte Modell besser zu den Daten passt als das Unabhängigkeitsmodell (bei Werten von 0 würde das analysierte Modell *nicht* besser passen als das Unabhängigkeitsmodell, Werte um 1 sprechen für viel bessere Anpassung).

```
Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -2737.277   -2737.277
  Scaling correction factor                                  1.346
    for the MLR correction
  Loglikelihood unrestricted model (H1)      -2729.131   -2729.131
  Scaling correction factor                                  1.228
    for the MLR correction

  Number of free parameters                         27          27
  Akaike (AIC)                                5528.554    5528.554
  Bayesian (BIC)                              5619.182    5619.182
  Sample-size adjusted Bayesian (BIC)         5533.628    5533.628

Root Mean Square Error of Approximation:

  RMSEA                                          0.000       0.000
  90 Percent Confidence Interval          0.000  0.060       0.000  0.057
  P-value RMSEA <= 0.05                          0.890       0.913

Standardized Root Mean Square Residual:

  SRMR                                           0.031       0.031
```

Mit Gütekriterien CFI und TLI bei 1.0 und Root Mean Square Error of Approximation (RMSEA; Mass für Abweichungen zwischen Modell und Daten) bei 0 sind hier auch deskriptiv praktisch keine Abweichungen zwischen Modell und Daten erkennbar.

Es folgen die Parameter- und Standardfehlerschätzungen in mehreren Abschnitten:

- `Latent variables:` Faktorladungen in den Messmodellen der latenten Variablen
- `Regressions:` Effekte/Pfadkoeffizienten zwischen Variablen im Strukturmodell
- `Covariances:` Kovarianzen, z. B. zwischen latenten unabhängigen/exogenen Variablen
- `Intercepts:` Intercepts bzw. Mittelwerte von Indikator- und latenten Variablen

- `Variances:` Varianzen, nämlich:
  - Messfehlervarianzen von Indikatorvariablen (hier `x1` bis `y2`)
  - Gesamtvarianzen von unabhängigen (exogenen) Variablen (hier `xi1` und `xi2`)
  - Residualvarianzen (unerklärte Anteile) von abhängigen (endogenen) Variablen (hier `eta1`)

Die Parameter- und Standardfehlerschätzungen stehen dabei in mehreren Spalten:

- `Estimate`: In der ersten Spalte stehen unstandardisierte Parameterschätzungen.

In den drei folgenden Spalten erhält man für alle frei geschätzten Parameter (also z. B. *nicht* für die zur Skalierung auf 1 festgesetzten Faktorladungen):

- `Std.err`: Schätzung des Standardfehlers
- `Z-value`: z-Wert, d. h. unstandardisierte Schätzung dividiert durch den Standardfehler (erste durch zweite Spalte)
- `P(>|z|)`: p-Wert für den z-Test, dass der jeweilige Parameter ungleich null ist

Im Falle von `standardized = TRUE` folgen zwei Spalten mit standardisierten Parameterschätzungen:

- `Std.lv`: Parameter für den Fall, dass nur die latenten Variablen standardisiert sind (nur selten benötigt, z. B. im Fall, dass die manifesten Variablen kategorial und daher nicht sinnvoll standardisierbar sind)
- `Std.all`: Parameter auf der Basis, dass alle latenten und manifesten Variablen standardisiert sind

Im Falle von `rsquare = TRUE` bzw. `modindices = TRUE` folgen dann noch R^2-Werte für abhängige Variablen im Modell bzw. Modifikationsindices für nicht im Modell enthaltene Parameter (hier nicht dargestellt).

```
Parameter estimates:

  Information                                 Observed
  Standard Errors                    Robust.huber.white

                   Estimate  Std.err  Z-value  P(>|z|)   Std.lv  Std.all
Latent variables:
  xi1 =~
    x1                1.000                               0.985    0.706
    x2                0.779    0.165    4.735    0.000    0.767    0.631
    x3                0.667    0.149    4.479    0.000    0.657    0.549
  xi2 =~
    x4                1.000                               1.225    0.807
    x5                0.623    0.099    6.286    0.000    0.764    0.578
    x6                0.581    0.119    4.886    0.000    0.711    0.545
  eta1 =~
    y1                1.000                               1.163    0.865
    y2                0.658    0.202    3.252    0.001    0.765    0.575

Regressions:
  eta1 ~
    xi1               0.255    0.106    2.404    0.016    0.216    0.216
    xi2               0.342    0.105    3.241    0.001    0.360    0.360

Covariances:
  xi1 ~~
    xi2               0.354    0.134    2.651    0.008    0.293    0.293

Intercepts:
    x1               -0.117    0.096   -1.224    0.221   -0.117   -0.084
    x2               -0.092    0.083   -1.106    0.269   -0.092   -0.076
    x3               -0.077    0.082   -0.935    0.350   -0.077   -0.064
    x4                0.125    0.104    1.199    0.231    0.125    0.082
    x5                0.134    0.091    1.476    0.140    0.134    0.101
    x6                0.047    0.090    0.530    0.596    0.047    0.036
    y1                0.087    0.092    0.941    0.347    0.087    0.065
    y2               -0.028    0.091   -0.306    0.760   -0.028   -0.021
    xi1               0.000                               0.000    0.000
    xi2               0.000                               0.000    0.000
    eta1              0.000                               0.000    0.000

Variances:
    x1                0.974    0.218                      0.974    0.501
    x2                0.888    0.146                      0.888    0.601
    x3                1.000    0.164                      1.000    0.698
    x4                0.803    0.230                      0.803    0.349
    x5                1.165    0.163                      1.165    0.666
    x6                1.198    0.176                      1.198    0.703
    y1                0.454    0.376                      0.454    0.252
    y2                1.181    0.226                      1.181    0.669
    xi1               0.970    0.258                      1.000    1.000
    xi2               1.501    0.380                      1.000    1.000
    eta1              1.052    0.359                      0.778    0.778
```