Statistik-IV

Beispiele und Übungen für die Vorlesung ´Statistik IV‘ des Studiengangs Bachelor of Science in Psychologie an der Universität Bern.

Boris Mayer Stefan Thoma

23.05.24

Inhaltsverzeichnis

Einleitung

Dieses Dokument enthält Beispiele und Übungen für die Vorlesung "Statistik IV" im Rahmen des Studiengangs Bachelor of Science in Psychologie an der Universität Bern.

Die statistische Notation orientiert sich am Lehrbuch Statistik und Forschungsmethoden von Eid, Gollwitzer und Schmitt (2017, 5. Aufl.).

Voraussetzungen

Installieren Sie bitte die neueste `R` Version: R version 4.3.2 (2023-10-31) und RStudio: RStudio Desktop.

Typografische Konventionen

Im Text werden wir die folgenden farbigen Textblöcke verwenden:

Hinweis

Für Erklärungen und Hinweise

Vertiefung

Für weiterführende Informationen (nicht prüfungsrelevant)

Lösung

Die Lösungen der Aufgaben werden so dargestellt.

Wichtig

Zur Hervorhebung wichtiger Informationen.

Zusätzlich zur Fliesstext enthält dieses Skript `R` Code. Code-Blöcke sehen so aus:

```r
x <- seq(from = 1, to = 10, by = 1)
```

Alles, was sich in einem Code-Chunk befindet, kann in die `R` Konsole eingefügt werden. Wenn Sie mit dem Mauszeiger über das "Copy to clipboard"-Zeichen in der oberen rechten Ecke eines Code-Chunks gehen, können Sie den Text in die Zwischenablage kopieren.

Code-Chunks können auch einen Output haben:

```r
x
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

In diesem Block ist `x` der Input und `[1] 1 2 3 4 5 6 7 8 9 10` der Output. In diesem Beispiel haben wir die Variable `x` ausgeführt, d.h. deren Inhalt ausgeben/drucken lassen. Dasselbe hätten wir auch mit `print(x)` erhalten. Im davorigen Code-Chunk haben wir diese Variable erstellt und ihr die Zahlen von 1 bis 10 zugewiesen.

Pacman: Ein Package für das Managen von Packages

Zuerst installieren wir `pacman`: Ein Package, um weitere Packages einfacher zu installieren und zu laden. Danach installieren und laden wir alle weiteren benötigten Packages mit `pacman::p_load()`.

```r
# Installiert das Package "pacman"
install.packages("pacman")

# Jetzt ist pacman installiert und kann weitere Packages installieren und laden
# Die folgenden benötigen wir gleich im nächsten Kapitel.
pacman::p_load(lme4, nlme, tidyverse, lmerTest, gridExtra, ggplot2)
```

Lizenz

Teil I.

Hierarchisch lineare Modelle

1. Modelle mit Level-1-Prädiktoren

In diesem Kapitel behandeln wir Hierarchische Lineare Modelle mit Level-1-Prädiktoren. Aufbau und Systematik der Darstellung orientieren sich am Lehrbuch "Statistik und Forschungsmethoden" von Eid, Gollwitzer, und Schmitt (2017).

Es handelt sich um Modelle mit einer 2-Ebenen-Struktur (Individuen genestet in Gruppen), und zunächst werden nur Modelle ohne jede Prädiktorvariable (Intercept-Only-Modell) sowie Modelle mit ausschliesslich Prädiktorvariablen auf Level 1 (Random-Intercept-Modell und Random-Coefficients-Modell) behandelt.

Laden der in diesem Kapitel benötigten Packages:

```
pacman::p_load(lme4, nlme, tidyverse, lmerTest, gridExtra, ggplot2)
```

1.1. HLM Packages

Zu Beginn ein kurzer Überblick über die beiden Packages, die wir in diesem Kurs für Hierarchische Lineare Modelle benutzen werden.

nlme

Dies ist das ursprüngliche HLM- oder Mixed-Model-Package. Es hat mehr Optionen, insbesondere im Hinblick auf die Struktur der Level-1-Fehler bei Daten mit wiederholten Messungen. Wir verwenden `nlme` im Kapitel zu HLM-Modellen mit Messwiederholung. Die Syntax-Struktur wird dort behandelt.

lme4

Dieses modernere HLM-Package ist sehr gut geeignet für "normale" Hierarchische Lineare Modelle mit genesteten Gruppen. Wir verwenden es im Folgenden für die Analysen der Modelle mit Level-1- und Level-2-Prädiktoren.

Die "lme4"-Syntax baut auf der model syntax auf, die wir schon von `lm()` kennen. Also:

```
lm(formula = AbhängigeVariable ~ UnabhängigeVariable, data = dataframe)
```

Bei der Funktion `lmer()` (aus dem Package `lme4`) kommt die Spezifizierung der Gruppenvariable hinzu, und welche zufälligen Effekte geschätzt werden sollen. Die Syntax in der Klammer unterscheidet die `lmer()`-Syntax von der simplen Syntax der `lm()`-Funktion. In der Klammer wird links vom `|` spezifiziert, welche zufälligen Effekte geschätzt werden sollen. Rechts des `|` steht die Variable, die die Gruppenstruktur (Nestungsstruktur) der Daten definiert.

Eine `1` im linken Teil der Klammer bedeutet, dass random intercepts geschätzt werden sollen. Wenn die Unabhängige Variable (also die Prädiktorvariable) auch links vor dem `|` steht, bedeutet dies, dass zusätzlich random slopes geschätzt werden.

Was bedeutet also folgende Syntax?

```
lmer(data = dataframe, AbhängigeVariable ~ UnabhängigeVariable + (1 | Gruppenvariable))
```

Antwort: die Abhängige Variable wird von der Unabhängigen Variable vorhergesagt. Gleichzeitig wird der Intercept separat für jede Ausprägung der Gruppenvariable geschätzt bzw. die Varianz der Level-2-Residuen des Intercepts ist ein Parameter des Modells.

Diese Syntax wird später noch erweitert.

1.2. Daten laden und aufbereiten

Wir laden die Daten und speichern sie im Objekt `df` (für "data frame"). Wir können das Datenfile direkt aus dem Internet (GitHub) herunterladen:

```
df <- read_csv(
  url("https://raw.githubusercontent.com/methodenlehre/data/master/statistik_IV/salary-dat
)
```

Beschreibung

Bei diesen Daten handelt es sich um Lohn-Daten von 20 Firmen mit je 30 Angestellten (fiktive, simulierte Daten). Zu jedem Angestellten haben wir Informationen zum Gehalt (`salary`, in 1000 CHF), bei welcher Firma die Person angestellt ist (`company`), und wie lange die Person schon dort arbeitet (`experience`). Ausserdem haben wir noch Informationen darüber, in welchem Sektor (public oder privat) die Firma tätig ist (`sector`). Mit dieser Variable beschäftigen wir uns aber erst in Übung 2.

Bevor wir uns die Daten anschauen, müssen `company` und `sector` noch als Faktoren definiert werden.

```r
df <- df |>
  mutate(
    company = as.factor(company),
    sector = as.factor(sector)
  )
```

Jetzt können wir uns den Daten-Frame anschauen:

```r
# Die Funktion "tail()" gibt uns die letzten 6 rows aus.
tail(df)
```

```
# A tibble: 6 x 4
  company    experience salary sector
  <fct>           <dbl>  <dbl> <fct>
1 Company 20       3.58   6.84 Private
2 Company 20       3.18   7.60 Private
3 Company 20       3.39   5.71 Private
4 Company 20       7.12  10.1  Private
5 Company 20       2.98   6.94 Private
6 Company 20       6.45   9.33 Private
```

```r
# Die "summary()" Funktion gibt uns deskriptive Statistiken
# zum Datensatz aus.
summary(df)
```

```
      company      experience         salary          sector
 Company 01: 30   Min.   : 0.000   Min.   : 4.587   Private:300
 Company 02: 30   1st Qu.: 4.027   1st Qu.: 7.602   Public :300
 Company 03: 30   Median : 5.170   Median : 8.564
 Company 04: 30   Mean   : 5.190   Mean   : 8.738
 Company 05: 30   3rd Qu.: 6.402   3rd Qu.: 9.840
 Company 06: 30   Max.   :10.000   Max.   :15.418
 (Other)   :420
```

1.3. Intercept-Only-Modell (Modell 1)

In diesem Modell ist keine Prädiktorvariable enthalten, es handelt sich also um eine Art Null-modell, das nur die Variation des Achsenabschnitts zwischen den Gruppen modelliert. Und in

einem Modell ohne Prädiktor beziehen sich die Gruppenabschnitte auf die Gruppenmittelwerte (in diesem Fall der 20 Firmen).

Level-1-Modell: $y_{mi} = \beta_{0i} + \varepsilon_{mi}$

Level-2-Modell: $\beta_{0i} = \gamma_{00} + \upsilon_{0i}$

Gesamtmodell: $y_{mi} = \gamma_{00} + \upsilon_{0i} + \varepsilon_{mi}$

1.3.1. Modell berechnen

```
# Fitten des Modells und Abspeichern in der Variable intercept.only.model
intercept.only.model <- lmer(salary ~ 1 + (1 | company), data = df, REML = TRUE)

# Abspeichern von Prädiktionen, die dieses Modell macht.
# In diesem Falle ist dies jeweils der durchschnittliche Lohn jeder Firma
# Die Funktion predict() gibt pro Zeile (in diesem Fall pro Angestelltem)
# eine Prädiktion aus, je nachdem, in welcher Firma die Person angestellt ist.
# Wir speichern diese vorhergesagten Werte, um das Modell später mit "ggplot2"
# zu veranschaulichen.
df$intercept.only.preds <- predict(intercept.only.model)

# Model output anschauen
summary(intercept.only.model)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: salary ~ 1 + (1 | company)
   Data: df

REML criterion at convergence: 2158

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.9816 -0.6506 -0.0494  0.5779  4.2131

Random effects:
 Groups   Name        Variance Std.Dev.
 company  (Intercept) 0.8512   0.9226
 Residual             1.9547   1.3981
Number of obs: 600, groups:  company, 20
```

```
Fixed effects:
            Estimate Std. Error      df t value Pr(>|t|)
(Intercept)   8.7376     0.2141 19.0000   40.82   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Mit der Funktion `ranef()` kann man sich die einzelnen Random-Effects (Level-2-Residuen des Intercepts, also die v_{0i}) ausgeben lassen:

```
ranef(intercept.only.model)
```

```
$company
           (Intercept)
Company 01  0.78965416
Company 02  1.10501807
Company 03  1.92302618
Company 04 -1.13650080
Company 05  0.95354595
Company 06 -0.95893185
Company 07 -0.01014627
Company 08 -0.25484048
Company 09 -0.65131802
Company 10  0.76848590
Company 11 -0.50626620
Company 12  0.94040709
Company 13 -0.74284383
Company 14 -0.97545482
Company 15 -1.16136931
Company 16 -0.09768008
Company 17  0.66196052
Company 18 -0.16811195
Company 19  0.35123926
Company 20 -0.82987351

with conditional variances for "company"
```

1.3.2. Intraklassenkorrelation

Die Intraklassenkorrelation quantifiziert das Ausmass der „Nicht-Unabhängigkeit" zwischen den Messwerten aufgrund systematischer Level-2-Unterschiede im gemessenen Merkmal. Je grösser der Anteil der Level-2-Varianz (Varianz der Gruppenmittelwerte) an der Gesamtvarianz

(Summe der Level-2-Varianz und der Level-1-Varianz), desto grösser sind die Ähnlichkeiten *innerhalb* der Level-2-Einheiten im Vergleich zu *zwischen* den Level-2-Einheiten.

Die Intraklassenkorrelation ist definiert als: $\rho = \frac{\sigma^2_{Level-2}}{\sigma^2_{Level-2}+\sigma^2_{Level-1}}$

Die Intraklassenkorrelation erhält man bei Schätzung eines Nullmodells (Intercept-Only-Modell, s.o.), bei dem sowohl die (zufällige) Varianz des Intercepts (in einem Modell ohne Prädiktoren also die Varianz der Mittelwerte der Level-2-Einheiten) als auch die Level-1-Residualvarianz ausgegeben wird.

Wenn die Level-2-Varianz sich nicht überzufällig von 0 unterscheidet, spielen die Ähnlichkeiten/Abhängigkeiten innerhalb der Level-2-Einheiten keine Rolle und ein Mehrebenenmodell ist nicht unbedingt notwendig.

Intraklassenkorrelation: $\hat{\rho} = \frac{\hat{\sigma}^2_{v_0}}{\hat{\sigma}^2_{v_0}+\hat{\sigma}^2_{\varepsilon}} = \frac{0.8512}{0.8512+1.9547} = 0.303$

-> ca. 30 % der Gesamtvarianz sind auf Level-2-(Firmen-)Unterschiede zurückzuführen.

1.3.3. Signifikanztest

Signifikanztest für $\sigma^2_{v_0}$: Modellvergleich mit absolutem Nullmodell

Wir kennen nun also die Level-2-Varianz, aber wir haben noch keinen Signifikanztest für diesen Parameter. Diesen erhalten wir über einen Modellvergleich (Likelihood-Ratio-Test) des Intercept-Only-Modells mit einem Modell, dass keinen Random-Intercept enthält, d.h. mit einem "normalen" linearen Modell ohne Prädiktor (mit dem Gesamt-Mittelwert γ_{00} als einzigem Modellparameter neben der Level-1-Residualvarianz σ^2_{ε}). Dieses Modell kann man auch als "absolutes Nullmodell" bezeichnen. Wir müssen ein solches Modell nicht extra spezifizieren, sondern können die Funktion `ranova()` auf das Outputobjekt `intercept.only.model` anwenden. `ranova()` (aus dem lme4-Hilfspackage `lmerTest`) führt automatisch Modellvergleiche (nur) für Random-Effects durch, indem die vorhandenen Random-Effects Schritt für Schritt entfernt werden und das Ausgangsmodell dann mit dem so reduzierten Modell verglichen wird. In diesem Fall (Intercept-Only-Modell als Ausgangsmodell) kann nur *ein* Random-Effect entfernt werden, nämlich der des Intercepts.

```
# Vergleich des Intercept-Only-Modells mit dem "absoluten Nullmodell"
ranova(intercept.only.model)
```

```
ANOVA-like table for random-effects: Single term deletions

Model:
salary ~ (1 | company)
             npar  logLik    AIC    LRT Df Pr(>Chisq)
```

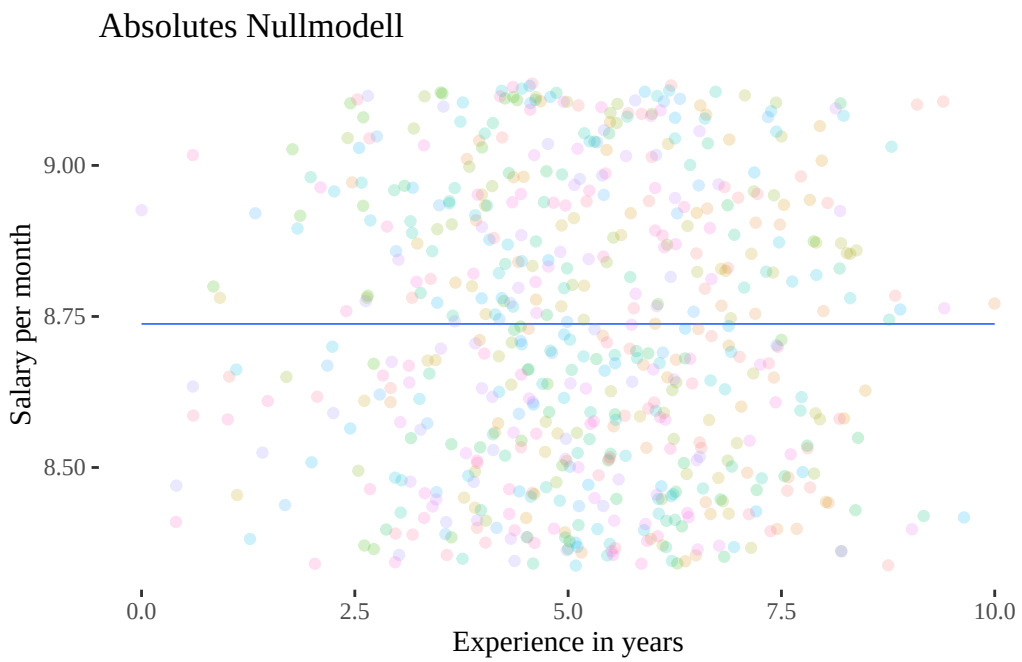
```
<none>           3 -1079.0 2164.0
(1 | company)    2 -1157.7 2319.4 157.44  1  < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Der Vergleich ist signifikant ($p < 0.001$). Wir können den p-Wert sogar noch halbieren, da es sich um einen einseitigen Test handelt. Der zu testende Parameter $\sigma^2_{v_0}$ weist nämlich einen *bounded parameter space* auf (kann nicht kleiner als 0 werden). Damit haben wir eine signifikante Level-2-Varianz des Intercepts und die Verwendung eines hierarchischen linearen Modells für die folgenden Analysen ist damit angezeigt.

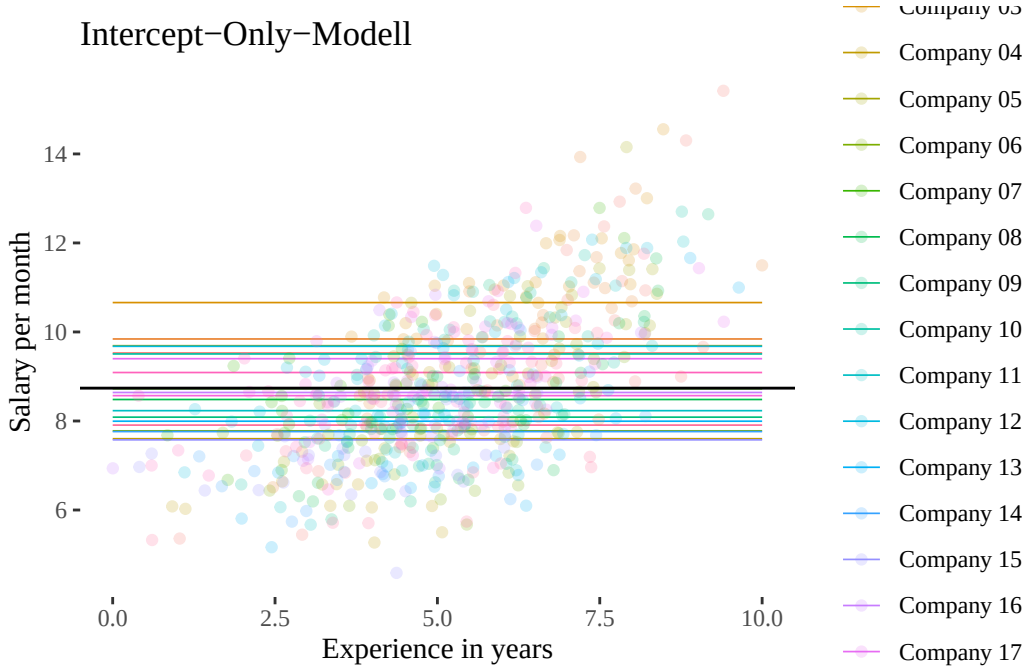
Weiter unten werden wir zeigen, wie Modellvergleiche mittels Likelihood-Ratio-Test auch von Hand durchgeführt werden können.

Visuell kann man sich den Vergleich zwischen dem absoluten Nullmodell und dem Intercept-Only-Modell folgendermassen vorstellen:

```r
xlabel <- "Experience in years"
ylabel <- "Salary per month"


# Visual comparison between the two models
null.model <- lm(salary ~ 1, data = df)
df$null.model.preds <- predict(null.model)

ggplot(data = df, aes(x = experience, y = null.model.preds)) + # , group=groupVar, colour=
  geom_smooth(method = "lm", fullrange = TRUE, se = F, size = .3) +
  geom_jitter(aes(group = company, colour = company), show.legend = FALSE, alpha = .2) +
  labs(x = xlabel, y = ylabel) +
  ggtitle("Absolutes Nullmodell") +
  scale_colour_discrete() +
  ggthemes::theme_tufte()
```




```
df$intercept.only.preds <- predict(intercept.only.model)

ggplot(data = df, aes(
  x = experience,
  y = intercept.only.preds,
  colour = company
)) +
  geom_smooth(method = "lm", fullrange = TRUE, se = F, size = .3) +
  geom_jitter(aes(y = salary), alpha = .2) +
  labs(x = xlabel, y = ylabel) +
  ggtitle("Intercept-Only-Modell") +
  scale_colour_discrete() +
  geom_abline(intercept = fixef(intercept.only.model), slope = 0) +
  ggthemes::theme_tufte()
```



Vertiefung: Shrinkage

Wir haben oben bereits angemerkt, dass in einem Intercept-Only-Modell die Gruppen-Intercepts den `salary` Firmenmittelwerten "entsprechen". Die geschätzten Gruppenmittelwerte ($\hat{\gamma}_{00} + \upsilon_{0i}$) erhält man mit `coef()`. Vergleichen wir diese mit den empirischen Durchschnittsgehältern pro Firma:

```
est_means <- coef(intercept.only.model)$company[, 1]
emp_means <- df |>
  group_by(company) |>
  summarise(emp_means = mean(salary)) |>
  ungroup() |>
  dplyr::select(emp_means)

salary <- tibble(est_means, emp_means) |>
  mutate(diff_means = est_means - emp_means)

salary
```

```
# A tibble: 20 x 3
   est_means emp_means diff_means
       <dbl>     <dbl>      <dbl>
```

```
 1      9.53      9.59  -0.0604
 2      9.84      9.93  -0.0846
 3     10.7      10.8   -0.147
 4      7.60      7.51   0.0870
 5      9.69      9.76  -0.0730
 6      7.78      7.71   0.0734
 7      8.73      8.73   0.000777
 8      8.48      8.46   0.0195
 9      8.09      8.04   0.0499
10      9.51      9.56  -0.0588
11      8.23      8.19   0.0388
12      9.68      9.75  -0.0720
13      7.99      7.94   0.0569
14      7.76      7.69   0.0747
15      7.58      7.49   0.0889
16      8.64      8.63   0.00748
17      9.40      9.45  -0.0507
18      8.57      8.56   0.0129
19      9.09      9.12  -0.0269
20      7.91      7.84   0.0635
```

Man kann erkennen, dass die geschätzten Intercepts pro Firma nicht mit den tatsächlichen empirischen Gruppenmitteln übereinstimmen. In einem Mehrebenenmodell werden diese Schätzungen mit "Shrinkage" (Schrumpfung) berechnet, d.h. sie werden in Richtung des Gesamtmittelwertes "gezogen". Dies liegt daran, dass die Level-2-Residuen in diesem Modell als normalverteilt angenommen werden und die Schätzunsicherheit in dem Sinne berücksichtigt wird, dass die Firmenmittelwerte der AV `salary` durch Gewichtung ("Pooling") mit dem Gesamtmittelwert korrigiert werden, um zu den Estimates $\beta_{0i} = \hat{\gamma}_{00} + \upsilon_{0i}$ zu gelangen.
Dies ist auch der Grund, warum Mehrebenenmodelle manchmal als "Partial-Pooling-Modelle" bezeichnet werden (z.B. bei Gelman und Hill 2007), im Gegensatz zu "No-Pooling-Modellen" (lineare Modelle mit Gruppen als fixed effects ohne jegliche Annahmen bezüglich der Varianz auf Ebene 2).
Schauen wir uns zum Vergleich ein solches "No-Pooling-Modell" an:

```r
no.pooling <- lm(salary ~ company, data = df)
summary(no.pooling)
```

```
Call:
lm(formula = salary ~ company, data = df)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-4.2291 -0.8918 -0.0401  0.7795  5.8300

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        9.58770    0.25526  37.560  < 2e-16 ***
companyCompany 02  0.33950    0.36099   0.940 0.347369
companyCompany 03  1.22013    0.36099   3.380 0.000774 ***
companyCompany 04 -2.07359    0.36099  -5.744 1.49e-08 ***
companyCompany 05  0.17644    0.36099   0.489 0.625201
companyCompany 06 -1.88243    0.36099  -5.215 2.57e-07 ***
companyCompany 07 -0.86102    0.36099  -2.385 0.017393 *
companyCompany 08 -1.12444    0.36099  -3.115 0.001931 **
companyCompany 09 -1.55127    0.36099  -4.297 2.03e-05 ***
companyCompany 10 -0.02279    0.36099  -0.063 0.949687
companyCompany 11 -1.39512    0.36099  -3.865 0.000124 ***
companyCompany 12  0.16229    0.36099   0.450 0.653188
companyCompany 13 -1.64980    0.36099  -4.570 5.96e-06 ***
companyCompany 14 -1.90022    0.36099  -5.264 1.99e-07 ***
companyCompany 15 -2.10036    0.36099  -5.818 9.84e-09 ***
companyCompany 16 -0.95525    0.36099  -2.646 0.008362 **
companyCompany 17 -0.13747    0.36099  -0.381 0.703488
companyCompany 18 -1.03108    0.36099  -2.856 0.004441 **
companyCompany 19 -0.47197    0.36099  -1.307 0.191585
companyCompany 20 -1.74349    0.36099  -4.830 1.75e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.398 on 580 degrees of freedom
Multiple R-squared:  0.3154,    Adjusted R-squared:  0.293
F-statistic: 14.06 on 19 and 580 DF,  p-value: < 2.2e-16
```

Das Modell ohne Pooling entspricht einem ANOVA-Modell, und die Effekte vergleichen die Gehaltsmittelwerte der Firmen 02-20 mit dem von Firma 01 (d.h. Effekte von dummy-codierten Variablen mit Firma 01 als Referenzkategorie). Das R^2 dieses Modells mit festen Effekten *(fixed effects ANOVA)* ist etwas höher als die Intraklassen-Korrelation des Intercept-Only-Modells (0.3154 vs. 0.3034), was eben die im Mehrebenenmodell vorhandene Shrinkage widerspiegelt.
Um ein Modell mit Schätzungen für alle Unternehmensmittelwerte (statt der Dummy-Effekte) zu erhalten, können wir den Intercept des `lm()`-Modells weglassen:

```r
no.pooling2 <- lm(salary ~ company - 1, data = df)
summary(no.pooling2)
```

```
Call:
lm(formula = salary ~ company - 1, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-4.2291 -0.8918 -0.0401  0.7795  5.8300

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
companyCompany 01   9.5877     0.2553   37.56   <2e-16 ***
companyCompany 02   9.9272     0.2553   38.89   <2e-16 ***
companyCompany 03  10.8078     0.2553   42.34   <2e-16 ***
companyCompany 04   7.5141     0.2553   29.44   <2e-16 ***
companyCompany 05   9.7641     0.2553   38.25   <2e-16 ***
companyCompany 06   7.7053     0.2553   30.19   <2e-16 ***
companyCompany 07   8.7267     0.2553   34.19   <2e-16 ***
companyCompany 08   8.4633     0.2553   33.16   <2e-16 ***
companyCompany 09   8.0364     0.2553   31.48   <2e-16 ***
companyCompany 10   9.5649     0.2553   37.47   <2e-16 ***
companyCompany 11   8.1926     0.2553   32.09   <2e-16 ***
companyCompany 12   9.7500     0.2553   38.20   <2e-16 ***
companyCompany 13   7.9379     0.2553   31.10   <2e-16 ***
companyCompany 14   7.6875     0.2553   30.12   <2e-16 ***
companyCompany 15   7.4873     0.2553   29.33   <2e-16 ***
companyCompany 16   8.6324     0.2553   33.82   <2e-16 ***
companyCompany 17   9.4502     0.2553   37.02   <2e-16 ***
companyCompany 18   8.5566     0.2553   33.52   <2e-16 ***
companyCompany 19   9.1157     0.2553   35.71   <2e-16 ***
companyCompany 20   7.8442     0.2553   30.73   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.398 on 580 degrees of freedom
Multiple R-squared:  0.9761,    Adjusted R-squared:  0.9753
F-statistic:  1185 on 20 and 580 DF,  p-value: < 2.2e-16
```

Nebenbemerkung: Da das Modell keinen Intercept enthält, beinhaltet die F-Statistik des

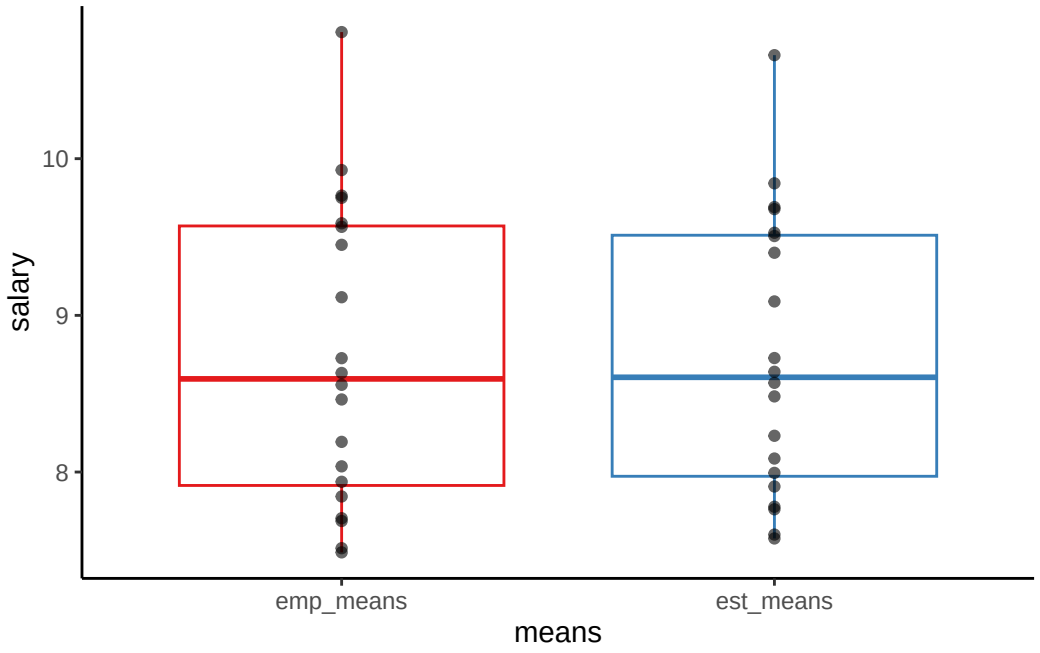
Modells nun auch den Test der Gruppenmittelwerte gegen 0, so dass wir sie nicht für den Vergleich der Gehaltsmittelwerte zwischen den Firmen verwenden können.
Überprüfen wir nun, ob die Schätzungen dieses Modells wirklich mit den Gruppenmittelwerten übereinstimmen:

```
# "no-pooling" Mittelwerte aus no.pooling2 extrahieren
np_means <- as.numeric(coef(no.pooling2))
# nicht benötigte Attribute entfernen
emp_means <- as.numeric(unlist(emp_means))
# Äquivalenz überprüfen
all.equal(np_means, emp_means)
```

```
[1] TRUE
```

Zum Schluss plotten wir noch die mit dem "partial pooling" Intercept-Only-Modell geschätzten Mittelwerte und die "no pooling" Mittelwerte des linearen Modells, also die empirischen Firmenmittelwerte:

```
salary |>
  pivot_longer(-diff_means, names_to = "means", values_to = "salary") |>
  mutate(means = as.factor(means)) |>
  ggplot(aes(x = means, y = salary)) +
  geom_boxplot(aes(color = means)) +
  geom_point(alpha = 0.6) +
  guides(color = "none") +
  scale_color_brewer(palette = "Set1") +
  theme_classic()
```



Wir sehen, dass die Verteilung der empirischen Mittelwerte etwas breiter ist als die der geschätzten ("geschrumpften") Mittelwerte. Im Falle ungleich grosser Unternehmensstichproben wäre die Shrinkage/Schrumpfung noch ausgeprägter, wobei die Mittelwerte von Firmen mit kleineren Stichproben stärker zum Gesamtmittelwert "gezogen" werden würden als die Mittelwerte von Firmen mit grossen Stichproben (was die geringere Genauigkeit der Schätzungen aus kleinen Stichproben widerspiegelt).

1.4. Random-Intercept-Modell (Modell 2)

Dieses Modell enthält einen Level-1-Prädiktor `(experience)`, der jedoch als fester Effekt konzeptualisiert wird (kein random slope, nur random intercept). Dieses Modell wird insbesondere benötigt, um den Anteil der durch den Level-1-Prädiktor erklärten Varianz zu ermitteln (vgl. Eid, Gollwitzer, und Schmitt 2017).

Level-1-Modell:
$y_{mi} = \beta_{0i} + \beta_{1i} \cdot x_{mi} + \varepsilon_{mi}$

Level-2-Modell:
$\beta_{0i} = \gamma_{00} + \upsilon_{0i}$
$\beta_{1i} = \gamma_{10}$

Gesamtmodell:
$y_{mi} = \gamma_{00} + \gamma_{10} \cdot x_{mi} + \upsilon_{0i} + \varepsilon_{mi}$

1.4.1. Modell berechnen

```r
# Fitten und Abspeichern des Modells.
# Wenn eine Prädiktorvariable im Modell ist,
# muss der Intercept (also 1) nicht zusätzlich angegeben werden.
random.intercept.model <- lmer(salary ~ experience + (1 | company),
  data = df, REML = TRUE
)

# Abspeichern von Prädiktionen, die dieses Modell macht.
# Wir speichern diese Prädiktionen (vorhergesagten Werte),
# um das Modell später mit "ggplot2" zu veranschaulichen.
df$random.intercept.preds <- predict(random.intercept.model)


# Modell output anschauen
summary(random.intercept.model)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: salary ~ experience + (1 | company)
   Data: df

REML criterion at convergence: 1865.4

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.8109 -0.6884  0.0005  0.5980  3.8833

Random effects:
 Groups   Name        Variance Std.Dev.
 company  (Intercept) 0.6144   0.7838
 Residual             1.1845   1.0883
Number of obs: 600, groups:  company, 20

Fixed effects:
             Estimate Std. Error        df t value Pr(>|t|)
(Intercept)   5.96418    0.22941  48.00060   26.00   <2e-16 ***
experience    0.53434    0.02721 589.48148   19.64   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Correlation of Fixed Effects:
           (Intr)
experience -0.615
```

Die Ergebnisse zeigen, dass der Fixed-Effect von `experience` positiv signifikant ist ($\hat{\gamma}_{10} = 0.534, p < 0.001$). Im Durchschnitt aller Firmen steigt das vorhergesagte Einkommen `(salary)` also mit jedem Jahr Arbeitserfahrung um ca. 0.534 Einheiten (534 CHF) an.

Mit der Funktion `ranef()` kann man sich wieder die einzelnen Random-Effects υ_{0i} ausgeben lassen:

```
ranef(random.intercept.model)
```

```
$company
           (Intercept)
Company 01  0.20430372
Company 02  0.64614732
Company 03  1.49200151
Company 04 -0.91078990
Company 05  0.38916512
Company 06 -0.92463977
Company 07  0.57766959
Company 08 -0.51651767
Company 09 -0.63824646
Company 10  0.76848113
Company 11 -0.61955111
Company 12  1.09133530
Company 13 -0.77367207
Company 14 -0.73817926
Company 15 -0.65294087
Company 16  0.05733923
Company 17  0.45805487
Company 18 -0.08938416
Company 19  0.94422822
Company 20 -0.76480474

with conditional variances for "company"
```

Die Werte scheinen im Vergleich zum Intercept-Only-Modell (s.o.) deutlich kleiner geworden zu sein. Was das bedeutet, sehen wir gleich bei der Berechnung der erklärten *Gesamtvarianz* durch den Prädiktor `experience`.

1.4.2. Varianzerklärung durch Level-1-Prädiktor

Wieviel Level-1-Varianz wurde durch `experience` erklärt?

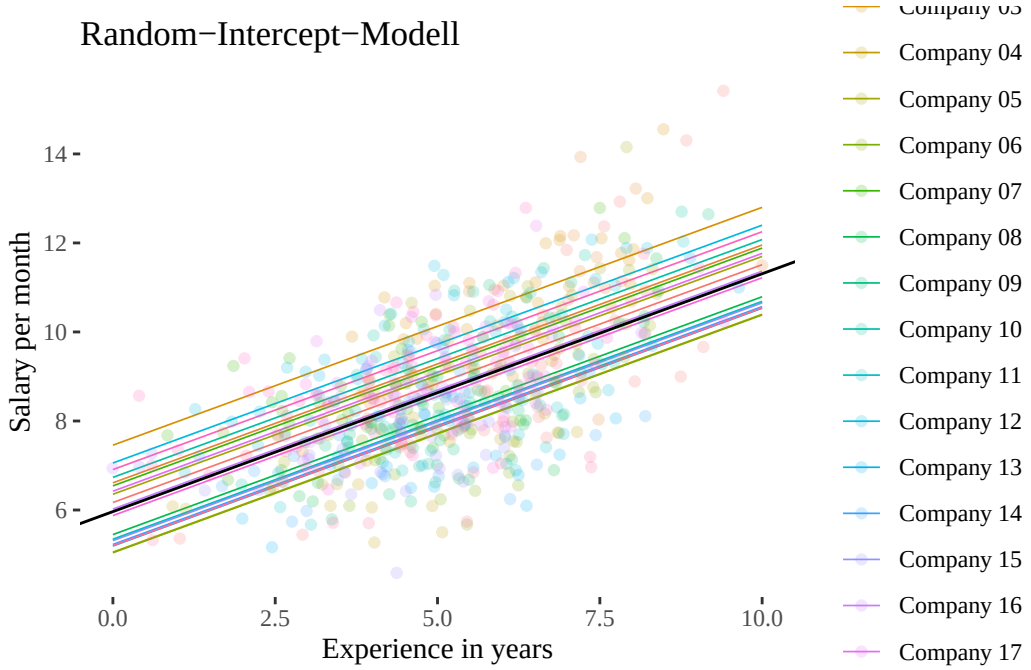
$R^2_{Level-1} = \frac{\sigma^2_{\varepsilon 1} - \sigma^2_{\varepsilon 2}}{\sigma^2_{\varepsilon 1}} = \frac{1.9547 - 1.1845}{1.9547} = 0.394$

Wieviel Gesamtvarianz wurde durch `experience` erklärt?

$R^2_{Gesamt} = \frac{(\sigma^2_{v_0 1} + \sigma^2_{\varepsilon 1}) - (\sigma^2_{v_0 2} + \sigma^2_{\varepsilon 2})}{\sigma^2_{v_0 1} + \sigma^2_{\varepsilon 1}} = \frac{(0.8512 + 1.9547) - (0.6144 + 1.1845)}{0.8512 + 1.9547} = 0.359$

Wir sehen, dass durch Aufnahme des Level-1 Prädiktors `experience` neben der Level-1-Varianz auch die Level-2-Varianz kleiner wurde. Das liegt daran, dass sich die Firmen nicht nur systematisch bezüglich der AV `salary` unterscheiden, sondern auch bezüglich des Prädiktors `experience`. Wie in einer Kovarianzanalyse verändert sich der Effekt (bzw. die Level-2-Varianz) von `company` durch Einbezug einer mit dem Faktor korrelierten Kovariate. Mehr dazu werden wir in der nächsten Übung erfahren.

```r
df$random.intercept.preds <- predict(random.intercept.model)

ggplot(data = df, aes(
  x = experience,
  y = random.intercept.preds,
  colour = company
)) +
  geom_smooth(method = "lm", fullrange = TRUE, se = F, size = .3) +
  geom_jitter(aes(y = salary), alpha = .2) +
  labs(x = xlabel, y = ylabel) +
  ggtitle("Random-Intercept-Modell") +
  geom_abline(
    intercept = fixef(random.intercept.model)["(Intercept)"],
    slope = fixef(random.intercept.model)["experience"]
  ) +
  scale_colour_discrete() +
  ggthemes::theme_tufte()
```



1.5. Random-Coefficients-Modell (Modell 3)

Dieses Modell enthält sowohl einen Random-Intercept, als auch einen Random-Slope. Mit Hilfe dieses Modells kann die Varianz des Level-1-Slopes geschätzt werden. Damit erhalten wir ein Mass für die Unterschiedlichkeit des Effekts (Slope) des Level-1-Prädiktors (X, `experience`) auf die AV (Y, `salary`) zwischen den Level-2-Einheiten `(company)`.

Level-1-Modell:
$y_{mi} = \beta_{0i} + \beta_{1i} \cdot x_{mi} + \varepsilon_{mi}$

Level-2-Modell:
$\beta_{0i} = \gamma_{00} + \upsilon_{0i}$
$\beta_{1i} = \gamma_{10} + \upsilon_{1i}$

Gesamtmodell:
$y_{mi} = \gamma_{00} + \gamma_{10} \cdot x_{mi} + \upsilon_{0i} + \upsilon_{1i} \cdot x_{mi} + \varepsilon_{mi}$

1.5.1. Modell berechnen

```
# Fitten und Abspeichern des Modells.
# Wenn für eine Prädiktorvariable ein Random-Slope im Modell ist, muss der
# Intercept (also 1) nicht mehr zusätzlich im Klammerausdruck für die zufälligen
# Effekte angegeben werden.
random.coefficients.model <- lmer(salary ~ experience + (experience | company),
  data = df, REML = TRUE
)

# Abspeichern von Prädiktionen, die dieses Modell macht.
# Wir speichern diese Prädiktionen, um das Modell später mit ggplot2
# zu veranschaulichen.
df$random.coefficients.preds <- predict(random.coefficients.model)


# Modell output anschauen
summary(random.coefficients.model)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: salary ~ experience + (experience | company)
   Data: df

REML criterion at convergence: 1855.4

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.8307 -0.6804  0.0037  0.5999  3.3608

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 company  (Intercept) 0.72257  0.8500
          experience  0.01839  0.1356   -0.51
 Residual             1.13629  1.0660
Number of obs: 600, groups:  company, 20

Fixed effects:
            Estimate Std. Error       df t value Pr(>|t|)
(Intercept)  5.93372    0.24038 18.88905   24.68 7.77e-16 ***
experience   0.53085    0.04059 18.95008   13.08 6.20e-11 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
           (Intr)
experience -0.690
```

Die Ergebnisse zeigen, dass der Fixed-Effect von `experience` wiederum positiv signifikant ist ($\hat{\gamma}_{10} = 0.531, p < 0.001$) (leicht unterschiedlich im Vergleich zum Random-Intercept-Modell). Das Hauptinteresse im Random-Coefficients-Modell gilt aber der Varianz bzw. Standardabweichung des Random-Slope: $\hat{\sigma}^2_{\upsilon_1} = 0.0184$ und damit $\hat{\sigma}_{\upsilon_1} = \sqrt{0.0184} = 0.1356$. Die durchschnittliche Abweichung (Level-2-Residuum υ_{1i}) des `experience`-Effekts einer Firma vom mittleren Effekt $\hat{\gamma}_{10}$ beträgt also 0.1356 Einheiten (135.6 CHF).

Mit der Funktion `ranef()` kann man sich wieder die einzelnen Random-Effects (jetzt Level-2-Residuen des Intercepts υ_{0i} und des Slopes υ_{1i}) ausgeben lassen:

```
ranef(random.coefficients.model)
```

```
$company
            (Intercept)   experience
Company 01 -1.003794102  0.203747776
Company 02 -0.160820541  0.143400958
Company 03  0.705132570  0.139886714
Company 04 -0.611829652 -0.054018876
Company 05 -0.033975063  0.076641659
Company 06 -0.123906149 -0.150704419
Company 07  0.669506844 -0.013616984
Company 08 -0.830781477  0.065813067
Company 09 -1.020757779  0.086008002
Company 10  0.392944052  0.081964065
Company 11 -0.769063062  0.037944670
Company 12  1.248886366 -0.025028992
Company 13 -0.593370830 -0.025453610
Company 14 -0.186191134 -0.109527150
Company 15  0.007222191 -0.150163600
Company 16  0.772829070 -0.140812590
Company 17  0.562032356 -0.011037598
Company 18  0.508160168 -0.112469594
Company 19  1.006168875 -0.006676886
Company 20 -0.538392700 -0.035896609

with conditional variances for "company"
```

Im Output befindet sich auch eine Angabe zum dritten Random-Effect-Parameter, der in diesem Modell geschätzt wurde, nämlich zur Kovarianz $\hat{\sigma}_{\upsilon_0\upsilon_1}$ zwischen den Level-2-Residen des Intercepts und den Level-2-Residuen des Slopes. Allerdings wird nicht die Kovarianz selbst ausgegeben, sondern ihre standardisierte Variante, die Korrelation $r_{\upsilon_0\upsilon_1} = -0.51$. Diese besagt, dass das durchschnittliche Gehalt eines Berufsanfängers in einer Firma (vorhergesagter Wert von `salary` an der Stelle `experience = 0`) negativ mit der Steigung von `experience` korreliert ist. Zur Interpretation: Wenn schon Berufsanfänger in einer Firma gut verdienen, gibt es offenbar weniger Spielraum für Gehaltssteigerungen als in Firmen mit weniger hohem Einstiegsgehalt.

1.5.2. Signifikanztest für $\sigma^2_{\upsilon_1}$: Modellvergleich mit Random-Intercept-Modell

Wenn wir dieses Modell mittels LR-Test mit dem Random-Intercept-Modell (s.o.) vergleichen, werden zwei Parameter gleichzeitig getestet: die Level-2-Varianz des Slopes ($\sigma^2_{\upsilon_1}$) und die Level-2-Kovarianz zwischen Intercept und Slope ($\sigma_{\upsilon_0\upsilon_1}$). Laut gegenwärtigem Forschungsstand ist dies die beste Methode, um zu überprüfen, ob es eine signifikante Slope-Varianz gibt (vgl. Eid, Gollwitzer, und Schmitt 2017).

1.5.2.1. Modellvergleich von Hand

Zunächst benötigen wir die -2-Log-Likelihoods (Devianzen) der beiden zu vergleichenden Modelle. Diese wird im Modelloutput des jeweiligen Modells unter der Bezeichnung "REML criterion at convergence" ausgegeben.

Wir folgen hier ausserdem den Bezeichnungen für uneingeschränkte Modelle (M_u) und für eingeschränkte (restringierte) Modelle (M_e), wie wir sie auch für die Modellvergleiche bei der logistischen Regression in Statistik III verwendet haben. M_u steht hier für das Random-Coefficients-Modell, das die zu testenden Parameter (Slope-Varianz $\sigma^2_{\upsilon_1}$ und Intercept-Slope-Kovarianz $\sigma_{\upsilon_0\upsilon_1}$) enthält, M_e für das Random-Intercept-Model, das genau diese Parameter nicht enthält.

-2-Log-Likelihood `random.intercept.model` (M_e): $-2 \cdot ln(L_e) = 1865.4$

-2-Log-Likelihood `random.coefficients.model` (M_u): $-2 \cdot ln(L_u) = 1855.4$

Berechnen des Modellvergleichstests $LR(M_u - M_e)$:

$\chi^2_{emp} = [-2 \cdot ln(L_e)] - [-2 \cdot ln(L_u)] = 1865.4 - 1855.4 = 10.0$

Anzahl Parameter $(nPar)M_u : \mathbf{6}(\gamma_{00}, \gamma_{10}, \sigma^2_\varepsilon, \sigma^2_{\upsilon_0}, \sigma^2_{\upsilon_1}, \sigma_{\upsilon_0\upsilon_1})$

Anzahl Parameter $(nPar)M_e : \mathbf{4}(\gamma_{00}, \gamma_{10}, \sigma^2_\varepsilon, \sigma^2_{\upsilon_0})$

$LR(M_u - M_e)$ ist χ^2-verteilt mit $df = nPar(M_u) - nPar(M_e) = 6 - 4 = 2$

Kritischer Wert: $\chi^2_{(0.95,df=2)} = 5.9915$

Aber: Es handelt sich genau genommen um eine Mischverteilung aus $\chi^2_{df=1}$ und $\chi^2_{df=2}$, da einer der beiden getesteten Parameter ($\sigma^2_{\upsilon_1}$) einen *bounded parameter space* aufweist (kann nicht kleiner als 0 werden). Daher ist der korrekte kritische Chi-Quadrat-Wert hier das 95 %-Quantil dieser Mischverteilung und liegt bei (gerundet) 5.14 (vgl. Eid, Gollwitzer, und Schmitt 2017).

Der Test ist signifikant, da $\chi^2_{emp}(= 10.0) > \chi^2_{krit}(= 5.14)$.

Beide Parameter zusammen unterscheiden sich also signifikant von 0. Der Fokus der Interpretation liegt hier aber auf dem Parameter der Slope-Varianz $\sigma^2_{\upsilon_1}$ (nur wenn dieser > 0 ist, kann auch die Intercept-Slope-Kovarianz $\sigma_{\upsilon_0\upsilon_1}$ überhaupt ungleich 0 sein).

Inhaltlich bedeutet dies, dass sich die Stärke des Effekts von `experience` auf `salary` signifikant zwischen den Firmen (`company`) unterscheidet. Die oben im Random-Coefficients-Modell geschätzte Slope-Varianz von `experience` ($\hat{\sigma}^2_{\upsilon_1} = 0.0184$) ist damit als signifikant zu bewerten.

1.5.2.2. Modellvergleich in R

Wie beim Signifikanztest der Random-Intercept-Varianz $\sigma^2_{\upsilon_0}$ im Intercept-Only-Modell können wir den Modellvergleich hier wieder mit `ranova()` vornehmen. Wie oben wird `ranova()` auf das uneingeschränkte Modell angewandt, die Angabe eines (eingeschränkten) Vergleichsmodells ist nicht notwendig.

```
ranova(random.coefficients.model)
```

```
ANOVA-like table for random-effects: Single term deletions

Model:
salary ~ experience + (experience | company)
                                     npar  logLik    AIC    LRT Df Pr(>Chisq)
<none>                                  6 -927.70 1867.4
experience in (experience | company)    4 -932.71 1873.4 10.011  2   0.006699

<none>
experience in (experience | company) **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Alternativ können wir den Modellvergleich auch mit der uns bereits bekannten Funktion `anova()` durchführen, indem wir die beiden zu vergleichenden Modelle explizit angeben:

```
anova(random.coefficients.model, random.intercept.model, refit = FALSE)
```

```
Data: df
Models:
random.intercept.model: salary ~ experience + (1 | company)
random.coefficients.model: salary ~ experience + (experience | company)
                          npar    AIC    BIC  logLik deviance  Chisq Df
random.intercept.model       4 1873.4 1891.0 -932.71   1865.4
random.coefficients.model    6 1867.4 1893.8 -927.70   1855.4 10.011  2
                          Pr(>Chisq)
random.intercept.model
random.coefficients.model   0.006699 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

> **Vertiefung**
>
> Bei der `anova()` der beiden mit `lmer()` geschätzten Modelle werden in der Default-Einstellung `(refit = TRUE)` beide Modelle vor Durchführung des LR-Tests automatisch nochmals mittels ML (statt REML) geschätzt (falls vorher REML verwendet wurde) und der LR-Test dann auf die ML-gefitteten Modellobjekte angewendet. Dies ist eine Vorsichtsmassnahme der Entwickler des Packages, um den häufigen Fehler zu vermeiden, REML-geschätzte Modelle, die sich (auch) in ihrer Fixed-Effects-Struktur voneinander unterscheiden, per LR-Test zu vergleichen. Da wir hier zwei Modelle vergleichen, die sich nur in ihrer Random-Effects-Struktur unterscheiden, wählen wir `refit = FALSE`, um den Vergleich der mit REML gefitteten Modelle zu erhalten. Beim Modellvergleich mit `ranova()` spielt das keine Rolle, da dort ein mit REML bzw. mit ML gefittetes Modell automatisch mit dem ensprechend gefitteten reduzierten Modell verglichen wird. Der Grund dafür ist, dass `ranova()` nur Modellvergleiche zum Test von *Random-Effects* durchführt und die Refit-Problematik daher dort nicht existiert.

Das Ergebnis zeigt einen signifikanten LR-Test mit $\chi^2_{emp} = 10.011$. Der (Rundungs-)Unterschied zum Modellvergleich von Hand ergibt sich wegen der ungenaueren Angabe der Devianzen in den Modelloutputs.

Hinweis

Man kann den empirischen Chi-Quadrat-Wert des Modellvergleichs auch mit den Angaben zu den Log-Likelihoods (logLik) der Modelle aus dem Output von `ranova()` bzw. `anova()` nachrechnen:
$\chi^2_{emp} = [-2 \cdot (-932.71)] - [-2 \cdot (-927.70)] = 1865.42 - 1855.40 = 10.02$
Hier ergibt sich dann eine Ungenauigkeit in die andere Richtung...

Der angegebene p-Wert ist allerdings noch nicht ganz korrekt, da einer der beiden Parameter - $\sigma^2_{v_1}$ - wie oben bereits festgestellt einen *bounded parameter space* aufweist. Der korrekte p-Wert auf Basis der Mischverteilung ist daher noch etwas kleiner als der ausgegebene.

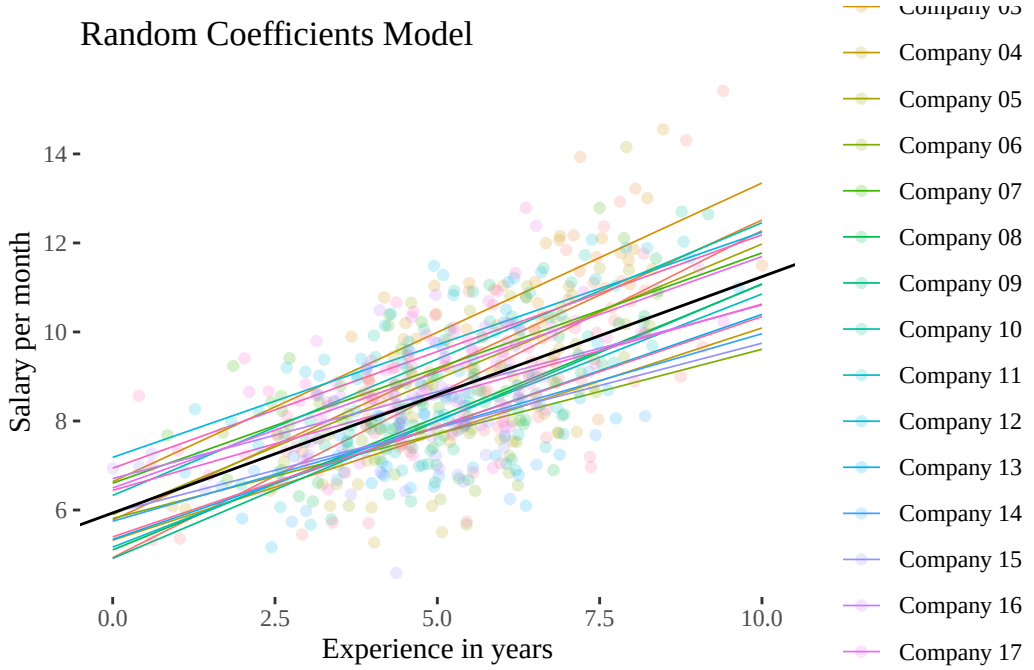
Vertiefung: p-Wert der Mischverteilung

Die exakte Bestimmung des p-Werts aufgrund der Mischverteilung erfolgt über die Mittelung der p-Werte der beiden Verteilungen $\chi^2_{df=1}$ und $\chi^2_{df=2}$:

```
0.5 * pchisq(10.011, 1, lower.tail = FALSE) +
  0.5 * pchisq(10.011, 2, lower.tail = FALSE)
```

```
[1] 0.004128535
```

```
# sal.visualisation$random.intercept.random.slope.plot
df$random.coefficients.preds <- predict(random.coefficients.model)

ggplot(data = df, aes(x = experience, y = random.coefficients.preds, colour = company)) +
  geom_smooth(method = "lm", fullrange = TRUE, se = F, size = .3) +
  labs(x = xlabel, y = ylabel) +
  geom_jitter(aes(y = salary), alpha = .2) +
  geom_abline(
    intercept = fixef(random.coefficients.model)["(Intercept)"],
    slope = fixef(random.coefficients.model)["experience"]
  ) +
  ggtitle("Random Coefficients Model") +
  scale_colour_discrete() +
  ggthemes::theme_tufte()
```



1.6. Übung

Wir möchten (fiktive) Daten zur Pisa-Studie analysieren. Dafür haben wir einen Datensatz mit den Variablen:

`id` : Durchnummerierung aller Schüler (1:1000)

`name` : Name der Schülerin / des Schülers

`grade` : Die durchschnittliche Vornote jedes Schülers (gerundet auf 0.25)

`motivation` : Wie motiviert die Schüler für den Pisa-Test sind

`class` : Variable der Schulklasse (durchnummeriert, 50 Klassen wurden zufällig ausgewählt)

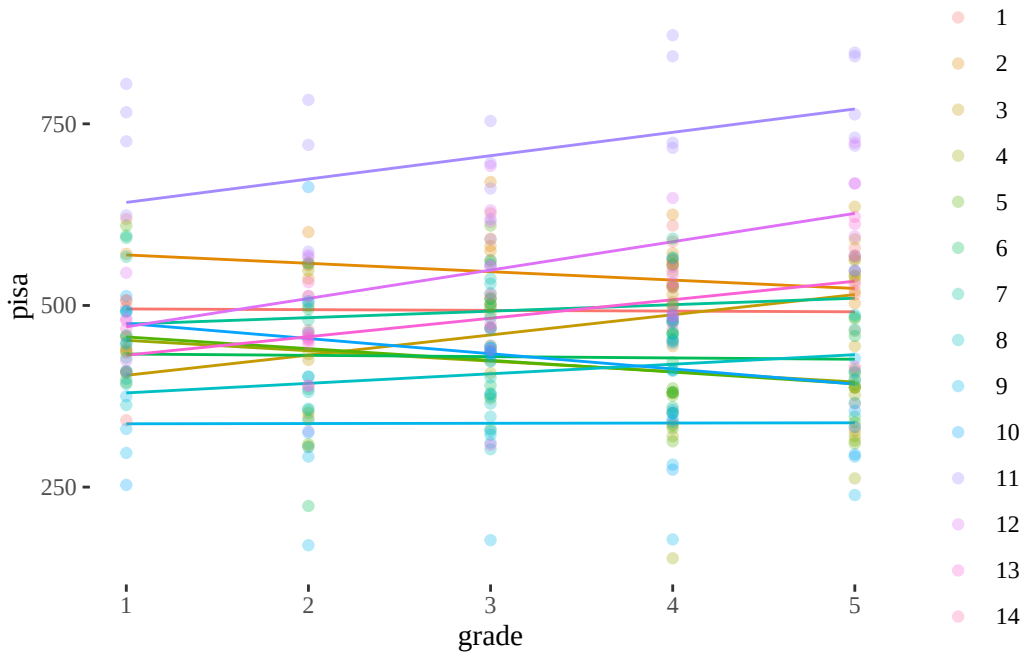
`teacher_salary` : Monatslohn des Lehrers (jede Klasse hat einen anderen Lehrer)

`school` : Kategoriale Variable der Schule (10 Schulen wurden zufällig ausgewählt)

`pisa` : Erreichte Punktzahl beim Pisa-Test.

Aus jeder der 10 Schulen wurden 5 Klassen ausgesucht. Insgesamt sind es also 50 Klassen. Für die folgenden Aufgaben beachten wir die Nestung der Klassen in den verschiedenen Schulen nicht, da wir dafür ein 3-Ebenen-Modell bräuchten. Unser Sample auf Level-2 sind also die 50 Schulklassen. Die Daten kann man sich etwa so vorstellen (subsample, nur 15 Klassen für die Visualisierung):

```r
school_df <- read_csv(url("https://raw.githubusercontent.com/methodenlehre/data/master/sta

subsample <- school_df[1:300, ]
subsample <- subsample |> mutate(class = as.factor(class))

ggplot(data = subsample, aes(x = grade, y = pisa, group = class, color = class)) +
  geom_smooth(method = "lm", alpha = .5, se = FALSE, show.legend = F, fullrange = TRUE, si
  geom_point(alpha = .3) +
  ggthemes::theme_tufte()
```



1. Aufbereitung der Daten

a) Laden Sie die Daten in R und schauen Sie sich die ersten paar Zeilen des Datensatzes an. Die Daten können entweder zuerst heruntergeladen und im Ordner versorgt werden, oder man kann sie auch direkt von R aus dem Internet laden (weil sie öffentlich im Internet zugänglich sind). Die Daten sind hier:

https://raw.githubusercontent.com/methodenlehre/data/master/statistik_IV/simulated_school_data.csv

Tipp: Verwenden Sie dieselbe Funktion wie weiter oben im Skript um Daten direkt aus dem Internet zu laden.

Lösung

```r
# 1a
pacman::p_load(lme4, nlme, tidyverse, lmerTest, gridExtra, ggplot2)
```

```r
school_df <- read_csv(url("https://raw.githubusercontent.com/methodenlehre/data/master/s
```

```
Rows: 1150 Columns: 8
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (2): name, school
dbl (6): id, motivation, grade, class, teacher_salary, pisa

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

```r
head(school_df)
```

```
# A tibble: 6 x 8
     id name   motivation grade class teacher_salary school   pisa
  <dbl> <chr>       <dbl> <dbl> <dbl>          <dbl> <chr>   <dbl>
1     1 Denis        4        2     1           4.74 Bitzius   513
2     2 Talia        5.25     4     1           4.74 Bitzius   563
3     3 Ahmed        6        4     1           4.74 Bitzius   610
4     4 Lamija       4.75     4     1           4.74 Bitzius   337
5     5 Titus        4.5      4     1           4.74 Bitzius   477
6     6 Justus       2        1     1           4.74 Bitzius   456
```

b) Stellen Sie sicher, dass die Variablen im richtigen Format sind. Als Tipp: Überlegen Sie sich, welche Variablen kategorial und welche numerisch sind.

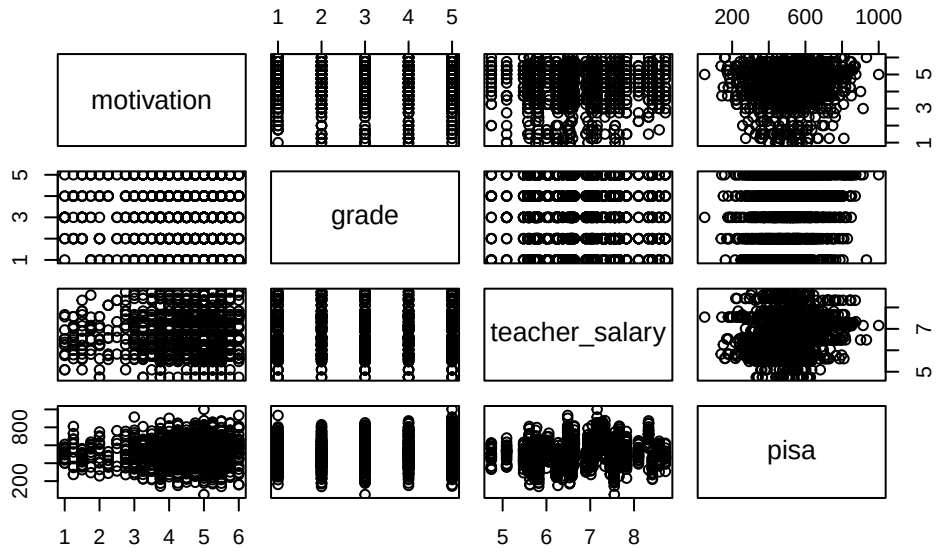
Lösung

```r
# 1b
school_df <- school_df |> mutate(
  id = as.factor(id),
  class = as.factor(class)
)
```

c) (optional) Visualisieren Sie die Daten mit der Funktion `plot()`. Als Tipp: Wenn nur numerische Variablen im Datenframe sind, können mit plot(datenframe) direkt die Zusammenhänge der Variablen dargestellt werden.

Lösung

```
# 1c
plot(school_df[, -c(1:2, 5, 7)])
```




```
# entfernt die kategorialen Columns (1, 2, 5 und 7) und plottet den Rest.
```

Man sieht hier keine offensichtlichen Korrelationen.

2. Fragestellung und Intercept-Only-Modell

Wir gehen davon aus, dass die Schulklassen zufällig ausgesucht wurden. Wir sind nicht daran interessiert, wie sich diese spezifischen Klassen voneinander unterscheiden. Uns interessiert vielmehr, ob die Vornote (`grade`) sowie die Motivation (`motivation`) einen Einfluss auf die Pisa-Werte haben. Die Schulklassen möchten wir als Random-Effect modellieren.

a) Berechnen Sie ein Intercept-Only-Modell.

Lösung

```r
# 2a
intOnly_fit <- lmer(
  formula = pisa ~ 1 + (1 | class),
  data = school_df
)
summary(intOnly_fit)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: pisa ~ 1 + (1 | class)
   Data: school_df

REML criterion at convergence: 13768.8

Scaled residuals:
    Min      1Q  Median      3Q     Max
-4.1991 -0.5967  0.0214  0.6522  2.9179

Random effects:
 Groups   Name        Variance Std.Dev.
 class    (Intercept) 12559    112.07
 Residual              7983     89.35
Number of obs: 1150, groups:  class, 50

Fixed effects:
            Estimate Std. Error     df t value Pr(>|t|)
(Intercept)   507.52      16.07  49.00   31.59   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

b) Testen Sie, ob der Random-Effect (`class`) in diesem Modell tatsächlich Varianz aufweist. Ist die Modellstruktur gerechtfertigt, d.h. benötigen wir überhaupt ein HLM-Modell?

Lösung

```r
# 2b
ranova(intOnly_fit)
```

```
ANOVA-like table for random-effects: Single term deletions

Model:
pisa ~ (1 | class)
            npar  logLik   AIC    LRT Df Pr(>Chisq)
<none>         3 -6884.4 13775
(1 | class)    2 -7332.0 14668 895.23  1  < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die Klassen unterscheiden sich signifikant im Pisa-Durchschnittswert. Man erkennt das daran, dass der p-Wert der Varianz für den random-effect kleiner als 0.05 ist. Wir können diesen p-Wert sogar noch halbieren (s.o.). Die Modellstruktur ist somit also gerechtfertigt.

c) Wie gross ist die Intraklassenkorrelation?

Lösung

Um die Intraklassenkorrelation zu berechnen, brauchen wir die Varianzen des `intOnly_fit` Objekts. Statt die Random-Intercept-Varianz und die Level-1-Residualvarianz aus dem Output abzulesen, kann man Sie auch aus dem Modell-Objekt extrahieren und abspeichern.

```r
# Varianz von intOnly_fit extrahieren
intOnly_class_var <- as.data.frame(VarCorr(intOnly_fit))$vcov

classVar <- intOnly_class_var[1] # das erste Element ist die Random-Intercept-Varianz
resVar <- intOnly_class_var[2] # das zweite Element ist die Level-1-Residualvarianz
totalVar <- classVar + resVar # die Summe aus beiden ist die Gesamtvarianz

intraclass_correlation <- (classVar / totalVar) |> round(3)

intraclass_correlation
```

```
[1] 0.611
```

Die Intraclass Correlation ist $\hat{\rho} = 0.611$. Klasse erklärt also ca 61.1% der gesamten Varianz der Pisa-Werte in diesem Modell.
Statt die Varianzen so wie oben aus dem Output-Objekt `intOnly_fit` zu indizieren, können sie auch aus dem Output (s.o. 2a) herausgelesen werden:

$$\hat{\rho} = \frac{\hat{\sigma}^2_{Level-2}}{\hat{\sigma}^2_{Level-2} + \hat{\sigma}^2_{Level-1}} = \frac{1.25587 \times 10^4}{1.25587 \times 10^4 + 7983.4} = 0.611$$

.

3. Random-Intercept-Modell

a) Berechnen Sie ein Random-Intercept-Modell mit den beiden Prädiktoren `grade` und `motivation`. Sind die fixed effects ($\hat{\gamma}_{10}$ und ($\hat{\gamma}_{20}$) beider Prädiktoren signifikant?

Lösung

```r
# 3a
randInt_fit <- lmer(
  formula = pisa ~ grade + motivation + (1 | class),
  data = school_df
)
summary(randInt_fit)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: pisa ~ grade + motivation + (1 | class)
   Data: school_df

REML criterion at convergence: 13699.4

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.9786 -0.6213  0.0302  0.6310  2.8518

Random effects:
 Groups   Name        Variance Std.Dev.
 class    (Intercept) 12536    111.96
 Residual              7555     86.92
Number of obs: 1150, groups:  class, 50

Fixed effects:
            Estimate Std. Error       df t value Pr(>|t|)
(Intercept)  440.336     20.073  118.333  21.937  < 2e-16 ***
grade         14.730      1.882 1100.532   7.826 1.18e-14 ***
motivation     4.890      2.355 1100.484   2.077   0.0381 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Correlation of Fixed Effects:
           (Intr) grade
grade      -0.309
motivation -0.526  0.036
```

Beide Prädiktoren sind signifikant. An den Estimates kann man jedoch nicht direkt ablesen, wessen Einfluss auf den Pisa-Wert stärker ist, weil diese Werte von der Skalierung der Prädiktoren abhängen.

b) Testen Sie, ob ein zusätzlicher Interaktionseffekt zwischen `grade` und `motivation` ins Modell aufgenommen werden sollte. Obwohl wir keine spezifische Moderationshypothese haben, können wir im Sinne eines Vermeidens von Underfitting überprüfen, ob eine bedeutsame Interaktion vorhanden ist.

Lösung

```r
# Wir können entweder das gesamte Modell neu definieren, neu auch mit
# grade:motivation, oder wir nutzen die Funktion update() und fügen
# einfach den Prädiktor grade:motivation hinzu.
# Beides führt zum selben Resultat.

randInt_interaction_fit <- lmer(
  formula = pisa ~ grade * motivation + (1 | class),
  data = school_df
)
summary(randInt_interaction_fit)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: pisa ~ grade * motivation + (1 | class)
   Data: school_df

REML criterion at convergence: 13694.8

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.8938 -0.6279  0.0308  0.6441  2.8305

Random effects:
 Groups   Name        Variance Std.Dev.
 class    (Intercept) 12507    111.8
```

```
 Residual              7552     86.9
Number of obs: 1150, groups:  class, 50

Fixed effects:
                 Estimate Std. Error       df t value Pr(>|t|)
(Intercept)       408.368     31.913  552.133  12.796  < 2e-16 ***
grade              24.842      8.074 1098.751   3.077  0.00215 **
motivation         12.009      6.009 1098.485   1.999  0.04589 *
grade:motivation   -2.261      1.755 1098.712  -1.288  0.19808
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) grade  motvtn
grade       -0.802
motivation  -0.845  0.898
grade:mtvtn  0.778 -0.972 -0.920
```

Ein Interaktionseffekt scheint in diesem Modell keinen Sinn zu machen, der p-Wert für den Interaktionseffekt ist nicht signifikant ($p_{grade:motivation} = 0.1981$), potentielle moderierende Einflüsse zwischen den beiden Prädiktoren sind also nicht auszumachen.

4. Random-Coefficients-Modell

Zusätzlich zum Random-Intercept können wir auch die Varianzen der Slopes von `grade` und `motivation` mit ins Modell aufnehmen. Doch "lohnt" sich das überhaupt, d.h. unterscheiden sich die Effekte der beiden Prädiktoren jeweils signifikant zwischen den Klassen?

a) Schätzen Sie ein Random-Coefficients Modell mit Random Slopes für `grade` und `motivation`.

Lösung

```r
randCoef_fit <- lmer(formula = pisa ~ grade + motivation +
  (grade + motivation | class), data = school_df)

summary(randCoef_fit)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
```

```
Formula: pisa ~ grade + motivation + (grade + motivation | class)
   Data: school_df

REML criterion at convergence: 13599.5

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.2014 -0.6149  0.0275  0.6347  2.7361

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 class    (Intercept) 9519.1   97.57
          grade        275.3   16.59    -0.10
          motivation   513.2   22.65    -0.56  0.23
 Residual             6382.5   79.89
Number of obs: 1150, groups:  class, 50

Fixed effects:
            Estimate Std. Error      df t value Pr(>|t|)
(Intercept)  430.401     18.178  46.240  23.677  < 2e-16 ***
grade         14.977      2.935  47.140   5.103 5.89e-06 ***
motivation     6.970      3.921  48.741   1.777   0.0817 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
           (Intr) grade
grade      -0.255
motivation -0.670  0.167
```

Im Random-Coefficients-Modell ist der Fixed Effect von `motivation` ($\hat{\gamma}_{20}$) nun nicht mehr signifikant ($p_{motivation} = 0.0817$). Das ist nicht ungewöhnlich, die Schätzung von Random Effects Parametern beeinflusst auch diejenigen der Fixed Effects, und derjenige von `motivation` war im Random-Intercept-Modell nur knapp signifikant.

b) Testen Sie die Signifikanz beider Random Slopes mit `ranova()`.

Lösung

Die `ranova()`-Funktion testet die Signifikanz von Level-2-Varianzparametern mithilfe von Likelihood-Ratio-Tests. In unserem Beispiel wird das `randCoef_fit` Modell mit Modellen

verglichen, die jeweils eine Slope-Varianz (und deren Kovarianzen mit dem Intercept und dem jeweils anderen Slope) weniger im Modell haben. Ein signifikanter Test bedeutet hier, dass die Daten schlechter auf die eingeschränkten Vergleichsmodelle passen als auf das uneingeschränkte Modell `randCoef_fit`.
Die p-Werte des Likelihood-Ratio-Tests müssten noch angepasst werden, weil hier zwei "unbounded" (die Kovarianz des Slopes des jeweils getesteten Prädiktors mit dem random Intercept und die Kovarianz dieses Slopes mit dem Slope des jeweils anderen Prädiktors) und ein "bounded" Parameter (die jeweilige Slope-Varianz) gleichzeitig getestet werden. Darauf gehen wir jedoch hier nicht weiter ein.

```
ranova(randCoef_fit)
```

```
ANOVA-like table for random-effects: Single term deletions

Model:
pisa ~ grade + motivation + (grade + motivation | class)
                                              npar  logLik   AIC    LRT Df
<none>                                          10 -6799.8 13620
grade in (grade + motivation | class)            7 -6822.0 13658 44.492  3
motivation in (grade + motivation | class)       7 -6828.7 13671 57.895  3
                                              Pr(>Chisq)
<none>
grade in (grade + motivation | class)        1.186e-09 ***
motivation in (grade + motivation | class)   1.655e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Beide Slope-Varianzen sind signifikant: Sowohl das Weglassen von `grade` in `(grade + motivation | class)` als auch das Weglassen von `motivation` in `(grade + motivation | class)` würde zu einem signifikant schlechteren Fit führen (signifikanter Likelihood-Ratio-Test LRT). Die Klassen unterscheiden sich also signifikant bezüglich der Effekte von `motivation` und `grade`. Ein Random-Coefficients-Modell ist daher notwendig und erklärt die Daten besser als das Random-Intercept-Modell.
Der durchschnittliche (fixed) Effekt von `motivation` über alle Klassen hinweg ist zwar nicht signifikant von 0 unterschieden (s.o.), trotzdem ist der Einfluss der `motivation` auf die `pisa`-Werte aber zwischen den Klassen verschieden. Möglicherweise gibt es Merkmale auf Klassenebene (wie z.B. Klassengrösse, um ein Beispiel zu nennen), die erklären könnten, in welchen Klassen `motivation` einen Effekt auf die `pisa`-Werte hat und in welchen nicht. Darauf gehen wir aber nicht weiter ein, auch weil wir (ausser `teacher_salary`) keine Merkmale auf Klassenebene im Datensatz haben.

Vertiefung

c) Basierend auf dem besten Modell: Was ist der vorhergesagte Wert eines Schülers mit…

- grade = 4.5, motivation = 1 (aus einer unbekannten Klasse)
- grade = 4, motivation = 6, class = 18

Lösung

```
# 1
(schüler1 <- fixef(randCoef_fit)[1] + 4.5 * fixef(randCoef_fit)[2] +
  1 * fixef(randCoef_fit)[3])
```

```
(Intercept)
   504.7657
```

```
# Überprüfen mit predict()
predict(randCoef_fit, newdata = tibble(grade = 4.5, motivation = 1), re.form = NA)
```

```
       1
504.7657
```

```
# 2
(schüler2 <- fixef(randCoef_fit)[1] + 4 * fixef(randCoef_fit)[2] +
  6 * fixef(randCoef_fit)[3] + ranef(randCoef_fit)$class$`(Intercept)`[18] +
  4 * ranef(randCoef_fit)$class$`grade`[18] + 6 *
    ranef(randCoef_fit)$class$`motivation`[18])
```

```
(Intercept)
   821.5456
```

```
# Überprüfen mit predict()
predict(randCoef_fit, newdata = tibble(grade = 4, motivation = 6, class = 18))
```

```
       1
821.5456
```

2. Modelle mit Level-2-Prädiktoren

2.1. Setup

Wir laden alle Packages mit `pacman`.

```
pacman::p_load(tidyverse, lme4, lmerTest, ggplot2)
```

Wir laden unsere Daten und definieren `company` und die neue Variable `sector` als Faktoren. `sector` besitzt die beiden Faktorstufen `Private` und `Public`, von denen `Private` als alphabetisch frühere Kategorie automatisch als erste und damit als Referenzkategorie gesetzt wird.

```
df <- read_csv(
  url("https://raw.githubusercontent.com/methodenlehre/data/master/statistik_IV/salary-dat
) |>
  mutate(
    company = as.factor(company),
    sector = as.factor(sector)
  )
```

Jetzt können wir uns die Daten anschauen.

```
tail(df)
```

```
# A tibble: 6 x 4
  company    experience salary sector
  <fct>           <dbl>  <dbl> <fct>
1 Company 20       3.58   6.84 Private
2 Company 20       3.18   7.60 Private
3 Company 20       3.39   5.71 Private
4 Company 20       7.12  10.1  Private
5 Company 20       2.98   6.94 Private
6 Company 20       6.45   9.33 Private
```

```
summary(df)
```

```
      company      experience         salary          sector
 Company 01: 30   Min.   : 0.000   Min.   : 4.587   Private:300
 Company 02: 30   1st Qu.: 4.027   1st Qu.: 7.602   Public :300
 Company 03: 30   Median : 5.170   Median : 8.564
 Company 04: 30   Mean   : 5.190   Mean   : 8.738
 Company 05: 30   3rd Qu.: 6.402   3rd Qu.: 9.840
 Company 06: 30   Max.   :10.000   Max.   :15.418
 (Other)   :420
```

2.2. Intercept-as-Outcome-Modell (Modell 4)

2.2.1. Kontexteffekt-Modell

Das Kontexteffekt-Modell ist eine Variante des Intercept-as-Outcome-Modells. Der Level-2-Prädiktor ist hier ein auf Gruppenebene aggregierter Level-1-Prädiktor (hier: `exp_groupmean`; diese Variable beinhaltet für jede Person den zugehörigen Firmenmittelwert von `experience`. Um die Einflüsse von Level-1- und Level-2-Prädiktor zu trennen, muss der Level-1-Prädiktor in diesem Modell als *gruppenzentrierte* Variable aufgenommen werden. Auf Level-1 wird also nur noch die *firmenspezifische* Abweichung von `experience` vom jeweiligen `exp_groupmean` modelliert, die gruppenzentierte Variable nennen wir daher `exp_centered`.

In unserem Beispiel ist der Level-2-Prädiktor die durchschnittliche Erfahrung pro Firma. Der Effekt dieser Variablen wird gemeinsam mit der gruppenzentrierten Erfahrung untersucht. Zusätzlich zu der Annahme, dass die individuelle Erfahrung mit einem höheren Lohn einhergeht, nehmen wir nun an, dass auch die durchschnittliche Erfahrung der Mitarbeiter einer Firma sich auf den durchschnittlichen Lohn einer Firma auswirkt. In diesem Beispiel ist die Annahme naheliegend, dass beide Effekte vorhanden sind und in die gleiche Richtung gehen. Das ist aber nicht immer der Fall (vgl. “Big-Fish-Little-Pond-Effekt”).

Im Kontexteffekt-Modell wird manchmal kein random slope zugelassen, weil man sich auf die fixed effects auf beiden Untersuchungsebenen konzentriert. Man könnte aber im Prinzip auch hier einen random slope von `exp_centered` zulassen. Wir verzichten hier darauf und verweisen auf das Allgemeine Intercept-as-Outcome-Modell weiter unten.

Level-1-Modell:

$y_{mi} = \beta_{0i} + \beta_{1i} \cdot (x_{mi} - \mu_{\cdot i(X)}) + \varepsilon_{mi}$

Level-2-Modell:

$\beta_{0i} = \gamma_{00} + \gamma_{01} \cdot \mu_{\cdot i(X)} + \upsilon_{0i}$

$\beta_{1i} = \gamma_{10}$

Gesamtmodell:
$y_{mi} = \gamma_{00} + \gamma_{10} \cdot (x_{mi} - \mu_{\cdot i(X)}) + \gamma_{01} \cdot \mu_{\cdot i(X)} + \upsilon_{0i} + \varepsilon_{mi}$

Der Level-2-Prädiktor `exp_groupmean` muss jedoch zuerst berechnet und jeder Person zugeteilt werden. Wir berechnen diese Variable mit der Pipe `|>` und benutzen `group_by()` und `mutate()` aus dem `tidyverse`-Package. Gleichzeitig berechnen wir eine gruppenzentrierte Variable `exp_centered`, indem wir für jede Person ihren jeweiligen Firmenmittelwert von der individuellen Erfahrung abziehen.

```
# Berechnen der Mittelwerte pro Gruppe (exp_groupmean),
# und der Gruppenmittelwert-zentrierten experience (exp_centered)
df <- df |>
  group_by(company) |>
  mutate(
    exp_groupmean = mean(experience),
    exp_centered = experience - exp_groupmean
  ) |>
  ungroup()
# select() zum Ändern der Variablenreihenfolge

# Jetzt können wir uns die Daten nochmals anschauen - hat alles funktioniert?
head(df)
```

```
# A tibble: 6 x 6
  company    experience salary sector  exp_groupmean exp_centered
  <fct>           <dbl>  <dbl> <fct>           <dbl>        <dbl>
1 Company 01       6.87   8.96 Private          6.37        0.496
2 Company 01       5.66   9.54 Private          6.37       -0.714
3 Company 01       4.58   7.30 Private          6.37       -1.79
4 Company 01       6.56   8.09 Private          6.37        0.186
5 Company 01       8.83  14.3  Private          6.37        2.46
6 Company 01       7.73  10.3  Private          6.37        1.36
```

```
tail(df)
```

```
# A tibble: 6 x 6
  company    experience salary sector  exp_groupmean exp_centered
  <fct>           <dbl>  <dbl> <fct>           <dbl>        <dbl>
1 Company 20       3.58   6.84 Private          5.04        -1.46
2 Company 20       3.18   7.60 Private          5.04        -1.86
```

```
3 Company 20       3.39   5.71 Private          5.04        -1.65
4 Company 20       7.12  10.1  Private          5.04         2.08
5 Company 20       2.98   6.94 Private          5.04        -2.06
6 Company 20       6.45   9.33 Private          5.04         1.41
```

```r
# exp_groupmean anschauen
df |>
  group_by(company) |>
  summarise(exp_groupmean = mean(exp_groupmean)) |>
  ungroup()
```

```
# A tibble: 20 x 2
   company    exp_groupmean
   <fct>              <dbl>
 1 Company 01          6.37
 2 Company 02          6.13
 3 Company 03          6.09
 4 Company 04          4.71
 5 Company 05          6.34
 6 Company 06          5.1
 7 Company 07          4.02
 8 Company 08          5.71
 9 Company 09          5.15
10 Company 10          5.21
11 Company 11          5.40
12 Company 12          4.91
13 Company 13          5.23
14 Company 14          4.70
15 Company 15          4.15
16 Company 16          4.88
17 Company 17          5.61
18 Company 18          5.03
19 Company 19          4.02
20 Company 20          5.04
```

Modell berechnen

```r
# Fitten und abspeichern des Modells
context.model <- lmer(salary ~ exp_centered + exp_groupmean + (1 | company),
  data = df, REML = TRUE
)

# Durch das Modell vorhergesagte Werte abspeichern
df$context.model.preds <- predict(context.model)

# Modell anschauen mit summary()
summary(context.model)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: salary ~ exp_centered + exp_groupmean + (1 | company)
   Data: df

REML criterion at convergence: 1865.5

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.7991 -0.6796  0.0043  0.6008  3.8752

Random effects:
 Groups   Name        Variance Std.Dev.
 company  (Intercept) 0.623    0.7893
 Residual             1.185    1.0884
Number of obs: 600, groups:  company, 20

Fixed effects:
               Estimate Std. Error        df t value Pr(>|t|)
(Intercept)     4.77925    1.38736  18.00000   3.445  0.00289 **
exp_centered    0.53188    0.02735 579.00000  19.446  < 2e-16 ***
exp_groupmean   0.76264    0.26499  18.00000   2.878  0.01001 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) exp_cn
exp_centerd  0.000
exp_groupmn -0.991  0.000
```

Der Effekt von `exp_centered` ist mit $\hat{\gamma}_{10} = 0.532$ sehr ähnlich wie im Random-Intercept-Modell. Der Effekt von `exp_groupmean` ist noch grösser - jedes Jahr durchschnittliche Erfahrung, die die Angestellten einer Firma mehr haben als die Angestellten einer anderen Firma, lässt den vorhergesagten Salary-Wert der Angestellten dieser Firma (und damit den Intercept $\hat{\beta}_{0i}$) um $\hat{\gamma}_{01} = 0.763$ Einheiten (ca. 763 CHF) steigen.

Visualisierung des Modells

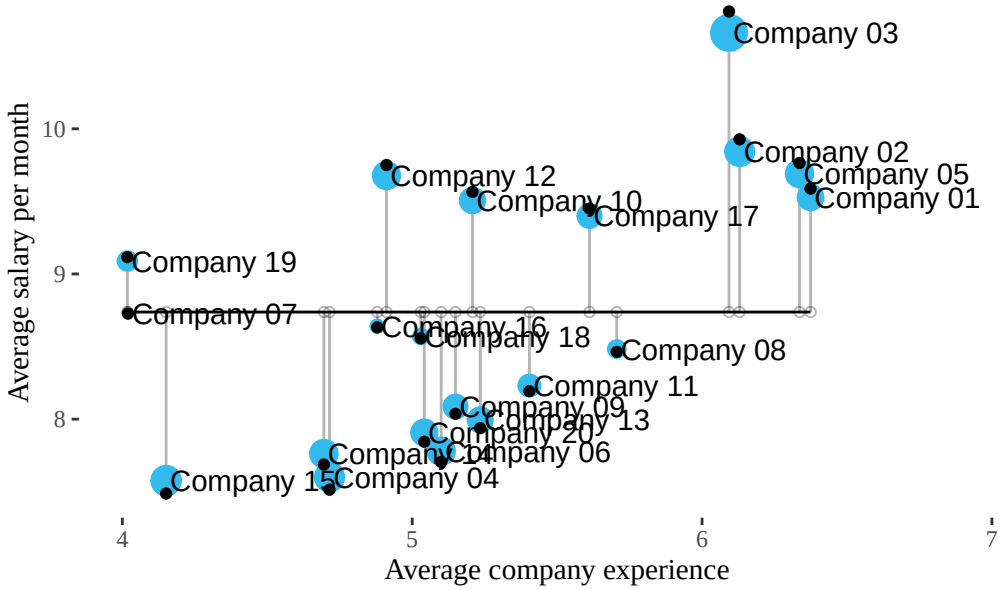
Die folgenden Plots spielen sich nur auf der Firmenebene (Level-2) ab. Der erste Plot zeigt die Abweichungen (Level-2-Residuen v_{0i}) der einzelnen Firmen-Intercepts vom Gesamtmittelwert der Intercepts ($\hat{\gamma}_{00}$) im Intercept-Only-Modell. Die Durchschnittsgehälter der Firmen unterscheiden sich stark vom Mittelwert der Gehälter aller Firmen (hellblaue Punkte). Zur Veranschaulichung sind ausserdem die (ohne Shrinkage geschätzten) empirischen `salary`-Mittelwerte pro Firma im Plot dargestellt (kleine schwarze Punkte).

Beim zweiten Plot kommt der Level-2 Prädiktor `exp_groupmean` (also die durchschnittliche Erfahrung, die die Angestellten jeder Firma haben) ins Spiel. Es zeigt sich, dass sich die `exp_groupmean` zwischen den Firmen unterscheidet, und dass `exp_groupmean` positiv mit den Intercepts des Intercept-Only-Modells korreliert (Steigung der Geraden $= \hat{\gamma}_{01} = 0.763$). Durch den Prädiktor nimmt die Abweichung der Gruppenmittelwerte von dem vorhergesagten Wert ab: Im Vergleich zum ersten Plot sind die Firmen-Punkte näher an der Linie, Teile der Level-2-Residuen konnten also durch den Level-2 Prädiktor erklärt werden.

Daraus hat das Modell 4 auch seinen Namen: Der Prädiktor auf Level-2 (`exp_groupmean`) sagt den (gruppenspezifischen) Intercept vorher ("Intercept-as-Outcome").

```
# Modell fitten
intercept.only.model <- lmer(salary ~ 1 + (1 | company), data = df, REML = TRUE)
# Daten vorbereiten
plot_df_intonly <- df |>
  mutate(
    intercept_only_predictions = predict(intercept.only.model), # level 1 predictions
    intercept_only_intercept = fixef(intercept.only.model)
  ) |> # overall
  group_by(company) |>
  summarise(
    overall = unique(intercept_only_intercept), # fixef intercept
    prediction = unique(intercept_only_predictions), # fixef intercept + ranef intercept
    observed = mean(salary), # actual mean of salary per company
    overall_to_prediction = prediction - overall, # ranef intercept
    company = unique(company), # name of company
    exp_groupmean = mean(experience)
```

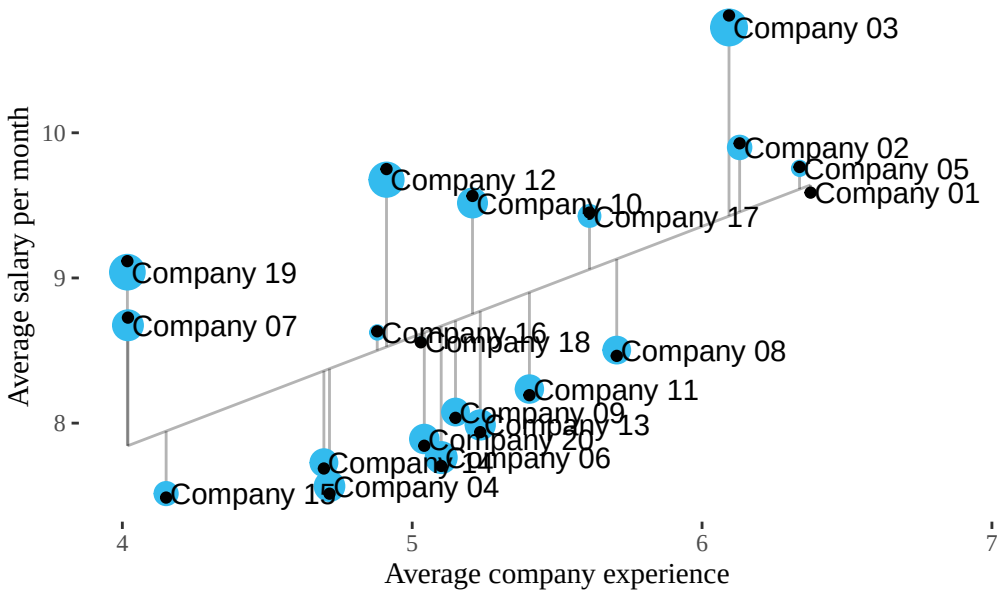
```r
  )
# Plotten
intercept_only_plot <- ggplot(aes(x = exp_groupmean, y = prediction),
  data = plot_df_intonly
) +
  geom_segment(aes(xend = exp_groupmean, yend = overall), alpha = .3) +
  # Color adjustments made here...
  geom_point(
    aes(
      # color = abs(overall_to_prediction),
      size = abs(overall_to_prediction)
    ),
    color = "#33BBEE"
  ) + # Size mapped to abs(residuals)
  geom_point(aes(y = observed)) +
  # scale_color_continuous(low = "black", high = "red") + # Colors to use here
  guides(color = "none", size = "none") + # Color legend removed
  geom_point(aes(y = overall), shape = 1, alpha = .3) +
  ggthemes::theme_tufte() +
  xlab("Average company experience") +
  ylab("Average salary per month") +
  geom_line(aes(x = exp_groupmean, y = overall, group = 1), alpha = 1) +
  ggtitle("Level-2-Residuen des Intercepts (Intercept-only Modell)")

# use ggplotly only if html:
if (knitr::is_html_output()) {
  intercept_only_plot |> plotly::ggplotly(tooltip = "company")
} else {
  intercept_only_plot +
    geom_text(aes(label = company), # add text to plot
      nudge_x = .3
    ) +
    lims(x = c(4, 7)) # zoom out a little, so everything can be seen
}
```



In dieser Grafik ist der Firmenmittelwert der Erfahrung auf der X-Achse abgebildet, obwohl diese Variable nicht im Modell ist. Dadurch lassen sich die Firmen und deren Residuen bezüglich der Löhne zwischen diesem und der nächsten Grafik direkt vergleichen, weil sich die Datenpunkte in der Darstellung nicht verschieben. Wir erkennen an der horizontalen Gerade der Modellprädiktion entlang der X-Achse, dass der Einfluss der durchschnittlichen Erfahrung nicht mit-modelliert wurde.

```
context.model <- lmer(salary ~ exp_centered + exp_groupmean + (1 | company), data = df, RE

companies <- df |>
  select(company) |>
  pull() |>
  levels()


plot_df_context <- tibble(
  company = companies,
  exp_groupmean = df |> group_by(company) |> pull(exp_groupmean) |> unique(),
  overall_nogroupmean = fixef(context.model)["(Intercept)"] + mean(exp_groupmean) * fixef(
  overall = fixef(context.model)["(Intercept)"] + exp_groupmean * fixef(context.model)["ex
  prediction = overall + ranef(context.model)$company |> select("(Intercept)") |> pull(),
```

```
  overall_to_prediction = ranef(context.model)$company |> select("(Intercept)") |> pull(),
  observed = df |> group_by(company) |> summarise(observed = mean(salary)) |> pull(observe
)

context_plot <- ggplot(aes(x = exp_groupmean, y = prediction, label = company),
  data = plot_df_context
) +
  geom_segment(aes(xend = exp_groupmean, yend = overall), alpha = .3) +
  # Color adjustments made here...
  geom_point(
    aes(
      # color = abs(overall_to_prediction),
      size = abs(overall_to_prediction)
    ),
    color = "#33BBEE"
  ) + # Size mapped to abs(residuals)
  geom_point(aes(y = observed)) +
  # scale_color_continuous(low = "black", high = "red") + # Colors to use here
  guides(color = FALSE, size = FALSE) + # Color legend removed
  geom_line(aes(y = overall), alpha = .3) +
  ggthemes::theme_tufte() +
  xlab("Average company experience") +
  ylab("Average salary per month") +
  ggtitle("Level-2-Residuen des Intercepts (Kontexteffekt-Modell)")

# use ggplotly only if html:
if (knitr::is_html_output()) {
  context_plot |> plotly::ggplotly(tooltip = "company")
} else {
  context_plot +
    geom_text(aes(label = company), # add text to plot
      nudge_x = .3
    ) +
    lims(x = c(4, 7)) # zoom out a little, so everything can be seen
}
```



2.2.2. Allgemeines Intercept-as-Outcome-Modell

Dies ist ein Modell *mit* Random Intercept, *mit oder ohne* Random Slope und *mit* Level-2-Haupteffekt (Prädiktion des Intercepts), aber ohne Prädiktor für den Slope (d.h. ohne Cross-Level-Interaktion).

Im Beispiel überprüfen wir, ob der Sektor (`sector`), in dem die Firmen tätig sind, einen Effekt auf `salary` hat, d.h. inwiefern die Höhe des Lohns davon abhängt, ob eine Firma im privaten oder öffentlichen Sektor operiert.

Der Effekt eines Level-2-Prädiktors könnte auch ohne weitere (Level-1-)Prädiktoren im Modell ermittelt werden. Wie in unserem Fall ist man häufig aber gleichzeitig am Effekt eines Level-1-Prädiktors `(experience)` interessiert. Der Level-1-Prädiktor kann in diesem Modell entweder mit ($\beta_{1i} = \gamma_{10} + \upsilon_{1i}$) oder ohne Random-Slope ($\beta_{1i} = \gamma_{10}$) modelliert werden (der Unterschied bezüglich dieses Effekts ähnelt dem Unterschied zwischen dem Random-Coefficients-Modell und dem Random-Intercept-Modell, vgl. Kapitel 1 Modelle mit Level-1-Prädiktoren). Wir beschränken uns hier auf ein Intercept-as-Outcome-Modell **mit** Random-Slope.

Level-1-Modell:
$y_{mi} = \beta_{0i} + \beta_{1i} \cdot x_{mi} + \varepsilon_{mi}$

Level-2-Modell:
$\beta_{0i} = \gamma_{00} + \gamma_{01} \cdot z_i + \upsilon_{0i}$
$\beta_{1i} = \gamma_{10} + \upsilon_{1i}$

Gesamtmodell:
$y_{mi} = \gamma_{00} + \gamma_{10} \cdot x_{mi} + \gamma_{01} \cdot z_i + \upsilon_{0i} + \upsilon_{1i} \cdot x_{mi} + \varepsilon_{mi}$

Berechnen des Modells

Wie bei den anderen Modellen benutzen wir die `lmer()`-Funktion aus dem Package `lme4`:

Im Gegensatz zur Variable `exp_groupmean` (Level-2-Prädiktor im Kontexteffekt-Modell) ist `sector` eine kategoriale Variable (Faktor). Praktischerweise übernimmt `lme4` die Dummy-Kodierung (mit der ersten Faktorstufe `Private` als Referenzkategorie) für uns.

```
intercept.as.outcome.model <- lmer(
  salary ~ experience + sector +
    (experience | company),
  data = df, REML = TRUE
)


# Prädiktionen abspeichern als Variable im Datenframe
df$intercept.as.outcome.preds <- predict(intercept.as.outcome.model)

# Modelloutput anschauen
summary(intercept.as.outcome.model)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: salary ~ experience + sector + (experience | company)
   Data: df

REML criterion at convergence: 1854.2

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.8397 -0.6713  0.0018  0.6041  3.3846

Random effects:
 Groups   Name        Variance Std.Dev. Corr
```

```
 company  (Intercept) 0.48711  0.6979
          experience  0.01861  0.1364   -0.31
 Residual             1.13631  1.0660
Number of obs: 600, groups:  company, 20

Fixed effects:
             Estimate Std. Error       df t value Pr(>|t|)
(Intercept)   5.65787    0.28568 24.12240  19.805  < 2e-16 ***
experience    0.53242    0.04077 19.06039  13.060 5.86e-11 ***
sectorPublic  0.52951    0.34342 18.15358   1.542     0.14
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) exprnc
experience  -0.491
sectorPublc -0.662  0.062
```

In der Stichprobe verdienen Angestellte im öffentlichen Sektor im Schnitt ca. 530 CHF mehr als Angestellte im privaten Sektor ($\hat{\gamma}_{01} = 0.530$), dieser Effekt ist aber nicht signifikant ($p = 0.1404$) und darf daher nicht weiter interpretiert werden. Auch die Interpretation des Intercepts $\hat{\gamma}_{00} = 5.658$ hat sich verändert: Dieser bedeutet jetzt, dass der durchschnittliche Monatslohn für einen Angestellten einer (unbekannten) Firma im privaten Sektor (Referenzkategorie von `sector`) ohne Erfahrung (an der Stelle `experience = 0`) ca. 5658 CHF beträgt.

> **Vertiefung**
>
> Wir können das mit der Funktion `predict()` überprüfen. Dafür müssen wir einen neuen Datenframe erstellen, der alle Modell-relevanten Variablen enthält (`experience`, `sector` & `company`).

```r
# Neues Daten-Frame für die Prädiktion einer Person aus einer
# unbekannten Firma mit 0 Jahre experience und aus dem
# öffentlichen Sektor (die Referenzkategorien).
df.new <- data.frame(
  company = NA,
  sector = "Private",
  experience = 0
)

prediction <- predict(intercept.as.outcome.model, # Modell für die Prädiktion
  newdata = df.new, #  neue Daten, für die salary vorhergesagt werden soll
  re.form = NA
) |> # damit firma = NA sein kann
  print()
```

```
       1
5.657873
```

```r
# Test, dass diese Vorhersage == dem Intercept aus dem
# intercept.as.outcome.model ist. Wir können uns hier dafür die Differenz anschauen.
prediction[[1]] - fixef(intercept.as.outcome.model)[["(Intercept)"]]
```

```
[1] 0
```

Die Differenz ist 0, die zwei Werte sind also identisch.

2.3. Slope-as-Outcome-Modell (Modell 5)

In diesem Modell wollen wir untersuchen, ob die in Modell 3 (Random-Coefficents-Modell mit nur Level-1-Prädiktor `experience`) gefundene (signifikante) Level-2-Varianz des Slopes von `experience` ($\hat{\sigma}^2_{\upsilon_1}$) durch den Level-2-Prädiktor `sector` vorhergesagt werden kann, d.h. ob der Effekt von `experience` vom Sektor, in dem eine Firma operiert, beeinflusst wird.

Auch in Modell 4 haben wir eine ähnlich hohe Slope-Varianz erhalten, diese aber nicht auf Signifikanz getestet. Streng genommen ist Modell 4 jetzt das Vergleichsmodell (und nicht Modell 3), da dieses (wie Modell 5) den Level-2-Effekt von `sector` auf den Intercept enthält. Dieser kleine Unterschied spielt aber hauptsächlich für die Berechnung der erklärten Varianz des Slopes eine Rolle (s.u.).

Level-1-Modell:
$y_{mi} = \beta_{0i} + \beta_{1i} \cdot x_{mi} + \varepsilon_{mi}$

Level-2-Modell:
$\beta_{0i} = \gamma_{00} + \gamma_{01} \cdot z_i + \upsilon_{0i}$
$\beta_{1i} = \gamma_{10} + \gamma_{11} \cdot z_i + \upsilon_{1i}$

Gesamtmodell:
$y_{mi} = \gamma_{00} + \gamma_{10} \cdot x_{mi} + \gamma_{01} \cdot z_i + \gamma_{11} \cdot x_{mi} \cdot z_i + \upsilon_{0i} + \upsilon_{1i} \cdot x_{mi} + \varepsilon_{mi}$

2.3.1. Berechnen des Modells

Mit `lmer()`-Funktion (Package `lme4` bzw. `lmerTest`):

```
# Berechnen und Abspeichern des Modells
slope.as.outcome.model <- lmer(
  salary ~ experience * sector +
    (experience | company),
  data = df, REML = TRUE
)

# Modell-output anschauen
summary(slope.as.outcome.model)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: salary ~ experience * sector + (experience | company)
   Data: df

REML criterion at convergence: 1849.1

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.8271 -0.6923 -0.0319  0.6088  3.4093

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 company  (Intercept) 0.341721 0.58457
          experience  0.007713 0.08782  0.13
 Residual             1.137612 1.06659
Number of obs: 600, groups:  company, 20
```

```
Fixed effects:
                          Estimate Std. Error       df t value Pr(>|t|)
(Intercept)                5.28141    0.29513 21.66283  17.895 1.84e-14 ***
experience                 0.64430    0.04789 19.20313  13.455 3.15e-11 ***
sectorPublic               1.21269    0.39405 17.30455   3.078  0.00672 **
experience:sectorPublic -0.21695    0.06666 18.05226  -3.255  0.00439 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) exprnc sctrPb
experience  -0.566
sectorPublc -0.749  0.424
exprnc:sctP  0.406 -0.718 -0.525
```

Das Ergebnis zeigt eine signifikante Cross-Level-Interaktion ($\hat{\gamma}_{11} = -0.217, p = 0.0044$) in die erwartete Richtung (stärkere Gehaltssteigerungen im privaten Sektor). Der simple slope (bedingte Effekt) von `experience` zeigt ausserdem die durchschnittliche Gehaltssteigerung im privaten Sektor an ($\hat{\gamma}_{10} = 0.644, p = 0$). Die durchschnittliche Gehaltssteigerung im öffentlichen Sektor können wir als $\hat{\gamma}_{10} + \hat{\gamma}_{11} = 0.644 + (-0.217) = 0.427$ berechnen.

Im Output ausserdem ablesbar (gewissermassen als "Nebeneffekt") ist das Ergebnis, dass unerfahrene Angestellte (`experience = 0`) im öffentlichen Sektor signifikant mehr ($\hat{\gamma}_{01} = 1.213, p = 0.0067$) verdienen als im privaten Sektor (bedingter Effekt/simple slope von `sector` im Term `sectorPublic`).

2.3.2. Modellvergleich (Signifikanztest) für die verbliebene Slope-Varianz

Jetzt möchten wir noch testen, ob die in Modell 5 verbliebene Slope-Varianz noch signifikant ist. Das machen wir über einen Modellvergleich mit einem Modell, das sich **nur darin** von unserem Slope-as-Outcome-Modell unterscheidet, dass es **keine** Slope-Varianz schätzt:

```r
ranova(slope.as.outcome.model)
```

```
ANOVA-like table for random-effects: Single term deletions

Model:
salary ~ experience + sector + (experience | company) + experience:sector
                                    npar  logLik    AIC    LRT Df Pr(>Chisq)
<none>                                 8 -924.53 1865.1
```

```
experience in (experience | company)    6 -927.04 1866.1 5.0152  2    0.08146 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Nicht mehr signifikant! Das gilt (knapp) auch nach Korrektur des p-Wertes:

```
0.5 * pchisq(5.0152, 1, lower.tail = FALSE) +
  0.5 * pchisq(5.0152, 2, lower.tail = FALSE)
```

```
[1] 0.05329462
```

Nach Berücksichtigung von `sector` als Prädiktor des Level-1-Effekts von `experience` (und damit der Cross-Level-Interaktion) gibt es also keine signifikante Slope-Varianz von `experience` mehr.

2.3.3. Varianzerklärung durch Cross-Level-Interaktionseffekt

Wir wissen nun bereits, dass die in den Modellen 3 und 4 vorhandenen `experience`-Steigungsunterschiede zwischen den Firmen auf den Sektor, in dem die Firmen operieren, zurückzuführen sind. D.h. dass nach Berücksichtigung der Tatsache, dass sich `Private` und `Public` Firmen im Schnitt bezüglich des Einflusses von Erfahrung auf den Lohn unterscheiden, sich innerhalb der beiden Sektoren keine signifikante Varianz des `experience`-Effekts mehr zwischen den Firmen zeigt.

Rein deskriptiv bleibt aber noch Slope-Varianz übrig. Welcher Anteil der Level-2-Varianz des `experience`-Slopes wurde also durch `sector` erklärt (siehe Eid, Gollwitzer, und Schmitt 2017, pp. 753)? Dazu vergleichen wir mit dem Intercept-as-Outcome-Modell von oben, welches sich nur durch den Interaktionseffekt von Modell 5 unterscheidet.

$$R^2_{Cross-level} = \frac{\sigma^2_{v_{14}} - \sigma^2_{v_{15}}}{\sigma^2_{v_{14}}} = \frac{0.0186 - 0.0077}{0.0186} = 0.5856$$

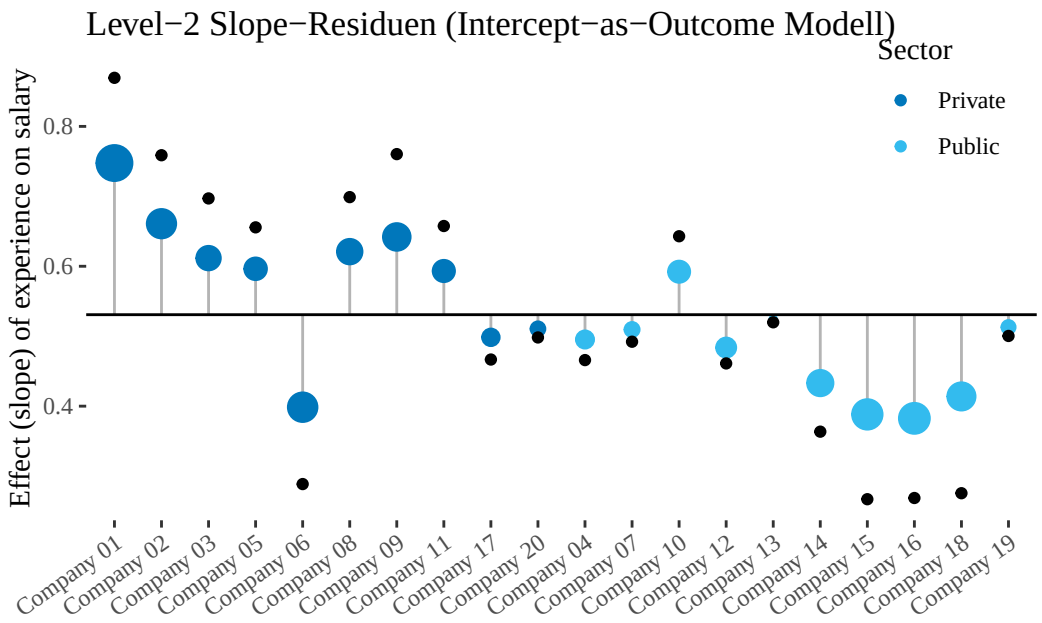
Im Sinne einer Effektgrösse für die Cross-Level-Interaktion ($\hat{\gamma}_{11}$) wurden 58.56 % der Slope-Varianz von `experience` durch Sektorenunterschiede im Effekt von `experience` auf `salary` erklärt.

In den folgenden beiden Plots wird die Auswirkung der Cross-Level-Interaktion auf die Slope-Residuen von `experience` veranschaulicht. Der erste Plot zeigt die Slope-Residuen relativ zum durchschnittlichen Effekt von `experience` im Intercept-as-Outcome Modell ($\hat{\gamma}_{10} = 0.532$). Der zweite Plot zeigt die Slope-Residuen im Modell mit Cross-Level-Interaktion (Slope-as-Outcome-Modell) relativ zu den jeweiligen Simple Slopes von `experience` in den Sektoren `Private` und `Public`. Die kleinen schwarzen Punkte stehen wieder für die pro Gruppe in getrennten

Regressionen (d.h. ohne Shrinkage) geschätzten Slopes. Klar erkennbar ist, dass die Residuen über alle Firmen hinweg im Slope-as-Outcome Modell deutlich kleiner ausfallen.

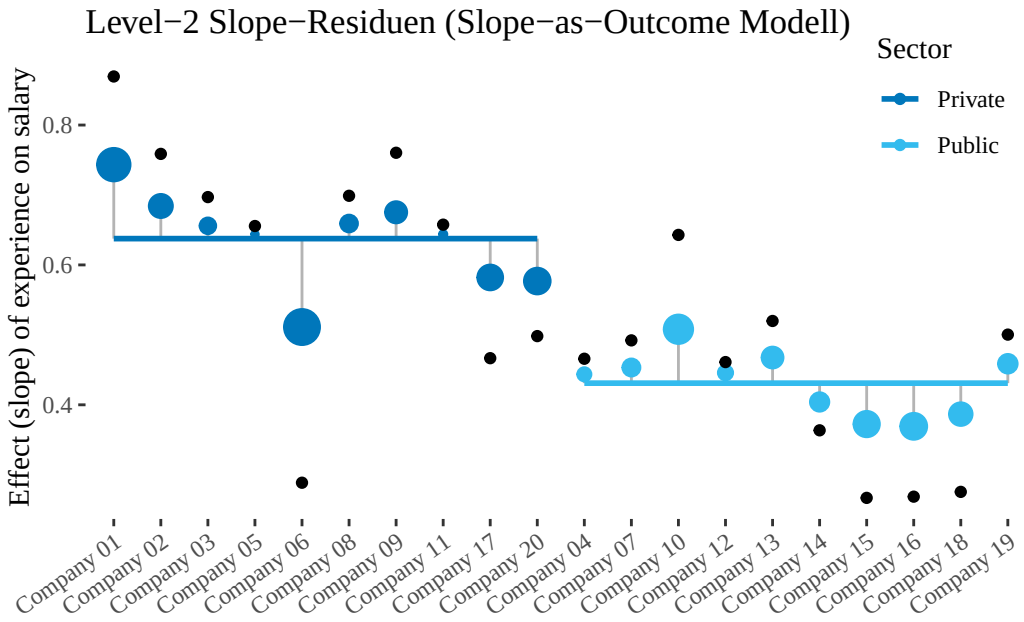
```
intercept.as.outcome <- lmer(salary ~ exp_centered + sector + (-1 + exp_centered | company
slope.as.outcome.lm <- lme4::lmList(formula = salary ~ exp_centered | company, data = df)


plot_df_iao_s <- tibble(
  company = companies,
  sector = df |> group_by(company) |> summarise(sector = unique(sector)) |> pull(),
  overall_nosector = fixef(intercept.as.outcome)["exp_centered"],
  prediction = overall_nosector + {
    ranef(intercept.as.outcome)$company |>
      select("exp_centered") |>
      pull()
  },
  overall_to_prediction = {
    ranef(intercept.as.outcome)$company |>
      select("exp_centered") |>
      pull()
  },
  observed = coef(slope.as.outcome.lm) |> select("exp_centered") |> pull()
) |>
  arrange(sector)
plot_df_iao_s <- plot_df_iao_s |>
  mutate(company = factor(company, c(unique(company))))


ggplot(aes(x = company, y = prediction),
  data = plot_df_iao_s
) +
  geom_segment(aes(xend = company, yend = overall_nosector), alpha = .3) +
  # Color adjustments made here...
  geom_point(aes(size = abs(overall_to_prediction), color = sector)) + # size mapped to ab
  geom_point(aes(y = observed)) +
  # scale_color_continuous(low = "black", high = "red") + # Colors to use here
  guides(size = "none") + # Color legend removed
  labs(color = "Sector") + # Change title of color legend
  geom_hline(aes(yintercept = plot_df_iao_s$overall_nosector[[1]])) +
  ggthemes::theme_tufte() +
  scale_color_manual(values = c("#0077BB", "#33BBEE")) +
  xlab("") +
```

```
  ylab("Effect (slope) of experience on salary") +
  ggtitle("Level-2 Slope-Residuen (Intercept-as-Outcome Modell)") +
  #  geom_text(aes(label = company), hjust = 1, vjust = 0) +# still have to fix it
  theme(
    axis.text.x = element_text(angle = 35, hjust = 1),
    legend.position = c(.9, .9)
  )
```




```
slope.as.outcome <- lmer(salary ~ exp_centered * sector + (exp_centered - 1 | company) + (


# to get the "observed" slopes without shrinkage we have to fit a linear model for each co
slope.as.outcome.lm <- lme4::lmList(formula = salary ~ exp_centered | company, data = df)

plot_df_sao <- tibble(
  company = companies,
  sector = df |> group_by(company) |> summarise(sector = unique(sector)) |> pull(),
  overall_nosector = fixef(slope.as.outcome)["exp_centered"] +
    mean(dummy(sector)) * fixef(slope.as.outcome)["exp_centered:sectorPublic"],
  overall = fixef(slope.as.outcome)["exp_centered"] +
    dummy(sector)[, 1] * fixef(slope.as.outcome)[["exp_centered:sectorPublic"]],
  prediction = overall + {
    ranef(slope.as.outcome)$company |>
```

```
      select("exp_centered") |>
      pull()
  },
  overall_to_prediction = {
    ranef(slope.as.outcome)$company |>
      select("exp_centered") |>
      pull()
  },
  overall_nosector_to_prediction = prediction - overall_nosector,
  observed = coef(slope.as.outcome.lm) |> select("exp_centered") |> pull()
) |>
  arrange(sector)



# reorder companies
plot_df_sao <- plot_df_sao |>
  mutate(company = factor(company, c(unique(company))))


ggplot(aes(x = company, y = prediction),
  data = plot_df_sao
) +
  geom_segment(aes(xend = company, yend = overall), alpha = .3) +
  geom_point(aes(color = sector, size = abs(overall_to_prediction))) + # Color mapped to a
  geom_point(aes(y = observed)) +
  scale_color_manual(values = c("#0077BB", "#33BBEE")) +
  guides(size = FALSE) + # Color legend removed
  labs(color = "Sector") + # Change title of color legend
  geom_smooth(aes(x = company, y = overall, color = sector, group = sector), method = "lm"
  ggthemes::theme_tufte() +
  xlab("") +
  ylab("Effect (slope) of experience on salary") +
  ggtitle("Level-2 Slope-Residuen (Slope-as-Outcome Modell)") +
  theme(
    axis.text.x = element_text(angle = 35, hjust = 1),
    legend.position = c(.9, .9)
  )
```



2.4. Übung

Wir verwenden wieder die selben Daten wie in Übung 1.

`id` : Durchnummerierung aller Schüler (1:1000)

`name` : Name der Schülerin / desSchülers

`motivation` : Wie motiviert die Schüler für den Pisa-Test sind

`grade` : Die durchschnittliche Vornote jedes Schülers (gerundet auf 0.25)

`class` : Variable der Schulklasse (durchnummeriert, 50 Klassen wurden zufällig ausgewählt)

`teacher_salary` : Monatslohn des Lehrers (jede Klasse hat einen anderen Lehrer)

`school` : Kategoriale Variable der Schule (10 Schulen wurden zufällig ausgewählt)

`pisa` : Erreichte Punktzahl beim Pisa-Test.

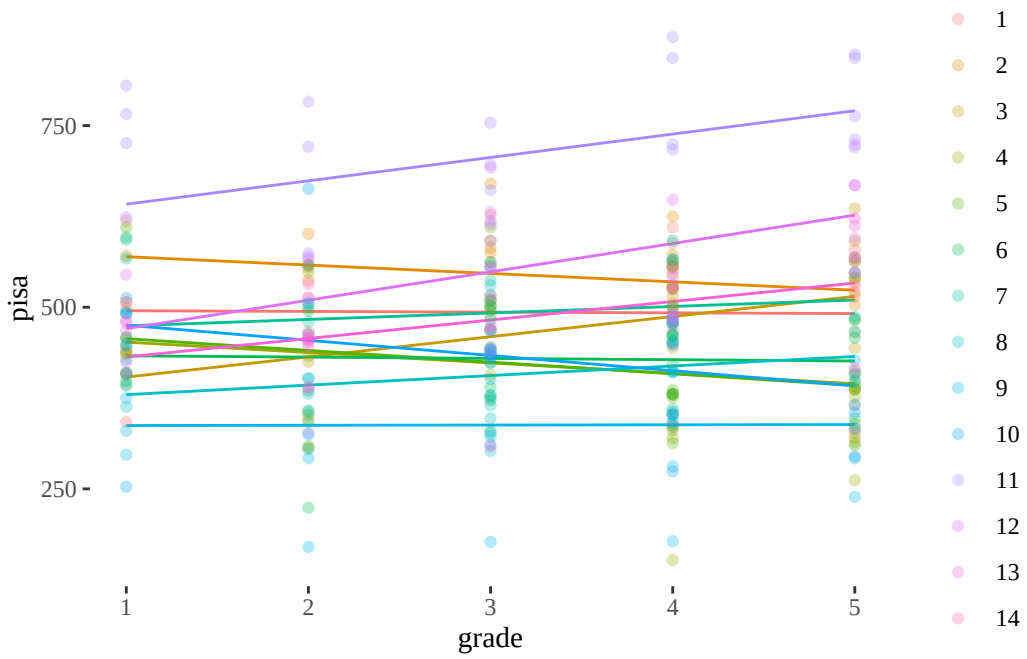
Aus jeder der 10 Schulen wurden 5 Klassen ausgesucht. Insgesamt sind es also 50 Klassen. Für die folgenden Aufgaben beachten wir die Nestung der Klassen in den verschiedenen Schulen nicht, da wir dafür ein 3-Ebenen-Modell bräuchten. Unser Sample auf Level-2 sind also die 50 Schulklassen.

Um diese herunterzuladen und aufzubereiten können wir also auf den Codechunk von letzter Woche zurückgreifen:

```
# Einlesen
pacman::p_load(lme4, nlme, tidyverse, lmerTest, gridExtra, ggplot2)
school_df <- read_csv(url("https://raw.githubusercontent.com/methodenlehre/data/master/sta
# Aufbereiten
school_df <- school_df |> mutate(
  id = as.factor(id),
  class = as.factor(class)
)
```

Die Daten kann man sich etwa so vorstellen (subsample, nur 15 Klassen für die Visualisierung):

```
subsample <- school_df[1:300, ]
subsample <- subsample |> mutate(class = as.factor(class))

ggplot(data = subsample, aes(x = grade, y = pisa, group = class, color = class)) +
  geom_smooth(method = "lm", alpha = .5, se = FALSE, show.legend = F, fullrange = TRUE, si
  geom_point(alpha = .3) +
  theme_tufte()
```



1. Kontexteffekt-Modell

a) Fitten Sie ein Kontexteffekt-Modell, welches die Klassenmittelwerte von `grade` und `motivation` als Prädiktoren im Modell hat (zusätzlich zu den Level-1-Effekten dieser Variablen).

Weil wir ja schon, dass es signifikante Random Slope Varianzen gibt, sollten diese auch Teil des Modells sein.

Beachten Sie, dass die Level-1-Prädiktoren der Motivation (`motivation`) und der Vornote (`grade`) in diesem Fall als gruppenzentrierte Variablen ins Modell genommen werden müssen. Diese müssen - wie auch die Gruppenmittelwerte dieser Variablen - zunächst berechnet werden.

Lösung

```
school_df <- school_df |>
  group_by(class) |>
  mutate(
    motivation_groupmean = mean(motivation),
    motivation_centered = motivation - motivation_groupmean,
    grade_groupmean = mean(grade),
    grade_centered = grade - grade_groupmean
  )

context_model <- lmer(
  formula = pisa ~ motivation_groupmean + motivation_centered + grade_groupmean + grade_
    (motivation_centered + grade_centered | class),
  data = school_df
)
```

b) Interpretieren Sie alle Fixed Effects dieses Modells.

Lösung

```
summary(context_model)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: pisa ~ motivation_groupmean + motivation_centered + grade_groupmean +
    grade_centered + (motivation_centered + grade_centered |      class)
```

```
   Data: school_df

REML criterion at convergence: 13579.6

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.2030 -0.6152  0.0243  0.6313  2.7130

Random effects:
 Groups   Name                Variance Std.Dev. Corr
 class    (Intercept)         12951.6  113.80
          motivation_centered   519.1   22.78   0.50
          grade_centered        272.7   16.51   0.57 0.23
 Residual                      6382.2   79.89
Number of obs: 1150, groups:  class, 50

Fixed effects:
                     Estimate Std. Error      df t value Pr(>|t|)
(Intercept)           427.267    305.723  48.972   1.398   0.1685
motivation_groupmean   -9.419     58.681  48.446  -0.161   0.8731
motivation_centered     7.012      3.944  48.697   1.778   0.0816 .
grade_groupmean        39.239     46.639  49.152   0.841   0.4042
grade_centered         14.957      2.927  47.024   5.110 5.78e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) mtvtn_g mtvtn_c grd_gr
mtvtn_grpmn -0.880
mtvtn_cntrd  0.049 -0.032
grade_grpmn -0.537  0.076  -0.001
grade_cntrd  0.013  0.010   0.162   0.005
```

Insgesamt wurde nur einer der festen Effekte signifikant, nämlich `grade_centered`. `motivation_groupmean`, `motivation_centered` & `grade_groupmean` scheinen in diesem Modell keinen Einfluss auf den `pisa`- Wert zu haben.
Auf die Vornote als Prädiktor bezogen bedeutet das: Höhere durchschnittliche Vornoten (`grade_groupmean`) der Schüler:innen einer Klasse sind *nicht* mit höheren (durchschnittlichen) `pisa`-Werten einer Klasse assoziiert. Die Vornote eines Schülers im Vergleich zu Mitschülern aus derselben Klasse (`grade_centered`) hat dagegen sehr wohl einen Einfluss auf die `pisa` Werte. Diesen Effekt kennen wir aber bereits aus dem Random-Coefficents-Modell aus der Übung zu Kapitel 1. So haben Schüler:innen derselben Klasse durch-

schnittlich einen um 14.957 Punkte höheren `pisa` Wert für jede (ganze) Note, die Sie besser sind als ihre Mitschüler:innen.
Die Motivation scheint hingegen weder auf Klassenebene noch auf Individualebene einen Einfluss auf die `pisa`-Werte zu haben.
Zu guter Letzt: Der Intercept wäre hier der vorhergesagte `pisa`-Wert einer Person mit Klassen-durchschnittlicher Motivation und Vornote (`motivation_centered = 0` und `grade_centered = 0`) in einer Klasse, deren Mittelwert sowohl von Motivation und Vornote gleich 0 ist (`motivation_groupmean = 0` und `grade_groupmean = 0`). Da letzteres insbesondere in Bezug auf `grade_groupmean` unmöglich ist (Notenskala von 1-6), ist eine Interpretation des Intercepts hier nicht wirklich sinnvoll.

2. Intercept-as-Outcome-Modell

a) Welche der Variablen im Datenframe bietet sich an, als Level-2-Prädiktor (auf Klassenebene) eingesetzt zu werden?

Lösung

Die Variable `teacher_salary` ist auf Klassen-Ebene (also Level-2), weil dieser Wert pro Klasse immer konstant bleibt und verschiedene Klassen unterschiedliche Werte haben. Wir können mit dieser Variable überprüfen, ob der Lohn eines Lehrers/einer Lehrerin mit dem Pisa-Resultat der Schüler zusammenhängt. Auch hier muss man mit der Frage der Kausalität aufpassen: Eine ziemlich offensichtliche konfundierende Variable könnte die Erfahrung des Lehrers/der Lehrerin sein (da erfahrenere Lehrer:innen oft mehr verdienen als weniger erfahrene).

b) Berechnen Sie das Intercept-as-Outcome-Modell mit random Slopes. Tipp: Dieses Modell hat nun 3 Prädiktoren im Modell.

Lösung

```
intercept_as_outcome_model <- lmer(formula = pisa ~ grade + motivation +
  teacher_salary + (motivation + grade | class), data = school_df)
summary(intercept_as_outcome_model)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: pisa ~ grade + motivation + teacher_salary + (motivation + grade |
    class)
   Data: school_df
```

```
REML criterion at convergence: 13592

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.2012 -0.6111  0.0230  0.6365  2.7374

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 class    (Intercept) 10129.1  100.64
          motivation    514.9   22.69   -0.57
          grade         274.8   16.58   -0.11  0.23
 Residual              6382.4   79.89
Number of obs: 1150, groups:  class, 50

Fixed effects:
               Estimate Std. Error      df t value Pr(>|t|)
(Intercept)     367.899     97.584  50.205   3.770 0.000431 ***
grade            14.965      2.933  47.163   5.102 5.92e-06 ***
motivation        6.967      3.927  48.742   1.774 0.082270 .
teacher_salary    9.314     14.283  48.816   0.652 0.517385
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) grade  motvtn
grade       -0.046
motivation  -0.124  0.166
teachr_slry -0.982 -0.002 -0.006
optimizer (nloptwrap) convergence code: 0 (OK)
Model failed to converge with max|grad| = 0.00624123 (tol = 0.002, component 1)
```

Der Effekt des Level-2-Prädiktors `teacher_salary` ist positiv, aber nicht signifikant. Zwar würde der vorhergesagte Pisawert rein deskriptiv für die vorliegende Stichprobe mit zusätzlichen 1000 Franken Lehrergehalt um $\hat{\gamma}_{01} = 9.314$ ansteigen, dieser Effekt darf aber aufgrund der fehlenden Signifikanz nicht weiter interpretiert werden.

3. Modelle mit Messwiederholung

3.1. Lineare Trendmodelle | Psychotherapie und Wohlbefinden

Für die folgenden Modelle benutzen wir das Package `nlme` und dessen Funktion `lme()`, das für RM-HLM Modelle besser geeignet ist als `lme4` mit der Funktion `lmer()`. Die Syntax unterscheidet sich kaum, der einzige grössere Unterschied bezieht sich auf die Formulierung des zufälligen Teils des Modells: statt `+ (woche | id)` (für einen random effect des Intercepts und des Level-1-Prädiktors `woche`, die in `id` genestet sind) schreibt man `, random = ~woche | id`

Packages und Daten laden:

Wir laden alle Packages mit `pacman`.

```
pacman::p_load(tidyverse, nlme, lme4, lmerTest, ggplot2, ggthemes)
```

Das Beispiel ist aus dem Buch von Eid, Gollwitzer, und Schmitt (2017). Die Daten sind in den Online-Materialien verfügbar.

Da wir die Variablen teilweise anders benennen und die Variable `bedingung` direkt mit den korrekten Faktorstufen-Bezeichnungen einlesen wollen, stellen wir sie unter folgendem Link angepasst zur Verfügung:

```
therapie <- read_csv("https://raw.githubusercontent.com/methodenlehre/data/master/therapie
  mutate(
    id = as.factor(id),
    bedingung = as.factor(bedingung)
  )

glimpse(therapie)
```

```
Rows: 563
Columns: 4
$ id          <fct> 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, ~
$ woche       <dbl> 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 5, 6, 0, 1, 2, 3, 4, ~
$ wohlbefinden <dbl> 2.4, 2.0, 3.2, 1.8, 1.6, 2.0, 1.6, 1.6, 3.6, 4.0, 3.4, 3.~
```

```
$ bedingung    <fct> Kontrollgruppe, Kontrollgruppe, Kontrollgruppe, Kontrollg~
```

```r
summary(therapie)
```

```
       id          woche        wohlbefinden            bedingung
 1      :  8   Min.   :0.000   Min.   :1.000   Kontrollgruppe:276
 4      :  8   1st Qu.:1.000   1st Qu.:2.800   Therapiegruppe:287
 6      :  8   Median :3.000   Median :3.800
 7      :  8   Mean   :3.259   Mean   :3.717
 14     :  8   3rd Qu.:5.000   3rd Qu.:4.800
 16     :  8   Max.   :7.000   Max.   :6.000
 (Other):515
```

```r
# nur Therapiegruppe
therapie_gr1 <- therapie |>
  filter(bedingung == "Therapiegruppe")
```

3.1.1. Level-1-Modelle (nur Therapiegruppe)

Eine Psychotherapeutin erhebt zu acht Messzeitpunkten das Wohlbefinden ihrer $n = 41$ Klienten mithilfe eines stetigen Merkmals, einmal zu Beginn der Therapie und dann in den folgenden sieben Wochen jeweils einmal im Anschluss an die jeweilige Therapiesitzung. Sie nimmt an, dass das Wohlbefinden mit jeder Woche (Therapiesitzung) konstant steigt, d.h. dass die UV Messzeitpunkt kodiert mit den Werten 0 bis 7 für die Messungen beginnend mit der Baseline vor Beginn der Therapie ($X = 0$) bis nach der siebten Therapiewoche ($X = 7$) einen positiven linearen Trend aufweist.

3.1.1.1. Intercept-Only-Modell (Modell 1) und Intraklassenkorrelation

Wir können davon ausgehen, dass sich die Personen in Bezug auf ihr Wohlbefinden insgesamt unterscheiden und es daher substantielle Level-2-Varianz in der AV gibt. Trotzdem wollen wir den Anteil der Level-2-Varianz (Personenvarianz) an der Gesamtvarianz der abhängigen Variablen quantifizieren (Intraklassenkorrelation).

Das Intercept-Only-Modell enthält keine Prädiktorvariable, nur der Intercept wird als Kombination eines festen (durchschnittlichen) Effekts und einer zufälligen Abweichung von diesem (Level-2-Residuum, Abweichung des durchschnittlichen Wohlbefindens einer Person vom Durchschnitt aller Personen) modelliert:

Gesamtmodell: $WB_{mi} = \gamma_{00} + \upsilon_{0i} + \epsilon_{mi}$

```r
intercept.only <- lme(wohlbefinden ~ 1,
  random = ~ 1 | id,
  therapie_gr1, method = "ML"
)
summary(intercept.only)
```

```
Linear mixed-effects model fit by maximum likelihood
  Data: therapie_gr1
       AIC     BIC    logLik
  857.5725 868.551 -425.7863

Random effects:
 Formula: ~1 | id
        (Intercept)  Residual
StdDev:    1.041132 0.9042715

Fixed effects:  wohlbefinden ~ 1
               Value Std.Error  DF  t-value p-value
(Intercept) 3.821784 0.1718299 246 22.24167       0

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-2.56699824 -0.48832807  0.05519999  0.58938383  3.07033213

Number of Observations: 287
Number of Groups: 41
```

Da `lme()` standardmässig nur die Standardabweichungen der random effects ausgibt, benutzt man die Funktion `VarCorr()`, um zusätzlich auch die Varianzen zu erhalten:

```r
VarCorr(intercept.only)
```

```
id = pdLogChol(1)
            Variance  StdDev
(Intercept) 1.0839548 1.0411315
Residual    0.8177069 0.9042715
```

Intraklassenkorrelation: $\hat{\rho} = \frac{\hat{\sigma}^2_{v_0}}{\hat{\sigma}^2_{v_0}+\hat{\sigma}^2_{\varepsilon}} = \frac{1.084}{1.084+0.8177} = 0.57$

$\Rightarrow$ 57 % der Gesamtvarianz sind auf Level-2-(Personen-)Unterschiede im Wohlbefinden zurückzuführen.

3.1.1.2. Random-Intercept-Modell (Modell 2)

Gibt es insgesamt (d.h. im Durchschnitt über alle Personen) einen konstanten (= linearen) Anstieg des Wohlbefindens über die Dauer der Therapie?

Berechnen Sie ein Random-Intercept-Modell (Modell 2) mit dem Level-1-Prädiktor `woche` (Messzeitpunkte mit den Werten 0-7) und und bestimmen Sie den Anteil der durch diesen linearen Trend erklärten Level-1-Varianz der AV `wohlbefinden`.

Dieses Modell enthält einen Level-1-Prädiktor, der jedoch als fester Effekt konzeptualisiert wird (kein Random Slope, nur Random Intercept).

Gesamtmodell: $WB_{mi} = \gamma_{00} + \gamma_{10} \cdot WOCHE_{mi} + \upsilon_{0i} + \varepsilon_{mi}$

```
random.intercept <- lme(wohlbefinden ~ woche,
  random = ~ 1 | id,
  therapie_gr1, method = "ML"
)
summary(random.intercept)
```

```
Linear mixed-effects model fit by maximum likelihood
  Data: therapie_gr1
       AIC      BIC    logLik
  835.5876 850.2256 -413.7938

Random effects:
 Formula: ~1 | id
        (Intercept)  Residual
StdDev:    1.044078 0.8615528

Fixed effects:  wohlbefinden ~ woche
               Value  Std.Error  DF   t-value p-value
(Intercept) 3.446048 0.18746082 245 18.382766       0
woche       0.114275 0.02285371 245  5.000283       0
 Correlation:
      (Intr)
woche -0.4
```

```
Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-2.8761965 -0.5191788  0.1015888  0.5007727  3.3528652

Number of Observations: 287
Number of Groups: 41
```

```
VarCorr(random.intercept)
```

```
id = pdLogChol(1)
            Variance  StdDev
(Intercept) 1.0900982 1.0440777
Residual    0.7422733 0.8615528
```

Die Ergebnisse zeigen, dass der fixed effect von `woche` positiv signifikant ist mit $\hat{\gamma}_{10} = 0.114$. Über alle Klienten hinweg steigt das vorhergesagte Wohlbefinden also mit jeder Therapiewoche um 0.114 Einheiten an.

Wieviel Level-1-Varianz wurde durch `woche` erklärt?

$R^2_{Level-1} = \frac{\hat{\sigma}^2_{\varepsilon_1} - \hat{\sigma}^2_{\varepsilon_2}}{\hat{\sigma}^2_{\varepsilon_1}} = \frac{0.8177 - 0.7423}{0.8177} = 0.0922$

$\Rightarrow$ ca. 9.22 % der Level-1-Varianz von `wohlbefinden` wurde durch `woche` (linearer Trend) erklärt.

3.1.1.3. Random-Coefficients-Modell (Modell 3)

Variiert der Effekt von `woche` (linearer Trend) signifikant zwischen den Personen?

Berechnen Sie ein Random-Coefficients-Modell (Modell 3) und überprüfen Sie mittels LR-Test (Vergleich mit Random-Intercept-Modell), ob sich die zufälligen Regressionsgewichte von `woche` signifikant zwischen den Personen (= Level-2-Einheiten) unterscheiden (Test von $\hat{\sigma}^2_{\upsilon_1}$ und $\hat{\sigma}_{\upsilon_0\upsilon_1}$ gegen 0).

Gesamtmodell: $WB_{mi} = \gamma_{00} + \gamma_{10} \cdot WOCHE_{mi} + \upsilon_{0i} + \upsilon_{1i} \cdot WOCHE_{mi} + \varepsilon_{mi}$

```
random.coefficients <- lme(wohlbefinden ~ woche,
  random = ~ woche | id,
  therapie_gr1, method = "ML"
)
summary(random.coefficients)
```

```
Linear mixed-effects model fit by maximum likelihood
  Data: therapie_gr1
       AIC     BIC   logLik
  830.5041 852.461 -409.252

Random effects:
 Formula: ~woche | id
 Structure: General positive-definite, Log-Cholesky parametrization
            StdDev    Corr
(Intercept) 1.0290556 (Intr)
woche       0.1381025 -0.182
Residual    0.8005709

Fixed effects:  wohlbefinden ~ woche
               Value  Std.Error  DF   t-value p-value
(Intercept) 3.455749 0.18260414 245 18.924809   0e+00
woche       0.114028 0.03074723 245  3.708548   3e-04
 Correlation:
      (Intr)
woche -0.384

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-2.78023820 -0.45874982  0.05703886  0.46723020  3.37849517

Number of Observations: 287
Number of Groups: 41
```

```
VarCorr(random.coefficients)
```

```
id = pdLogChol(woche)
            Variance   StdDev    Corr
(Intercept) 1.05895541 1.0290556 (Intr)
woche       0.01907231 0.1381025 -0.182
Residual    0.64091383 0.8005709
```

Die Standardabweichung des Effekts von Woche ist $\sqrt{\hat{\sigma}^2_{\upsilon_1}} = \hat{\sigma}_{\upsilon_1} = 0.1381$. Diese kann man sich als eine Art durchschnittliche Abweichung des linearen Trends einer Person vom durchschnittlichen linearen Trend vorstellen.

Schauen wir uns dazu mit `ranef()` einen Teil der Level-2-Residuen des `woche`-Effekts an:

```
ranef(random.coefficients)["woche"] |>
  round(2) |>
  tail(20)
```

```
   woche
52  0.00
54  0.00
58  0.05
59 -0.03
61 -0.04
64  0.14
66 -0.04
67  0.05
68 -0.15
71  0.00
72 -0.12
74  0.02
75  0.11
80  0.00
81 -0.11
83  0.06
84 -0.05
85 -0.27
88 -0.04
91  0.24
```

Es handelt sich wohlgemerkt um die *Residuen* des Random Slopes, also um die *Abweichungen* vom durchschnittlichen `woche`-Effekt $\hat{\gamma}_{10} = 0.114$. Die Standardabweichung dieser Abweichungswerte (und damit auch der individuellen `woche`-Effekte) ist wie schon gesagt $\hat{\sigma}_{\upsilon_1} = 0.1381$.

Wenn wir das Random-Coefficients-Modell mittels LR-Test mit dem Random-Intercept-Modell vergleichen, werden zwei Parameter gleichzeitig getestet: die Level-2-Varianz des Slopes ($\hat{\sigma}^2_{\upsilon_1}$) und die Level-2-Kovarianz/Korrelation zwischen Intercept und Slope ($\hat{\sigma}_{\upsilon_0\upsilon_1}$).

```
anova(random.coefficients, random.intercept)
```

```
                   Model df      AIC      BIC    logLik   Test  L.Ratio
random.coefficients     1  6 830.5041 852.4610 -409.2520
random.intercept        2  4 835.5876 850.2256 -413.7938 1 vs 2 9.083551
                   p-value
```

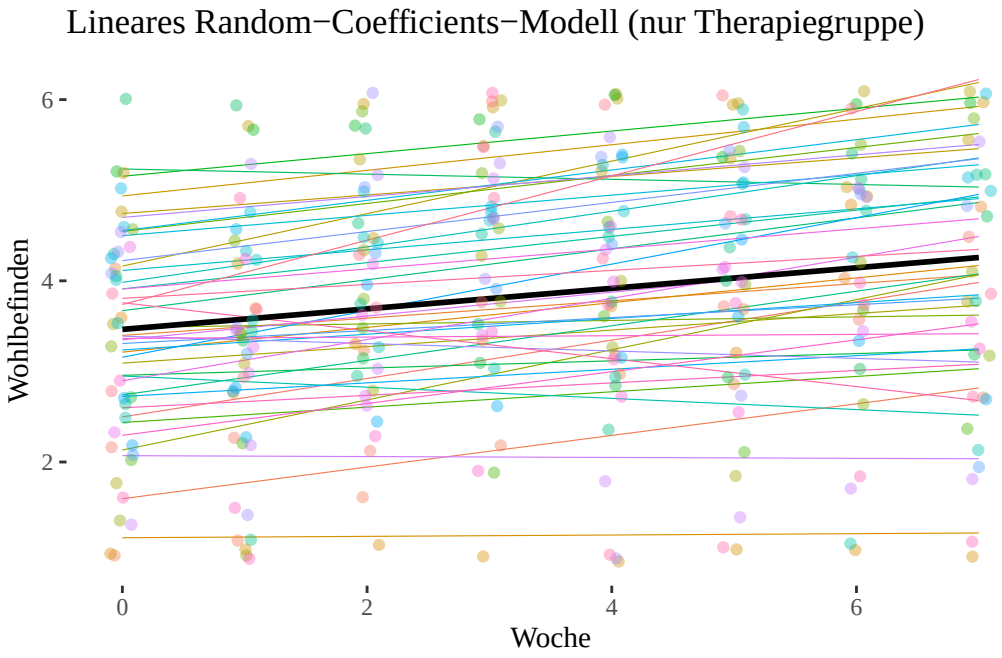
```
random.coefficients
random.intercept     0.0107
```

Das Ergebnis zeigt einen signifikanten LR-Test, die Klienten der Therapiegruppe unterscheiden sich also signifikant bezüglich der (linearen) Veränderung des Wohlbefindens über den Therapieverlauf. Die exakte Bestimmung des p-Werts erfolgt durch die Mittelung der p-Werte der beiden Verteilungen (oder durch Vergleich mit dem kritischen Wert von 5.14 für diese Mischverteilung):

```
0.5 * pchisq(9.083551, 1, lower.tail = FALSE) +
  0.5 * pchisq(9.083551, 2, lower.tail = FALSE)
```

```
[1] 0.00661683
```

```
therapie_gr1$random.coefficients.preds <- predict(random.coefficients)

ggplot(aes(x = woche, y = random.coefficients.preds, group = id), data = therapie_gr1) +
  geom_smooth(aes(color = id), method = "lm", se = FALSE, fullrange = TRUE, linewidth = .2
  geom_smooth(aes(group = 1), method = "lm", se = FALSE, fullrange = TRUE, color = "black"
  ggtitle("Lineares Random-Coefficients-Modell (nur Therapiegruppe)") +
  geom_jitter(aes(y = wohlbefinden, color = id), alpha = .4, width = 0.1, height = 0.1) +
  guides(color = "none") + # Legends removed
  xlab("Woche") +
  ylab("Wohlbefinden") +
  theme_tufte()
```



3.1.2. Level-2-Modelle (Therapie- und Kontrollgruppe)

Zusätzlich zu der in den Aufgaben 1-3 betrachteten Therapiegruppe ($n = 41$) wurde auch eine Kontrollgruppe erhoben ($n = 44$). Die Dummy-Variable `bedingung` ist jetzt ein Level-2-(Personen-)Merkmal Z. Wir erwarten, dass der lineare Anstieg des Wohlbefindens in der Therapiegruppe stärker ausfällt als in der Kontrollgruppe (Referenzkategorie von `bedingung`).

3.1.2.1. Slope-as-Outcome Modell (Modell 5)

Ist die Cross-Level-Interaktion signifikant und geht sie in die erwartete Richtung? Berechnen Sie für die Gesamtstichprobe ein Slope-as-Outcome Modell (Modell mit Cross-Level-Interaktion, Modell 5) mit dem Level-2-Prädiktor `bedingung`.

Level-1-Modell:

$$WB_{mi} = \beta_{0i} + \beta_{1i} \cdot WOCHE_{mi} + \varepsilon_{mi}$$

Level-2-Modell:

$$\beta_{0i} = \gamma_{00} + \gamma_{01} \cdot BEDINGUNG_i + \upsilon_{0i}$$

$$\beta_{1i} = \gamma_{10} + \gamma_{11} \cdot BEDINGUNG_i + \upsilon_{1i}$$

Gesamtmodell:

$$WB_{mi} = \gamma_{00} + \gamma_{10} \cdot WOCHE_{mi} + \gamma_{01} \cdot BEDINGUNG_i + +\gamma_{11} \cdot WOCHE_{mi} \cdot BEDINGUNG_i + \upsilon_{0i} + \upsilon_{1i} \cdot WO$$

> **!** Wichtig
>
> Jetzt muss der vollständige Datensatz `therapie` verwendet werden!

```r
slope.outcome <- lme(wohlbefinden ~ woche * bedingung,
  random = ~ woche | id,
  data = therapie, method = "ML"
)
summary(slope.outcome)
```

```
Linear mixed-effects model fit by maximum likelihood
  Data: therapie
       AIC      BIC    logLik
  1562.275 1596.941 -773.1373

Random effects:
 Formula: ~woche | id
 Structure: General positive-definite, Log-Cholesky parametrization
            StdDev    Corr
(Intercept) 1.0847725 (Intr)
woche       0.1388061 -0.26
Residual    0.7409710

Fixed effects:  wohlbefinden ~ woche * bedingung
                                  Value  Std.Error  DF   t-value p-value
(Intercept)                    3.591048 0.18137152 476 19.799404  0.0000
woche                         -0.007404 0.03038093 476 -0.243695  0.8076
bedingungTherapiegruppe       -0.133607 0.26096310  83 -0.511975  0.6100
woche:bedingungTherapiegruppe  0.121185 0.04251483 476  2.850423  0.0046
 Correlation:
                               (Intr) woche  bdngnT
woche                          -0.393
bedingungTherapiegruppe        -0.695  0.273
woche:bedingungTherapiegruppe   0.281 -0.715 -0.400

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-3.17091724 -0.47081098  0.03428825  0.49678310  3.62043764

Number of Observations: 563
Number of Groups: 85
```

```
  VarCorr(slope.outcome)
```

```
id = pdLogChol(woche)
            Variance   StdDev    Corr
(Intercept) 1.17673129 1.0847725 (Intr)
woche       0.01926715 0.1388061 -0.26
Residual    0.54903808 0.7409710
```

Die Cross-Level-Interaktion ist signifikant positiv mit $\hat{\gamma}_{11} = 0.121$. In der Therapiegruppe ist die Steigung von `woche` - die lineare Zunahme des Wohlbefindens pro Woche - also um 0.121 höher als in der Kontrollgruppe.

Der Haupteffekt von `woche` ($\hat{\gamma}_{10} = -0.007$) ist jetzt der bedingte Effekt (simple slope) von `woche` in der Kontrollgruppe. In der Kontrollgruppe gibt es daher keine signifikante Zunahme des Wohlbefindens über den Therapieverlauf. Der Haupteffekt von Bedingung ($\hat{\gamma}_{01} = -0.134$) ist jetzt der Unterschied im Wohlbefinden zwischen den beiden Gruppen bei der ersten Messung ($woche = 0$). Der Unterschied zu Beginn war also nicht signifikant.

Welcher Anteil der Varianz des linearen Trendeffekts von `woche` wird durch `bedingung` erklärt? Oder um es anders auszudrücken: Um welchen Anteil reduziert sich $\hat{\sigma}^2_{v_1}$ im Vergleich zu einem Modell ohne CLI-Effekt? Dazu muss zusätzlich ein Intercept-as-Outcome-Modell (Modell 4) berechnet werden, das sich nur dadurch von Modell 5 unterscheidet, dass es keine CLI enthält.

Intercept-as-Outcome Modell (Modell 4)

```
  intercept.outcome <- lme(wohlbefinden ~ woche + bedingung,
    random = ~ woche | id,
    data = therapie, method = "ML"
  )
  summary(intercept.outcome)
```

```
Linear mixed-effects model fit by maximum likelihood
  Data: therapie
       AIC      BIC    logLik
  1567.995 1598.328 -776.9973

Random effects:
 Formula: ~woche | id
 Structure: General positive-definite, Log-Cholesky parametrization
            StdDev    Corr
(Intercept) 1.0978840 (Intr)
```

```
woche       0.1528629 -0.296
Residual    0.7401163

Fixed effects:  wohlbefinden ~ woche + bedingung
                            Value Std.Error  DF   t-value p-value
(Intercept)              3.445564 0.1749099 477 19.699078  0.0000
woche                    0.054394 0.0224303 477  2.425020  0.0157
bedingungTherapiegruppe 0.165298 0.2391333  83  0.691236  0.4913
 Correlation:
                         (Intr) woche
woche                    -0.302
bedingungTherapiegruppe -0.659 -0.020

Standardized Within-Group Residuals:
         Min           Q1          Med           Q3          Max
-3.126104779 -0.466024529  0.007887565  0.492033382  3.585481939

Number of Observations: 563
Number of Groups: 85
```

```
VarCorr(intercept.outcome)
```

```
id = pdLogChol(woche)
            Variance   StdDev    Corr
(Intercept) 1.20534930 1.0978840 (Intr)
woche       0.02336708 0.1528629 -0.296
Residual    0.54777210 0.7401163
```

Anteil der durch `bedingung` erklärten Slope-Varianz:

$R^2_{Cross-level} = \frac{\hat{\sigma}^2_{v_1 4} - \hat{\sigma}^2_{v_1 5}}{\hat{\sigma}^2_{v_1 4}} = \frac{0.0234 - 0.0193}{0.0234} = 0.1752$

$\Rightarrow$ 17.52 % der Varianz zwischen den Personen in Bezug auf den linearen Trend (Effekt von `woche`) können durch die Zugehörigkeit zur Therapie- versus Kontrollgruppe erklärt werden.

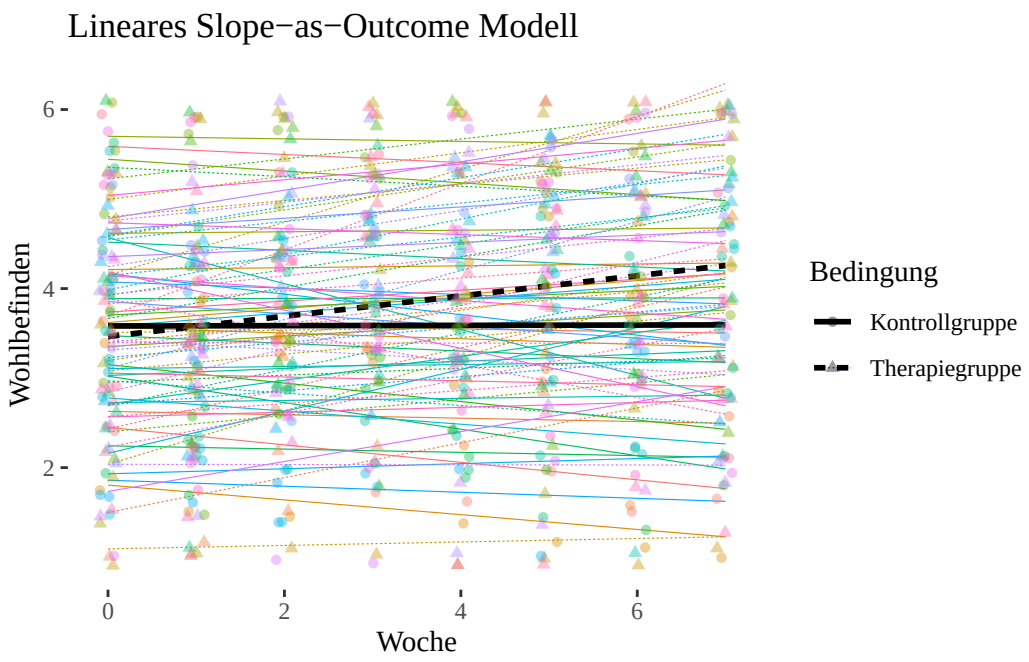
Zu guter Letzt können wir noch überprüfen, ob die im Modell 5 verbliebene Slope-Varianz $\hat{\sigma}^2_{v_1} = 0.0193$ noch signifikant ist. Den entsprechenden Modellvergleich erhält man über `ranova()`. Diese Funktion ist aber nur für mit `lmer()` aus `lme4` gefittete Modelle verfügbar. Bei den linearen Trendmodellen macht es eigentlich keinen Unterschied, welche der beiden Funktionen/Packages wir verwenden. Daher fitten wir Modell 5 für diesen Zweck einfach nochmals mit `lmer()` und wenden `ranova()` dann auf dieses Modell an:

```r
slope.outcome2 <- lmer(wohlbefinden ~ woche * bedingung + (woche | id),
  data = therapie, REML = FALSE
)
ranova(slope.outcome2)
```

```
ANOVA-like table for random-effects: Single term deletions

Model:
wohlbefinden ~ woche + bedingung + (woche | id) + woche:bedingung
                     npar  logLik    AIC    LRT Df Pr(>Chisq)
<none>                  8 -773.14 1562.3
woche in (woche | id)   6 -785.45 1582.9 24.619  2  4.509e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Der Vergleich zeigt, dass auch nach Berücksichtigung der Gruppenzugehörigkeit eine signifikante Zwischen-Personen-Varianz des linearen Trends verbleibt. Auf die Berechnung des exakten p-Werts für diesen Vergleich verzichten wir an dieser Stelle, da der Vergleich auf jeden Fall hochsignifikant ist.

```r
therapie$slope.outcome.preds <- predict(slope.outcome)

ggplot(aes(x = woche, y = slope.outcome.preds, group = id), data = therapie) +
  geom_smooth(aes(linetype = bedingung, color = id), method = "lm", se = FALSE, fullrange
  geom_smooth(aes(group = bedingung, linetype = bedingung), method = "lm", se = FALSE, ful
  ggtitle("Lineares Slope-as-Outcome Modell") +
  guides(color = FALSE) + # Color legend removed
  geom_jitter(aes(y = wohlbefinden, shape = bedingung, color = id), alpha = .4, width = 0.
  labs(linetype = "Bedingung", shape = "Bedingung") +
  xlab("Woche") +
  ylab("Wohlbefinden") +
  theme_tufte()
```



3.1.3. Zusammenfassung

Im Intercept-Only-Modell (Modell 1) der Therapiegruppe waren 57 % der Varianz der abhängigen Variablen `wohlbefinden` auf Unterschiede zwischen den Personen zurückzuführen. Auch wenn wir die Level-2-Varianz hier nicht auf Signifikanz getestet haben, ist offensichtlich, dass sie substantiell ist und wir daher den Personenfaktor berücksichtigen müssen (etwas anderes wäre auch nicht zu erwarten - Varianz zwischen Personen in Bezug auf ein psychologisches Merkmal gibt es so gut wie immer).

Im Random-Intercept-Modell (Modell 2) der Therapiegruppe sahen wir, dass der Effekt von `woche` signifikant positiv war, dass es also in dieser Gruppe einen konstanten Anstieg des Wohlbefindens gab. Allerdings konnte dieser lineare Trend nur knapp 10 % der Varianz des Wohlbefindens zwischen den Messzeitpunkten (innerhalb der Personen) erklären. In Folgeanalysen wäre es sinnvoll, die Unterschiede zwischen den Messzeitpunkten mit Hilfe von kontrastkodierten Variablen genauer zu analysieren (s.u.).

Über einen Vergleich von Random-Coefficients-Modell (Modell 3) und Random-Intercept-Modell (Modell 2) konnten wir feststellen, dass sich der lineare Anstieg des Wohlbefindens zwischen den Personen signifikant unterscheidet, dass also manche Personen einen stärkeren, andere einen schwächeren (bzw. manche sogar negativen) linearen Trend des Wohlbefindens aufwiesen.

Im Modell mit Cross-Level-Interaktion (Modell 5) haben wir zuletzt mit allen Daten (inkl. Kontrollgruppe) untersucht, ob sich in der Therapiegruppe ein stärkerer Anstieg des Wohlbefindens ergab. Die signifikante und vom Vorzeichen her positive CLI zeigte uns, dass genau dies der Fall war. Ein substantieller Anteil (17.5 %) der Unterschiedlichkeit der Verläufe zwischen den Personen war auf die Gruppenzugehörigkeit zurückzuführen. Diesen Wert erhielten wir aus dem Vergleich der Slope-Varianzen aus Modell 4 (Intercept-as-Outcome-Modell) und Modell 5.

3.2. Kontrastmodelle | Psychotherapie und Wohlbefinden

In Kontrastmodellen werden die Veränderungen über die Messzeitpunkte statt über einen linearen Trend direkt als (Mittelwerts-)Unterschiede zwischen den Messzeitpunkten mit Hilfe von Codiervariablen (hier: Dummy-Codierung) modelliert. Es handelt sich also um eine Art Varianzanalyse mit Messwiederholung, mit dem HLM-Kontrast-Modell sind aber weniger strenge Annahmen in Bezug auf die Varianz-Kovarianzstruktur verbunden als in der Varianzanalyse mit Messwiederholung (dort: Compound Symmetry bzw. Sphärizität). Für weitere Informationen siehe die Vorlesungsunterlagen.

Wir betrachten ein Kontrastmodell mit 6 Messzeitpunkten (Wochen 0-5). Das Modell mit allen 8 Messzeitpunkten braucht wegen seiner Komplexität sehr viel Rechenzeit und ist daher für diese Übung ungeeignet.

Erstellen des Datensatzes nur für die Therapiegruppe beschränkt auf die Wochen 0-5 mit neuer Faktorvariable `woche_f`:

```
therapie05_gr1 <- therapie_gr1 |>
  filter(woche <= 5) |>
  mutate(woche_f = as.factor(woche))
```

Erstellen des Datensatzes für beide Gruppen beschränkt auf die Wochen 0-5 mit neuer Faktorvariable `woche_f`:

```
therapie05 <- therapie |>
  filter(woche <= 5) |>
  mutate(woche_f = as.factor(woche))
```

3.2.1. Level-1-Modelle (nur Therapiegruppe)

Bevor wir uns dem Random-Coefficients-Modell (Modell 3) als endgültigen Kontrastmodell zuwenden, wie es in Eid, Gollwitzer, und Schmitt (2017) als saturiertes Modell ohne Level-1-Residuum beschrieben ist, wollen wir zunächst ein Random-Intercept-Modell betrachten (inkl. Level-1-Residuum).

Dieses Modell dient einerseits der Testung des Gesamteffekts von `woche_f` (über einen Modellvergleich mit dem Intercept-Only-Modell) sowie der Überprüfung der Frage, ob überhaupt ein Kontrastmodell notwendig ist oder ob ggf. das oben besprochene lineare Trendmodell bzw. ein Modell mit polynomialen Trends, das weniger Parameter schätzt als das Kontrastmodell, ausreicht.

3.2.1.1. Random-Intercept-Modell (Modell 2)

```r
random.intercept.05.c <- lme(wohlbefinden ~ woche_f,
  random = ~ 1 | id,
  data = therapie05_gr1,
  method = "ML"
)
summary(random.intercept.05.c)
```

```
Linear mixed-effects model fit by maximum likelihood
  Data: therapie05_gr1
       AIC      BIC    logLik
  667.7816 695.1459 -325.8908

Random effects:
 Formula: ~1 | id
        (Intercept)  Residual
StdDev:    1.019026 0.8380026

Fixed effects:  wohlbefinden ~ woche_f
                Value Std.Error  DF   t-value p-value
(Intercept)  3.298903 0.2127571 180 15.505486  0.0000
woche_f1    -0.029438 0.1933249 180 -0.152271  0.8791
woche_f2     0.593702 0.1975901 180  3.004716  0.0030
woche_f3     0.987673 0.1998182 180  4.942858  0.0000
woche_f4     0.661877 0.1981677 180  3.339985  0.0010
woche_f5     0.713265 0.1965381 180  3.629141  0.0004
 Correlation:
         (Intr) wch_f1 wch_f2 wch_f3 wch_f4
woche_f1 -0.468
woche_f2 -0.456  0.502
woche_f3 -0.454  0.499  0.490
woche_f4 -0.458  0.503  0.494  0.495
woche_f5 -0.461  0.507  0.498  0.496  0.500

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-2.96644596 -0.54297993  0.04957161  0.53007502  3.08147650

Number of Observations: 226
Number of Groups: 41
```

```
VarCorr(random.intercept.05.c)
```

```
id = pdLogChol(1)
            Variance  StdDev
(Intercept) 1.0384137 1.0190259
Residual    0.7022484 0.8380026
```

Die Ergebnisse zeigen einen nicht signifikanten Effekt der ersten Dummy-Variable (D_1) von $\hat{\gamma}_{10} = -0.029$. Zwischen der Baseline und Woche 1 zeigt sich in der Therapiegruppe also keine statistisch bedeutsame Veränderung des Wohlbefindens. Die Effekte der zweiten bis fünften Dummy-Variablen ($D_2 - D_5$) sind dagegen signifikant mit $\hat{\gamma}_{20} = 0.594, \hat{\gamma}_{30} = 0.988, \hat{\gamma}_{40} = 0.662, \hat{\gamma}_{50} = 0.713$. Von Woche 0 zu den Wochen 2, 3, 4 und 5 zeigt sich also jeweils eine statistisch bedeutsame Zunahme des Wohlbefindens.

Gibt es auch einen signifikanten Gesamteffekt? Dazu müssen wir das Random-Intercept-Modell mit einem Modell ohne Prädiktoren vergleichen, also einem Intercept-Only-Modell:

```
intercept.only.05 <- lme(wohlbefinden ~ 1,
  random = ~ 1 | id,
  data = therapie05_gr1,
  method = "ML"
)

anova(random.intercept.05.c, intercept.only.05)
```

```
                     Model df      AIC      BIC    logLik   Test  L.Ratio
random.intercept.05.c     1  8 667.7816 695.1459 -325.8908
intercept.only.05         2  3 697.8729 708.1345 -345.9365 1 vs 2 40.09131
                     p-value
random.intercept.05.c
intercept.only.05      <.0001
```

Ja, auch insgesamt zeigen sich Unterschiede zwischen den verschiedenen Messzeitpunkten in der Variable `wohlbefinden`.

3.2.1.1.1. * Vergleich mit dem linearen Modell

Es könnte ja sein, dass die oben berechneten linearen Modelle ausreichen, um die Veränderung des Wohlbefindens über die Zeit zu modellieren. Das können wir über einen Modellvergleich des RI-Kontrast-Modells mit dem RI-linearen Modell herausfinden. Das lineare Modell haben

wir oben für alle 8 Messzeitpunkte (0-7) berechnet. Wir müssen es daher nochmal beschränkt auf die ersten sechs Messzeitpunkte berechnen, um den Vergleich durchführen zu können.

```
random.intercept.05.l <- lme(wohlbefinden ~ woche,
  random = ~ 1 | id,
  data = therapie05_gr1,
  method = "ML"
)

anova(random.intercept.05.c, random.intercept.05.l)
```

```
                     Model df      AIC      BIC    logLik   Test  L.Ratio
random.intercept.05.c     1  8 667.7816 695.1459 -325.8908
random.intercept.05.l     2  4 676.1171 689.7992 -334.0586 1 vs 2 16.33551
                     p-value
random.intercept.05.c
random.intercept.05.l  0.0026
```

Das lineare Modell passt signifikant schlechter auf die Daten als das Kontrast-Modell. Somit ist es notwendig, die nicht-linearen Anteile des Verlaufs zu berücksichtigen.

i Vertiefung: Vergleich mit polynomialen Trendmodellen

Man kann jetzt noch überprüfen, ob ggf. ein polynomiales Trendmodell (mit quadratischem, kubischem oder quartischem) Effekt ausreicht, man also die Nicht-Linearität des Verlaufs nicht unbedingt mit dem Kontrastmodell beschreiben muss, sondern auf ein (etwas) sparsameres Modell ausweichen kann.
Um das Kontrastmodell mit den verschiedenen polynomialen Trendmodellen vergleichen zu können, müssen diese zunächst geschätzt werden. Da wir uns hier nicht für die einzelnen Trendeffekte im Detail interessieren, verzichten wir auf die Ausgabe der geschätzten Parameter dieser Modelle mit `summary()`. Stattdessen führen wir gleich die Modellvergleiche mit dem Kontrastmodell durch:

```r
random.intercept.05.quad <- lme(wohlbefinden ~ woche + I(woche^2),
  random = ~ 1 | id,
  data = therapie05_gr1,
  method = "ML"
)

random.intercept.05.cub <- lme(wohlbefinden ~ woche + I(woche^2) + I(woche^3),
  random = ~ 1 | id,
  data = therapie05_gr1,
  method = "ML"
)

random.intercept.05.quart <- lme(wohlbefinden ~ woche + I(woche^2) + I(woche^3) + I(woch
  random = ~ 1 | id,
  data = therapie05_gr1,
  method = "ML"
)


anova(random.intercept.05.c, random.intercept.05.quart, random.intercept.05.cub, random.
```

```
                        Model df      AIC      BIC    logLik   Test  L.Ratio
random.intercept.05.c       1  8 667.7816 695.1459 -325.8908
random.intercept.05.quart   2  7 666.0751 690.0189 -326.0376 1 vs 2 0.293525
random.intercept.05.cub     3  6 671.3163 691.8395 -329.6582 2 vs 3 7.241181
random.intercept.05.quad    4  5 671.7553 688.8580 -330.8777 3 vs 4 2.439023
random.intercept.05.l       5  4 676.1171 689.7992 -334.0586 4 vs 5 6.361775
                        p-value
random.intercept.05.c
random.intercept.05.quart  0.5880
random.intercept.05.cub    0.0071
random.intercept.05.quad   0.1183
random.intercept.05.l      0.0117
```

Der hierarchische Modellvergleich der fünf Modelle (Kontrastmodell, Modell mit quartischem Trend `I(woche^4)`, Modell mit kubischem Trend `I(woche^3)`, Modell mit quadratischem Trend `I(woche^2)`, Modell mit linearem Trend `woche`) zeigt, dass sich der Model-Fit (Devianz) des Kontrastmodells nicht signifikant von dem des quartischen Trendmodells unterscheidet ($p = 0.5880$), sich aber das letztere signifikant vom kubischen Trendmodell unterscheidet ($p = 0.0071$). Das bedeutet, dass für eine angemessene Beschreibung des Verlaufs des Wohlbefindens (mindestens) ein quartisches Trendmodell (d.h. ein Modell

mit mindestens drei Richtungsänderungen) benötigt wird. Dass der folgende Modellvergleich zwischen dem kubischen Trendmodell und dem quadratischen Trendmodell dann wieder nicht signifikant ist ($p = 0.1183$), spielt keine Rolle mehr, da wir beim ersten signifikanten Modellvergleich (quartisch vs. kubisch) stoppen müssen, da das kubische Trendmodell schon keine ausreichende Passung mehr aufweist.
Für unsere Zwecke ist es nicht sonderlich relevant, dass wir streng genommen kein Kontrastmodell benötigen, sondern ein quartisches polynomiales Trendmodell (mit einem Parameter weniger) ausreicht, da dieses nur geringfügig weniger komplex und zudem schwieriger zu interpretieren ist als das Kontrastmodell.

3.2.1.2. Random-Coefficients-Modell (Modell 3)

Bei den Kontrastmodellen spielt das Random-Coefficients-Modell eine besondere Rolle, da es sich bei diesem Modell um ein *saturiertes Modell* handelt, d.h. um ein Modell ohne Level-1-Residuum, da auf Ebene 1 alle Variation der Beobachtungen durch die Modellparameter (fixed und random effects) repräsentiert ist. Im Lehrbuch von Eid, Gollwitzer, und Schmitt (2017) wird im Abschnitt zu den Kontrastmodellen nur dieses Random-Coefficients-Modell beschrieben, ein Random-Intercept-Modell wie wir es oben beschrieben haben, wird dort gar nicht erwähnt.

Da die `lme()`-Funktion - wenn wir sie nicht daran hindern - auch in diesem Modell versucht, eine Level-1-Residualvarianz zu schätzen, müssen wir die `lme()`-Modellsyntax folgendermassen mit einem `control`-Argument ergänzen:

```
control = list(opt = "optim", sigma = 1e-7)
```

Hier wird zum einen ein Optimizer für den Schätzalgorithmus namens `optim` festgelegt, und zum anderen `sigma`, d.h. die Level-1-Residual-Standardabweichung (Quadratwurzel aus der Level-1-Residualvarianz) auf einen sehr kleinen, d.h. sehr nahe bei 0 liegenden Wert festgesetzt (eine Festsetzung auf exakt 0 ist nicht möglich).

```
random.coefficients.05.c <- lme(wohlbefinden ~ woche_f,
  random = ~ woche_f | id,
  data = therapie05_gr1,
  method = "ML",
  control = list(opt = "optim", sigma = 1e-7)
)
summary(random.coefficients.05.c)
```

```
Linear mixed-effects model fit by maximum likelihood
  Data: therapie05_gr1
        AIC       BIC   logLik
```

```
  -6442.107 -6349.752 3248.053

Random effects:
 Formula: ~woche_f | id
 Structure: General positive-definite, Log-Cholesky parametrization
            StdDev    Corr
(Intercept) 1.2845973 (Intr) wch_f1 wch_f2 wch_f3 wch_f4
woche_f1    0.9129692 -0.297
woche_f2    0.8557933 -0.379  0.223
woche_f3    1.2249202 -0.474  0.244  0.592
woche_f4    1.4606758 -0.515  0.432  0.416  0.595
woche_f5    1.3621709 -0.483  0.165  0.314  0.643  0.859
Residual    0.0000001

Fixed effects:  wohlbefinden ~ woche_f
                Value Std.Error  DF   t-value p-value
(Intercept)  3.337688 0.2052587 180 16.260884  0.0000
woche_f1    -0.059310 0.1484144 180 -0.399627  0.6899
woche_f2     0.618576 0.1420488 180  4.354675  0.0000
woche_f3     0.921607 0.2020530 180  4.561215  0.0000
woche_f4     0.589726 0.2361702 180  2.497038  0.0134
woche_f5     0.694147 0.2205539 180  3.147289  0.0019
 Correlation:
         (Intr) wch_f1 wch_f2 wch_f3 wch_f4
woche_f1 -0.313
woche_f2 -0.379  0.236
woche_f3 -0.469  0.255  0.570
woche_f4 -0.515  0.434  0.410  0.582
woche_f5 -0.486  0.183  0.316  0.630  0.851

Standardized Within-Group Residuals:
          Min            Q1           Med            Q3           Max
-4.884981e-07 -9.270362e-08  1.332268e-08  1.043610e-07  4.618528e-07

Number of Observations: 226
Number of Groups: 41
```

```
VarCorr(random.coefficients.05.c)
```

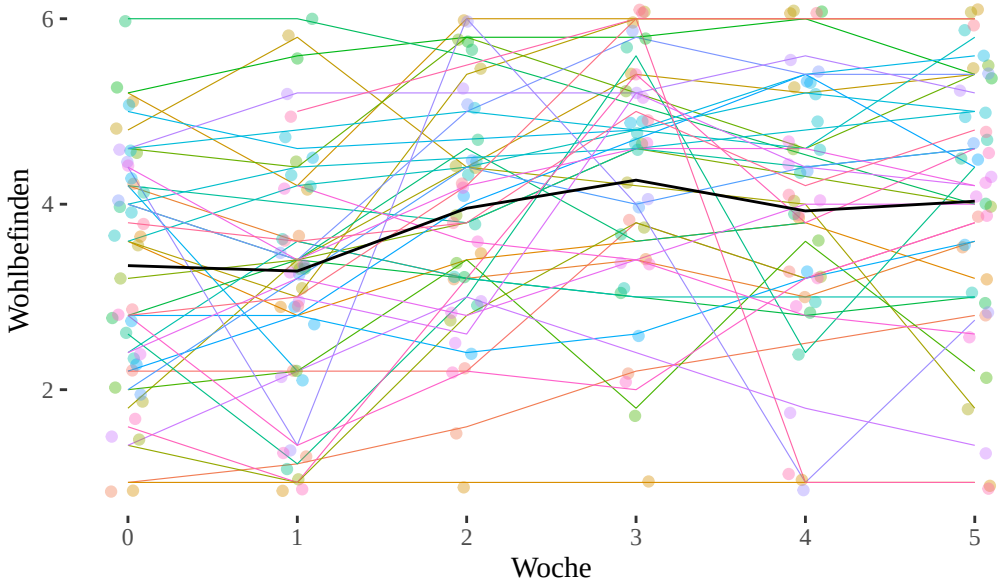
```
id = pdLogChol(woche_f)
            Variance     StdDev    Corr
```

```
(Intercept) 1.650190e+00 1.2845973 (Intr) wch_f1 wch_f2 wch_f3 wch_f4
woche_f1    8.335127e-01 0.9129692 -0.297
woche_f2    7.323822e-01 0.8557933 -0.379  0.223
woche_f3    1.500429e+00 1.2249202 -0.474  0.244  0.592
woche_f4    2.133574e+00 1.4606758 -0.515  0.432  0.416  0.595
woche_f5    1.855509e+00 1.3621709 -0.483  0.165  0.314  0.643  0.859
Residual    1.000000e-14 0.0000001
```

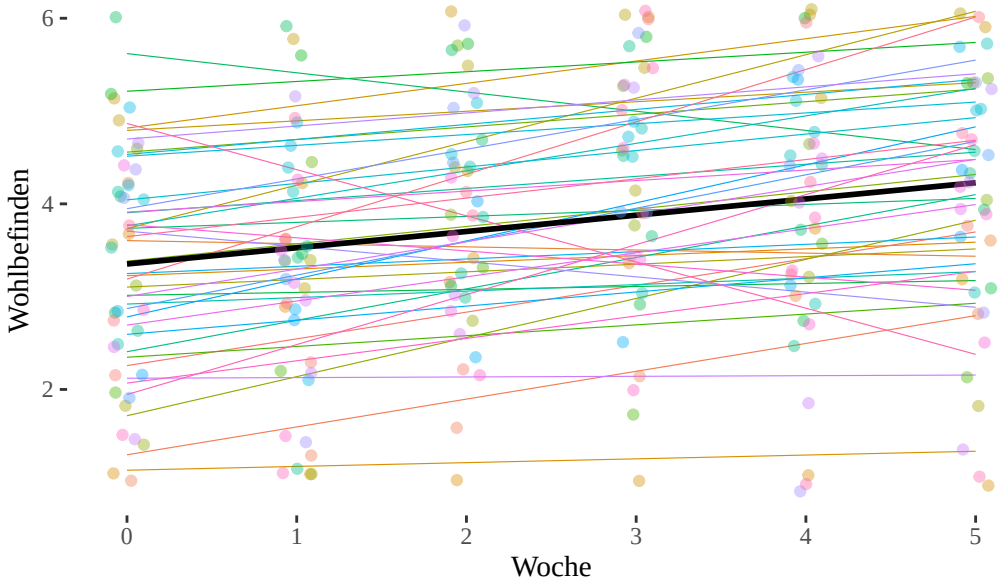
Die fixed effects Parameter (Effekte der Dummy-Variablen `woche_f1` bis `woche_f5`) des Random-Coefficents-Modells unterscheiden sich nur geringfügig von denen des Random-Intercept-Modells weiter oben, auch hier gibt es signifikante Effekte der Dummy-Variablen `woche_f2` bis `woche_f5`, während der Wohlbefindensunterschied zwischen Woche 0 und Woche 1 (Effekt von `woche_f1`) nicht signifikant ist.

Zusätzlich bekommen wir die Random-Effects-Varianzen und Korrelationen der Random Intercept/Slopes der Dummy-Variablen ausgegeben. Es zeigt sich, dass die Varianzen der Woche-Effekte mit Zunahme des Abstandes von der Baseline tendenziell grösser werden, also dass die interindividuellen Unterschiede in der Veränderung des Wohlbefindens (d.h. im Vergleich zum Wohlbefinden zu Beginn) im Verlauf der Therapie grösser wurden. Von Interesse sind insbesondere die Korrelationen: In der ersten Spalte von `Corr` erhalten wir die Korrelationen zwischen Intercept und den Dummy-Effekten, und diese sind alle negativ und werden mit weiter entfernten `woche_f` Effekten tendenziell grösser. Das bedeutet, dass Personen, die in Woche 0 mit einem relativ hohen Wohlbefinden in die Therapie starten, tendenziell kleinere Wohlbefindenszunahmen aufweisen als Personen, die mit relativ niedrigerem Wohlbefinden starten. Die Korrelationen der Woche-Effekte untereinander sind dagegen alle positiv und näher zusammen liegende Wochen-Effekte sind tendenziell stärker miteinander korreliert. Das spiegelt die Tatsache wider, das Personen, die eine z.B. starke Zunahme des Wohlbefindens von Woche 0 (Baseline) nach Woche 2 (Effekt von `woche_f2`) erleben, auch eine (relativ zu anderen) starke Zunahme von Woche 0 nach Woche 3 (Effekt von `woche_f3`) erleben ($r_{v_2 v_3} = 0.592$).

Kategoriales Random−Coefficients−Modell (nur Therapiegruppe)



Lineares Random−Coefficients−Modell (nur Therapiegruppe)



3.2.2. Level-2-Modelle (Therapie- und Kontrollgruppe)

3.2.2.1. Slope-as-Outcome-Modell (Modell 5)

Wir starten gleich mit dem Slope-as-Outcome-Modell, da dieses Modell alles enthält, wofür wir uns interessieren, wenn wir beide Gruppen (Therapie- und Kontrollgruppe) analysieren.

Es handelt sich hier auch wieder um ein Modell mit allen Random Effects, d.h. wie oben um ein saturiertes Modell ohne Level-1-Residuum. Die in diesem Modell enthaltene Cross-Level-Interaktion testet die Unterschiedlichkeit der Dummy-Variablen-Effekte zwischen Therapie- und Kontrollgruppe. Ein positiver CLI-Term bedeutet jeweils, dass der entsprechende Effekt in der Therapiegruppe stärker (positiv) ist als in der Kontrollgruppe.

Dieses Modell muss jetzt mit dem vollständigen Datensatz (alle Gruppen) der Wochen 0-5 geschätzt werden:

```r
slope.outcome.05.c <- lme(wohlbefinden ~ woche_f * bedingung,
  random = ~ woche_f | id,
  data = therapie05,
  method = "ML",
  control = list(opt = "optim", sigma = 1e-7)
)
summary(slope.outcome.05.c)
```

```
Linear mixed-effects model fit by maximum likelihood
  Data: therapie05
        AIC       BIC   logLik
  -12590.83 -12455.89 6328.416

Random effects:
 Formula: ~woche_f | id
 Structure: General positive-definite, Log-Cholesky parametrization
            StdDev    Corr
(Intercept) 1.3504717 (Intr) wch_f1 wch_f2 wch_f3 wch_f4
woche_f1    0.9804853 -0.410
woche_f2    0.8960763 -0.440  0.412
woche_f3    1.0762727 -0.453  0.341  0.546
woche_f4    1.4344046 -0.554  0.672  0.556  0.626
woche_f5    1.4885280 -0.555  0.439  0.422  0.672  0.820
Residual    0.0000001

Fixed effects:  wohlbefinden ~ woche_f * bedingung
                                    Value Std.Error  DF   t-value p-value
```

```
(Intercept)                          3.829845 0.2071566 346 18.487684  0.0000
woche_f1                            -0.479391 0.1523475 346 -3.146695  0.0018
woche_f2                            -0.334966 0.1450746 346 -2.308926  0.0215
woche_f3                            -0.204510 0.1823110 346 -1.121766  0.2627
woche_f4                            -0.290397 0.2344358 346 -1.238705  0.2163
woche_f5                            -0.114591 0.2505564 346 -0.457347  0.6477
bedingungTherapiegruppe             -0.433488 0.2994563  83 -1.447583  0.1515
woche_f1:bedingungTherapiegruppe     0.357381 0.2206313 346  1.619813  0.1062
woche_f2:bedingungTherapiegruppe     0.886971 0.2082029 346  4.260127  0.0000
woche_f3:bedingungTherapiegruppe     1.075708 0.2559371 346  4.203018  0.0000
woche_f4:bedingungTherapiegruppe     0.821419 0.3300128 346  2.489052  0.0133
woche_f5:bedingungTherapiegruppe     0.732874 0.3478820 346  2.106674  0.0359
 Correlation:
                                    (Intr) wch_f1 wch_f2 wch_f3 wch_f4 wch_f5
woche_f1                            -0.411
woche_f2                            -0.425  0.403
woche_f3                            -0.412  0.313  0.480
woche_f4                            -0.522  0.636  0.510  0.547
woche_f5                            -0.506  0.401  0.371  0.571  0.775
bedingungTherapiegruppe             -0.692  0.285  0.294  0.285  0.361  0.350
woche_f1:bedingungTherapiegruppe     0.284 -0.691 -0.278 -0.216 -0.439 -0.277
woche_f2:bedingungTherapiegruppe     0.296 -0.281 -0.697 -0.334 -0.356 -0.259
woche_f3:bedingungTherapiegruppe     0.294 -0.223 -0.342 -0.712 -0.389 -0.406
woche_f4:bedingungTherapiegruppe     0.371 -0.451 -0.363 -0.388 -0.710 -0.550
woche_f5:bedingungTherapiegruppe     0.364 -0.289 -0.267 -0.411 -0.558 -0.720
                                    bdngnT wc_1:T wc_2:T wc_3:T wc_4:T
woche_f1
woche_f2
woche_f3
woche_f4
woche_f5
bedingungTherapiegruppe
woche_f1:bedingungTherapiegruppe    -0.418
woche_f2:bedingungTherapiegruppe    -0.433  0.411
woche_f3:bedingungTherapiegruppe    -0.432  0.333  0.506
woche_f4:bedingungTherapiegruppe    -0.539  0.651  0.528  0.579
woche_f5:bedingungTherapiegruppe    -0.530  0.423  0.395  0.612  0.792

Standardized Within-Group Residuals:
          Min            Q1           Med            Q3           Max
-4.840572e-07 -8.881784e-08  4.440892e-09  8.881784e-08  5.240253e-07

Number of Observations: 441
```

```
Number of Groups: 85
```

```
VarCorr(slope.outcome.05.c)
```

```
id = pdLogChol(woche_f)
            Variance     StdDev    Corr
(Intercept) 1.823774e+00 1.3504717 (Intr) wch_f1 wch_f2 wch_f3 wch_f4
woche_f1    9.613513e-01 0.9804853 -0.410
woche_f2    8.029528e-01 0.8960763 -0.440  0.412
woche_f3    1.158363e+00 1.0762727 -0.453  0.341  0.546
woche_f4    2.057517e+00 1.4344046 -0.554  0.672  0.556  0.626
woche_f5    2.215716e+00 1.4885280 -0.555  0.439  0.422  0.672  0.820
Residual    1.000000e-14 0.0000001
```

Die Ergebnisse zeigen, dass alle bis auf den ersten Interaktionsterm signifikant sind. Nur von Woche 0 zu Woche 1 (Interaktionsterm $D_1 \times Z : \hat{\gamma}_{11} = 0.357$) ist der Unterschied zwischen Therapie- und Kontrollgruppe nicht signifikant. Bei allen anderen Kontrasten mit der Baseline gibt es einen signifikanten Unterschied in der Veränderung von Wohlbefinden: Woche 2 (Interaktionsterm $D_2 \times Z : \hat{\gamma}_{21} = 0.887$), Woche 3 (Interaktionsterm $D_3 \times Z : \hat{\gamma}_{31} = 1.076$), Woche 4 (Interaktionsterm $D_4 \times Z : \hat{\gamma}_{41} = 0.821$), Woche 5 (Interaktionsterm $D_5 \times Z : \hat{\gamma}_{51} = 0.733$).

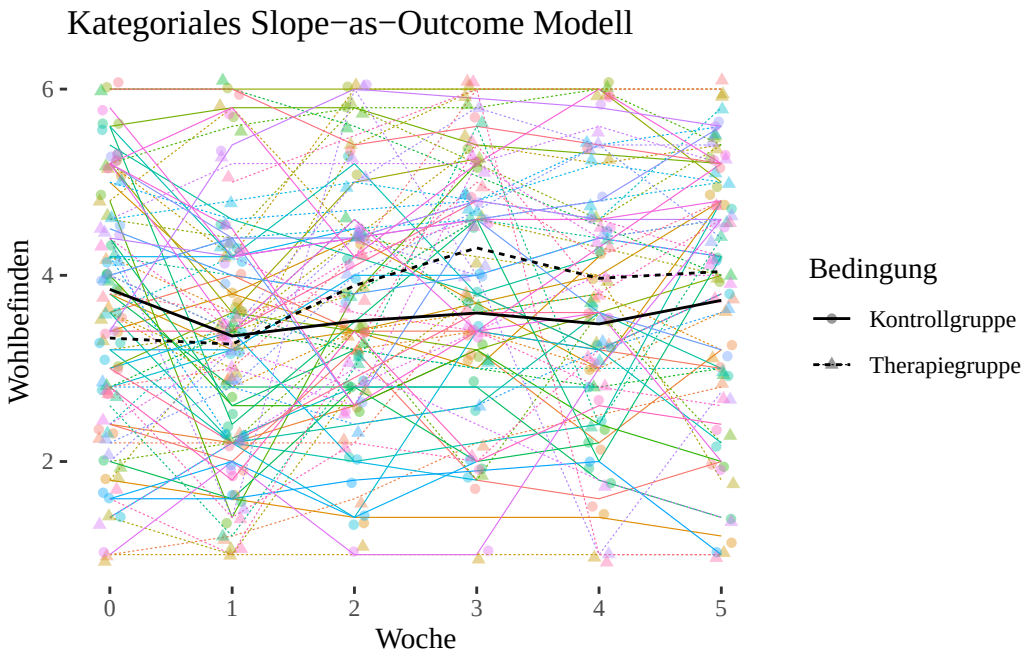
Interessant sind auch die simple slopes von `woche_f` (Effekte der Dummyvariablen $D_1 - D_5$): Diese testen die Veränderungen in der Kontrollgruppe (Referenzkategorie von `bedingung`). Man sieht, dass es in dieser Gruppe in den ersten beiden Wochen signifikant bergab mit dem Wohlbefinden geht, ehe es sich stabilisiert und ab Woche 3 keine signifikanten Kontraste mit der Baseline mehr zu beobachten sind.

Auch hier brauchen wir streng genommen noch einen Test für den Gesamteffekt der Interaktion (also aller fünf Interaktionsterme zusammen). Diesen bekommen wir über einen Vergleich mit einem Intercept-as-Outcome-Modell, das keine CLI enthält (aber sich darüber hinaus nicht von Modell 5 unterscheidet) :

```
intercept.outcome.05.c <- lme(wohlbefinden ~ woche_f + bedingung,
  random = ~ woche_f | id,
  data = therapie05,
  method = "ML",
  control = list(opt = "optim", sigma = 1e-7)
)

anova(slope.outcome.05.c, intercept.outcome.05.c)
```

```
                         Model df       AIC       BIC   logLik   Test  L.Ratio
slope.outcome.05.c           1 33 -12590.83 -12455.89 6328.416
intercept.outcome.05.c       2 28 -12740.81 -12626.31 6398.404 1 vs 2 139.9745
                         p-value
slope.outcome.05.c
intercept.outcome.05.c    <.0001
```

Wie zu erwarten war, ist die Cross-level-Interaktion auch insgesamt signifikant ($p < 0.0001$).



3.2.3. Zusammenfassung

Das RI-Kontrast-Modell (Modell 2) nur für die Therapiegruppe ergab signifikante Zunahmen des Wohlbefindens über die fünf Therapiewochen. Nur das nach der ersten Woche gemessene Wohlbefinden war nicht signifikant höher als das Baseline-Wohlbefinden, alle anderen Kontraste zur Baseline waren signifikant. Wegen der Dummy-Codierung können wir nur Aussagen zu den Baseline-Kontrasten machen, ob sich das Wohlbefinden auch zwischen den einzelnen Wochen (also z.B. von Woche 2 zu Woche 3) signifikant verbessert (bzw. ggf. wieder verschlechtert) hat, können wir mit dieser Codierung nicht feststellen. Um das zu testen, bräuchten wir eine Codierung im Sinne wiederholter Kontraste, die wir uns aber hier nicht näher anschauen. Ausserdem überprüften wir über einen Vergleich mit einem linearen RI-Modell, ob die Linearitätsannahme, die wir im ersten Teil dieser Übung getroffen hatten, angemessen war. Da das lineare Modell signifikant schlechter auf die Daten passte als das Kontrast-Modell, ist letzteres

zu bevorzugen. Die Veränderungen über den Therapieverlauf sind also nicht linear, sondern es gibt "Sprünge". Insbesondere scheint die Verbesserung des Wohlbefindens zwischen den Wochen 1 und 3 am stärksten zu sein (vgl. Plot). In einer Vertiefung betrachteten wir zudem noch polynomiale Trendmodelle: um den Verlauf des Wohlbefindens angemessen zu beschreiben, wird nach den Ergebnissen der Modellvergleiche mindestens ein Modell mit quartischem Trend benötigt, das im Sinne der Anzahl zu schätzender Parameter aber nur wenig "sparsamer" als das Kontrastmodell ist.

Das *saturierte* RC-Kontrast-Modell (Modell 3) hat uns zusätzlich zu den durchschnittlichen (=festen) Effekten des Woche-Faktors die Varianzen und Kovarianzen dieser Effekte geschätzt. Dort zeigten sich insbesondere negative Korrelationen des personenspezifischen Intercepts mit den Effekten der Wochen 1-5: Personen, die bereits mit einem höheren Wohlbefinden gestartet sind, zeigten tendenziell eine niedrigere Wohlbefindensverbesserung als Personen, die zu Beginn ein niedrigeres Wohlbefinden aufwiesen. Die Effekte der Wochen 1-5 waren dagegen allesamt positiv korreliert.

Der Hauptfokus dieser Analysen lag aber auf dem Slope-as-Outcome-Kontrast-Modell (Modell 5)), das mit dem vollständigen Datensatz (Therapie- und Kontrollgruppe) gerechnet wurde. Hier zeigten sich wie auch schon beim linearen Slope-as-Outcome-Modell bedeutsame Unterschiede zwischen Therapie- und Kontrollgruppe bzgl. der Veränderung des Wohlbefindens über den Therapieverlauf: Mit Ausnahme der Veränderung von der Baseline zur ersten Woche, zeigte sich bezüglich aller anderen Vergleiche mit der Baseline, dass die Veränderung (Verbesserung) des Wohlbefindens in der Therapiegruppe signifikant stärker war als in der Kontrollgruppe.

3.3. Übung

Wir verwenden (echte) Daten aus einem psychologischen Experiment mit Messwiederholung (Beispiel angepasst aus Bates 2010). Die Studie untersuchte den Effekt von Schlafentzug/Schlafmangel auf die Reaktionszeit von Lastwagenfahrerinnen und Lastwagenfahrern (Belenky u. a. 2003). Die Reaktionszeit wurde jeden Tag (über 10 Tage hinweg) getestet. Die Daten, die wir haben, sind ausschliesslich von der Gruppe, die pro Tag nur 3 Stunden schlafen durften. Tatsächlich sind die Daten im Package `lme4` gespeichert. Das bedeutet, wir können das Datenfile `sleepstudy` direkt verwenden, sobald das Package `lme4` geladen ist.

Wir können uns zuerst mal die Daten anschauen.

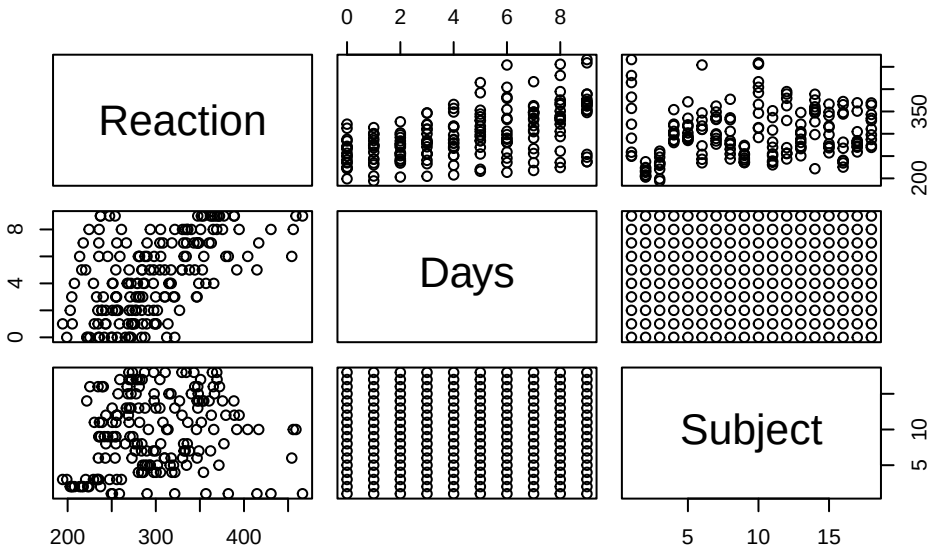
```
pacman::p_load(lme4, tidyverse, nlme, viridis)
str(sleepstudy)
```

```
'data.frame':   180 obs. of  3 variables:
 $ Reaction: num  250 259 251 321 357 ...
 $ Days    : num  0 1 2 3 4 5 6 7 8 9 ...
```

```
 $ Subject : Factor w/ 18 levels "308","309","310",..: 1 1 1 1 1 1 1 1 1 1 ...
```

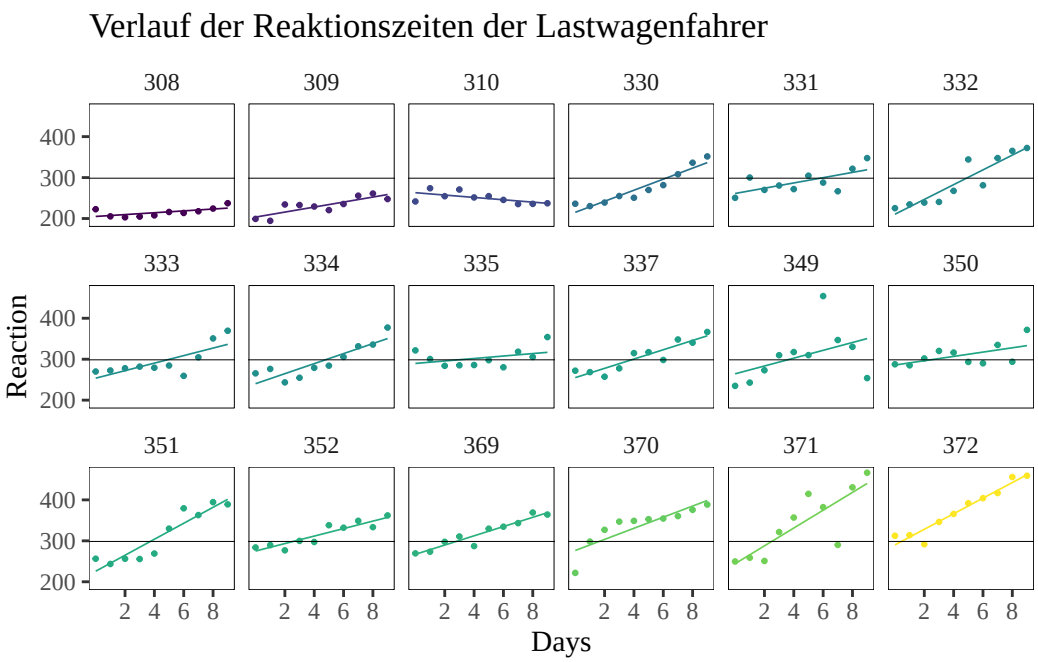
Wir haben insgesamt 18 Lastwagenfahrerinnen und Lastwagenfahrer `(Subject)`, jeweils den Tag der Testung `(Days, 1-10)` und die Reaktionszeit `(Reaction)` für jede Lastwagenfahrerin bzw. Lastwagenfahrer an jedem Tag (insgesamt 180 Beobachtungen). In diesem Beispiel gibt es keine Level-2-Prädiktoren.

```
plot(sleepstudy)
```



Dieser sehr basale Plot gibt eine Streudiagramm-Matrix aller Variablen aus. Der Streudiagramm zum Zusammenhang zwischen `Days` und `Reaction` berücksichtigt die Gruppierung der Werte innerhalb der einzelnen Lastwagenfahrer nicht. Und das Streudiagramm für `Days` und `Subject` ist natürlich komplett sinnfrei…

Wir versuchen daher mal etwas anderes: Jede(r) Lastwagenfahrer(in) bekommt ihren/seinen eigenen Plot für den Zusammenhang von `Days` und `Reaction`.



Die Plots sind nach Intercept und Slope geordnet geordnet. Die Farben visualisieren die Höhe der Intercepts.

1. Intraklassenkorrelation

Berechnen Sie die Intraklassenkorrelation. Tipp: Berechnen Sie ein Intercept-Only-Modell.

Lösung

```r
sleep.intercept.only <- lme(Reaction ~ 1,
  random = ~ 1 | Subject,
  data = sleepstudy, method = "ML"
)
VarCorr(sleep.intercept.only)
```

```
Subject = pdLogChol(1)
            Variance StdDev
(Intercept) 1196.436 34.58954
Residual    1958.865 44.25907
```

Die Berechnung erfolgt analog der der Intraklassenkorrelation im Therapie-Beispiel auf Basis des Intercept-Only-Modells.
Also:

$\frac{1196.436}{(1196.436+1958.865)} = 0.3792$
Das bedeutet, dass 38 % der gesamten Varianz der Reaktionszeiten auf (durchschnittliche) Unterschiede zwischen den Personen zurückzuführen sind und im Gegenzug 62 % der Gesamtvarianz innerhalb der Personen zu verorten sind.

2. Lineare Trendmodelle

Schätzen Sie den linearen Trend für die Population der Lastwagenfahrer und begründen Sie, weshalb Sie sich für das Random-Intercept oder für das Random-Coefficients Modell entschieden haben.

Lösung

```
sleep.random.intercept <- lme(Reaction ~ Days,
  random = ~ 1 | Subject,
  data = sleepstudy, method = "ML"
)
sleep.random.coefficients <- lme(Reaction ~ Days,
  random = ~ Days | Subject,
  data = sleepstudy, method = "ML"
)
```

```
anova(sleep.random.intercept, sleep.random.coefficients)
```

```
                         Model df      AIC      BIC    logLik   Test L.Ratio
sleep.random.intercept       1  4 1802.079 1814.851 -897.0393
sleep.random.coefficients    2  6 1763.939 1783.097 -875.9697 1 vs 2 42.1393
                         p-value
sleep.random.intercept
sleep.random.coefficients  <.0001
```

Wir haben beide Modelle geschätzt und sie danach mittels LR-Test verglichen (Signifikanztest für Slope-Varianz; auf die Berechnung des p-Wert mittels Mischverteilung verzichten wir hier). Es hat sich gezeigt, dass der Modell-Fit sich verbessert, wenn wir auch die Slopes als Random Effects modellieren, d.h. die Slope-Varianz ist signifikant. Daher sehen wir uns im Folgenden nur die Ergebnisse des Random-Coefficient-Modells im Detail an:

```
summary(sleep.random.coefficients)
```

```
Linear mixed-effects model fit by maximum likelihood
  Data: sleepstudy
       AIC      BIC    logLik
  1763.939 1783.097 -875.9697

Random effects:
 Formula: ~Days | Subject
 Structure: General positive-definite, Log-Cholesky parametrization
            StdDev    Corr
(Intercept) 23.780376 (Intr)
Days         5.716807 0.081
Residual    25.591842

Fixed effects:  Reaction ~ Days
                Value Std.Error  DF  t-value p-value
(Intercept) 251.40510  6.669396 161 37.69533       0
Days         10.46729  1.510647 161  6.92901       0
 Correlation:
     (Intr)
Days -0.138

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-3.94156355 -0.46559311  0.02894656  0.46361051  5.17933587

Number of Observations: 180
Number of Groups: 18
```

Und wenn wir uns die Modell-Parameter anschauen, ist der Fixed-Effekt Parameter für `Days` (linearer Trend) signifikant, $b_{Days} = 10.47, p < 0.001$. Wir können diesen Wert folgendermassen interpretieren: Jeder weitere Tag, an dem ein (zufällig ausgewählter) Lastwagenfahrer/Lastwagenfahrerin nur drei Stunden schläft, nimmt seine/ihre Reaktionszeit durchschnittlich um 10.47ms zu.

3. Kontrastmodelle

Wir möchten jetzt `Days` als Faktorvariable mit ins Modell nehmen. Das bedeutet, dass wir nicht mehr an einem linearen Effekt von `Days` interessiert sind. Stattdessen modellieren wir jeden einzelnen Tag separat. Dies ist flexibler, ein solches Modell benötigt aber mehr Parameter, was sich stark auf den Rechenaufwand für die Modellschätzung auswirkt (Schätzung braucht viel mehr Zeit).

Um den Prozess etwas abzukürzen, fassen wir die Daten ein wenig zusammen. Wir kreieren einen neuen Datenframe `sleepstudy_f`, der jeweils zwei Tage zusammenfasst. Insgesamt hat jede Person in `sleepstudy_f` nur noch fünf Einträge für `Reaction`, und zwar jeweils den Durchschnitt der ersten zwei Tage (Tag 0 und Tag 1), dann den Durchschnitt von Tag 2 und Tag 3, und so weiter.

Folgender Code-Chunk tut das für uns:

```
newdays <- paste(0:4 * 2, 1:5 * 2 - 1, sep = "-")[rep(1:5, each = 2)]
sleepstudy_f <- sleepstudy |>
  mutate(Days_f = factor(
    levels = c(newdays[1:5 * 2]),
    rep(newdays, times = length(sleepstudy$Days) / 10)
  )) |>
  group_by(Subject, Days_f) |>
  summarise(Reaction = mean(Reaction)) |>
  ungroup()

levels(sleepstudy_f$Days_f)
```

```
[1] "0-1" "2-3" "4-5" "6-7" "8-9"
```

```
sleepstudy_f
```

```
# A tibble: 90 x 3
   Subject Days_f Reaction
   <fct>   <fct>     <dbl>
 1 308     0-1        214.
 2 308     2-3        204.
 3 308     4-5        212.
 4 308     6-7        216.
 5 308     8-9        231.
 6 309     0-1        197.
 7 309     2-3        234.
 8 309     4-5        225.
 9 309     6-7        246.
10 309     8-9        254.
# i 80 more rows
```

Jetzt ist die Variable `Days_f` bereits als Faktor gespeichert. Wir können sie also direkt verwenden.

```r
# Hinzufügen der numerische Variable Days zum Datensatz
sleepstudy_f <- sleepstudy_f |>
  mutate(Days = as.numeric(Days_f) - 1) |>
  select(Subject, Days_f, Days, Reaction) # Reihenfolge anpassen

# Check...
sleepstudy_f
```

```
# A tibble: 90 x 4
   Subject Days_f  Days Reaction
   <fct>   <fct>  <dbl>    <dbl>
 1 308     0-1        0     214.
 2 308     2-3        1     204.
 3 308     4-5        2     212.
 4 308     6-7        3     216.
 5 308     8-9        4     231.
 6 309     0-1        0     197.
 7 309     2-3        1     234.
 8 309     4-5        2     225.
 9 309     6-7        3     246.
10 309     8-9        4     254.
# i 80 more rows
```

a) Definieren Sie ein Random-Coefficients-Kontrast-Modell (mit `Days_f` als Faktor)

Lösung

```r
sleep.random.coefficients.contrast <- lme(Reaction ~ Days_f,
  random = ~ Days_f | Subject,
  sleepstudy_f, method = "ML",
  control = list(opt = "optim", sigma = 1e-7)
)
```

b) Interpretieren Sie die Ergebnisse.

Lösung

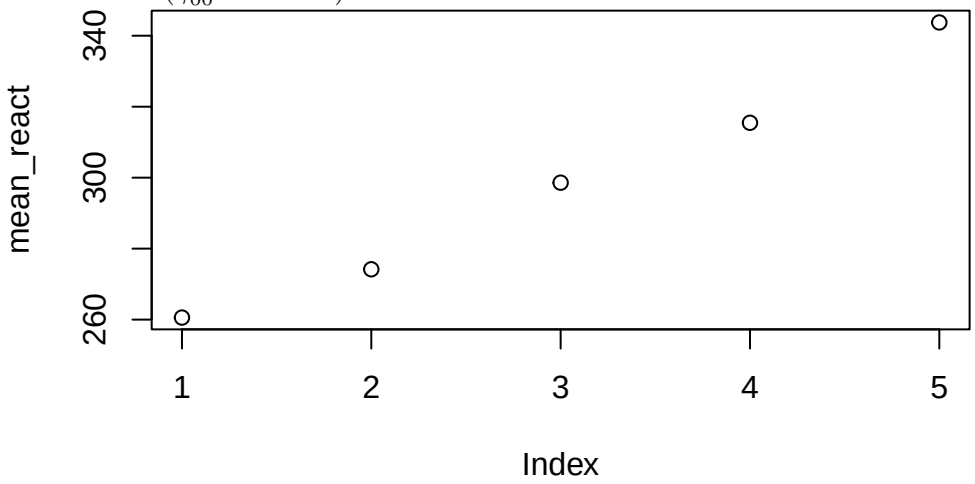
```r
summary(sleep.random.coefficients.contrast)
```

```
Linear mixed-effects model fit by maximum likelihood
```

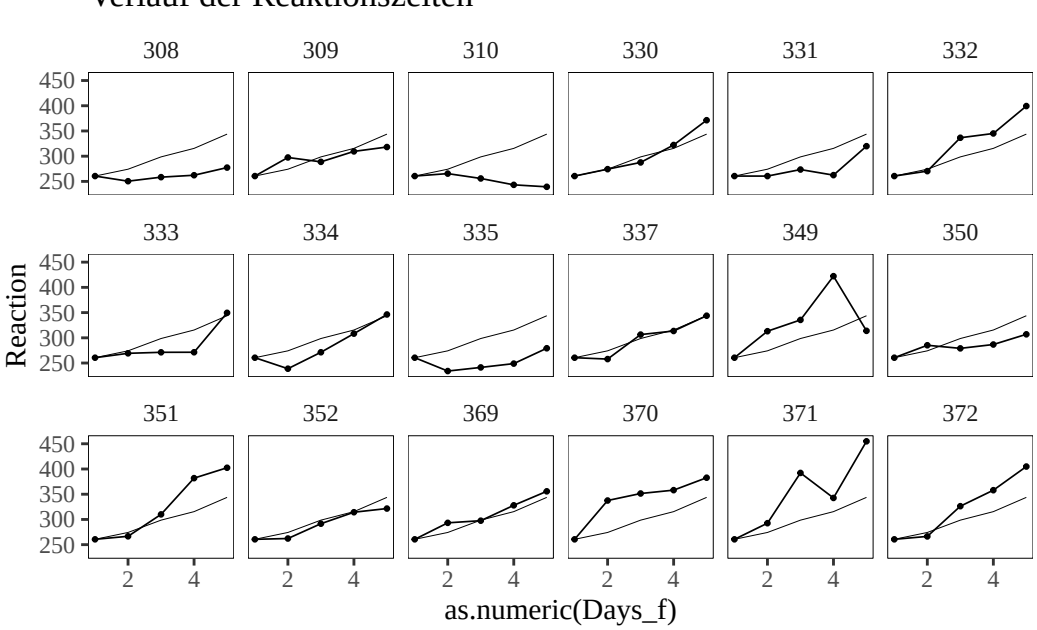
```
  Data: sleepstudy_f
        AIC      BIC   logLik
  -1847.586 -1797.59 943.7931

Random effects:
 Formula: ~Days_f | Subject
 Structure: General positive-definite, Log-Cholesky parametrization
            StdDev     Corr
(Intercept) 29.6864414 (Intr) Dy_2-3 Dy_4-5 Dy_6-7
Days_f2-3   24.8876950 -0.315
Days_f4-5   37.1239870 -0.141  0.638
Days_f6-7   46.7946065 -0.174  0.572  0.770
Days_f8-9   51.9289952  0.005  0.300  0.802  0.630
Residual     0.0000001

Fixed effects:  Reaction ~ Days_f
                Value Std.Error DF  t-value p-value
(Intercept) 260.57378  7.200020 68 36.19070  0.0000
Days_f2-3    13.60317  6.036153 68  2.25362  0.0274
Days_f4-5    38.01016  9.003889 68  4.22153  0.0001
Days_f6-7    54.89064 11.349359 68  4.83645  0.0000
Days_f8-9    83.16658 12.594631 68  6.60334  0.0000
 Correlation:
          (Intr) Dy_2-3 Dy_4-5 Dy_6-7
Days_f2-3 -0.315
Days_f4-5 -0.141  0.638
Days_f6-7 -0.174  0.572  0.770
Days_f8-9  0.005  0.300  0.802  0.630

Standardized Within-Group Residuals:
          Min             Q1            Med             Q3            Max
-2.842171e-06 -1.136868e-06 -5.684342e-07 -5.684342e-07  0.000000e+00

Number of Observations: 90
Number of Groups: 18
```

Die Effekte von `Days_f` kontrastieren die Faktorstufen `2-3`, `4-5`, `6-7`, `8-9` jeweils mit der Referenzkategorie `0-1`. Diese Effekte ($\hat{\gamma}_{10} = 13.6$, $\hat{\gamma}_{20} = 38.01$, $\hat{\gamma}_{30} = 54.89$, $\hat{\gamma}_{40} = 83.17$) sind allesamt signifikant und zeigen eine relativ gleichmässige Zunahme, was die Linearität des Zusammenhangs recht gut widerspiegelt.
Man kann sich das in etwa so vorstellen: Auf der Y-Achse ist jetzt die durchschnittliche Reaktionszeit `mean_react` pro Faktorstufe 1-5 dargestellt, beginnend mit dem Intercept

des Modells ($\hat{\gamma}_{00} = 260.57$) für die erste Faktorstufe:



Und jetzt für die einzelnen Lastwagenfahrerinnen und Lastwagenfahrer (schwarze Linie mit Punkten). Zum Vergleich repräsentiert die dünne Linie den durchschnittlichen Verlauf.



Teil II.

Exploratorische Datenreduktion

Das Ziel der PCA (Hauptkomponentenanalyse) und der EFA (Exploratorische Faktorenanalyse) ist es, Daten zu reduzieren, d.h. die Interkorrelationen von Variablen durch wenige, möglichst voneinander unabhängige "Faktoren" bzw. "Komponenten" zu repräsentieren. Das ist besonders nützlich, wenn wir Daten mit vielen Variablen haben. Beide Verfahren können aufzeigen, welcher Anteil der Gesamtvarianz der Daten erklärt werden kann, wenn die Daten auf eine gewisse Anzahl Komponenten/Faktoren reduziert werden. Typischerweise können so Gruppen von Variablen, die stark miteinander interkorrelieren, zu zugrundeliegenden Dimensionen zusammengefasst werden. Für die Unterschiede zwischen den beiden Verfahren siehe die Vorlesungsunterlagen.

EFA und PCA sind beides explorative Verfahren. Bei beiden wird in der Regel keine a-priori Annahme getroffen, wie viele Dimensionen den Daten zu Grunde liegen.

Die Aufgabe solcher Verfahren ist es zunächst, herauszufinden, wieviele Faktoren extrahiert werden sollen. Wenn auf zu wenige Faktoren reduziert wird, gehen möglicherweise wertvolle Informationen verloren. Extrahiert man zu viele Faktoren, läuft man Gefahr, die Daten zu overfitten und zufällig entstandene Kovarianzen auf einen nicht vorhandenen oder unwichtigen Faktor zurückzuführen.

Die State-of-the-Art Methode, um die Anzahl zu extrahierender Faktoren zu determinieren, ist die **Parallelanalyse**. Dabei wird ein Datenframe mit der selben Anzahl von Variablen und derselben Stichprobengrösse simuliert, d.h. es wird eine Stichprobe aus einer Population gezogen, in der alle Interkorrelationen der Variablen = 0 sind. Auf diese Daten wird dann eine PCA/EFA angewendet und geschaut, welche Eigenwerte die zufälligen Faktoren aufweisen. Wenn die Eigenwerte der extrahierten Faktoren in unserem Datensatz grösser sind als die der entsprechenden zufälligen Faktoren, sollten sie extrahiert werden, da sie dann Varianz repräsentieren, die über das Zufallsniveau hinaus geht. Das macht man nicht nur einmal, sondern normalerweise mindestens 1000 Mal und man betrachtet dann den Mittelwert oder den Median der Verteilung der entsprechenden Eigenwerte. Manchmal - wie per default in der Funktion `psych::fa.parallel()`, die wir im Folgenden verwenden werden - wird auch das 95 %-Quantil benutzt, dann im Sinne eines Signifikanztests für die "Überzufälligkeit" eines beobachteten Eigenwerts.

Neben der Paralellanalyse, die wichtige empirische Hinweise zur Anzahl der zu extrahierenden Faktoren gibt, gibt es weitere Kriterien wie eine weitergehende Interpretation des empirischen Verlaufs der Eigenwerte ("Scree-Plot") und die inhaltliche Interpretierbarkeit der Lösung (nach einer Rotation).

Sobald die Anzahl der zu extrahierenden Faktoren festgelegt ist, können die Faktoren so rotiert werden, dass möglichst eine **Einfachstruktur** erzielt wird. Dies ist dann erreicht, wenn es keine substantiellen Querladungen mehr gibt, jedes Item soll also nur auf einen Faktor laden und die Ladungen der Items auf einem Faktor sollten möglichst hoch und homogen sein.

Daten

Packages laden, Daten einlesen und aufbereiten:

```
pacman::p_load(tidyverse, haven, EFAutilities)
```

```
# Daten einlesen
data <- read_sav("https://github.com/methodenlehre/data/blob/master/beispieldaten.sav?raw=

# Datenframe mit nur den Items zur Lebenszufriedenheit erstellen
ls <- data %>%
  select(num_range("leben", 1:10)) %>%
  drop_na()
```

Die Daten der Lebenszufriedenheit bestehen aus 10 Items, welche sich auf verschiedene Aspekte/Bereiche der Lebenszufriedenheit beziehen. Die Jugendlichen wurden gefragt:

"Wie zufrieden bist Du...

Tabelle 3.1.: Variablennamen und zugehörige Fragen

	Fragen
leben1	mit deinen Schulnoten?
leben2	mit deinem Aussehen und deiner Erscheinung?
leben3	mit der Beziehung zu deinen Lehrern?
leben4	mit allem, was mit der Schule zu tun hat?
leben5	mit allem, was mit deiner Beziehung zu anderen Menschen zu tun hat?
leben6	mit deiner Person?
leben7	mit der Beziehung zu deinen Freunden?
leben8	mit der Beziehung zu deinen Eltern?
leben9	mit dem Zusammenleben mit den anderen Familienmitgliedern?
leben10	mit dem Lebensstandard und dem Ansehen deiner Familie?

Geantwortet wurde auf einer 7-stufigen Skala von 1 = "überhaupt nicht zufrieden" bis 7 = "sehr zufrieden".

Die Fragen zur Lebenszufriedenheit können in vier Aspekte/Bereiche gegliedert werden:

Familie Item: 8, 9, 10

Schule Item: 1, 3, 4

Selbst Item: 2, 6

Freunde Item: 7, 5

Entsprechend ändern wir jetzt die Namen der Variablen, damit die Zuordnung etwas klarer wird. Um die Reihenfolge der Variablen mit ihrer inhaltlichen Ausrichtung in Übereinstimmung zu bringen, ändern wir die Reihenfolge noch mit `select()`.

```
ls <- ls %>%
  rename(
    leben8_familie = leben8,
    leben9_familie = leben9,
    leben10_familie = leben10,
    leben1_schule = leben1,
    leben3_schule = leben3,
    leben4_schule = leben4,
    leben2_selbst = leben2,
    leben6_selbst = leben6,
    leben5_freunde = leben5,
    leben7_freunde = leben7
  ) %>%
  select(
    leben1_schule, leben3_schule, leben4_schule, leben2_selbst,
    leben6_selbst, leben5_freunde, leben7_freunde, leben8_familie,
    leben9_familie, leben10_familie
  )
```

```
summary(ls)
```

```
leben1_schule   leben3_schule   leben4_schule   leben2_selbst
Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
1st Qu.:4.000   1st Qu.:4.000   1st Qu.:4.000   1st Qu.:5.000
Median :5.000   Median :5.000   Median :5.000   Median :5.000
Mean   :4.754   Mean   :4.797   Mean   :4.435   Mean   :5.214
3rd Qu.:6.000   3rd Qu.:6.000   3rd Qu.:5.000   3rd Qu.:6.000
Max.   :7.000   Max.   :7.000   Max.   :7.000   Max.   :7.000
leben6_selbst   leben5_freunde  leben7_freunde  leben8_familie
Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
1st Qu.:5.000   1st Qu.:5.000   1st Qu.:6.000   1st Qu.:5.000
Median :6.000   Median :6.000   Median :6.000   Median :6.000
Mean   :5.678   Mean   :5.594   Mean   :6.217   Mean   :5.833
3rd Qu.:6.000   3rd Qu.:6.000   3rd Qu.:7.000   3rd Qu.:7.000
Max.   :7.000   Max.   :7.000   Max.   :7.000   Max.   :7.000
leben9_familie  leben10_familie
```

```
 Min.   :1.000   Min.   :1.000
 1st Qu.:5.000   1st Qu.:5.000
 Median :6.000   Median :6.000
 Mean   :5.627   Mean   :5.808
 3rd Qu.:6.000   3rd Qu.:7.000
 Max.   :7.000   Max.   :7.000
```

Packages

Für EFA und PCA verwenden wir hier das Package `psych`.

Mit der Funktion `principal()` können wir PCAs berechnen. Wenn wir die Faktoren *nicht* rotieren, heissen die Hauptkomponenten im Output `PC#` (principal component #), zum Beispiel `PC1`. Rotieren wir die Komponenten mit einer orthogonalen Rotationsmethode, werden sie als `RC#` (rotated component #) beschrieben, zum Beispiel `RC1`. Rotieren wir sie mit einer obliquen Rotationsmethode, werden sie als `TC#` (transformed component #) beschrieben, zum Beispiel `TC1`.

Für die EFA verwenden wir die Funktion `fa()`. Für die von uns betrachtete Maximum-Likelihood-EFA benötigen wir noch das Argument `fm = "ml"`.

Im Package `psych` nehmen die Funktionen für PCA und EFA direkt das Datenframe als Input. Zusätzlich müssen die Anzahl zu extrahierender Komponenten/Faktoren definiert werden, und ob bzw. wie sie rotiert werden sollen.

Für die ML-EFA: Um Konfidenzintervalle (CI) und Standardfehler (SE) zu schätzen, benötigen wir zusätzlich das Package `EFAutilities`.

Die Funktion `fa.parallel()` führt eine Parallelanalyse sowohl für die PCA als auch für die EFA durch.

Beurteilung der Angemessenheit einer PCA/EFA für die Daten

Bevor eine PCA/EFA durchgeführt wird, muss zunächst sichergestellt sein, dass sich die Daten aufgrund ihrer Korrelationsstruktur auch für eine Dimensionsreduktion im Sinne der PCA/EFA eignen. Wir betrachten in einem ersten Schritt die Korrelationsmatrix und verschaffen uns so einen Überblick über die Zusammenhänge in den Daten. Danach führen wir den Bartlett's Test auf Sphärizität durch, gefolgt von der Berechnung des KMO-MSA (Kaiser-Meyer-Olkin Measure of Sampling Adequacy).

Korrelationen

Deskriptiver Überblick über die Interkorrelationsstruktur der Items: `pairs.panels()` aus `psych` gibt einen Plot aus, der Informationen zur Korrelationsmatrix visualisiert. An diesem Plot (siehe Abbildung nächste Seite) kann man sehen, dass praktisch alle Items in einem gewissen Masse miteinander korrellieren, manche (insbesondere solche aus dem selben Inhaltsbereich) aber stärker als andere. Insgesamt zeigen sich aber keine allzu starken korrelativen Zusammenhänge zwischen den Lebenszufriedenheitsvariablen.

Ausserdem können wir anhand der abgebildeten Streudiagramme eine grundlegende Korrelations- bzw. Regressionsdiagnostik vornehmen: Voraussetzung für die PCA/EFA sind lineare Zusammenhänge (da lineare Korrelationen als Input verwendet werden) und dass keine Outlier vorliegen. Für die Maximum-Likelihood EFA ist ausserdem multivariate Normalverteilung Voraussetzung. Da einige Outlier zu erkennen sind, die auch die Linearität der Zusammenhänge massgeblich verzerren (siehe rote Linien z.B. bei`leben5_freunde` ↔ `leben9_familie`) ist es als Vorsichtsmassnahme ratsam, univariate Outlier, die mehr als 3 Standardabweichungen vom Mittelwert der jeweiligen Variablen entfernt sind, aus den Daten zu entfernen.

Outlier-Entfernung:

```r
# Definition einer Funktion `keep`, die nur Datenpunkte auswählt, die zwischen
# +/- 3 Standardabweichungen einer Variablen liegen
keep <- function(x) {
  mx <- mean(x) # Wir speichern den Mittelwert der Variable
  sd3 <- 3 * sd(x) # Hier speichern wir die Standardabweichung * 3
  between(x, left = mx - sd3, right = mx + sd3)
}

# Wir nutzen die Funktion filter(). Wir behalten alle Rows (Personen), die keinen Wert hab
# der stärker vom Variablen-Mittelwert abweicht als 3 sd's.
ls_clean <- ls %>%
  filter(rowMeans(sapply(ls, keep)) == 1)
```

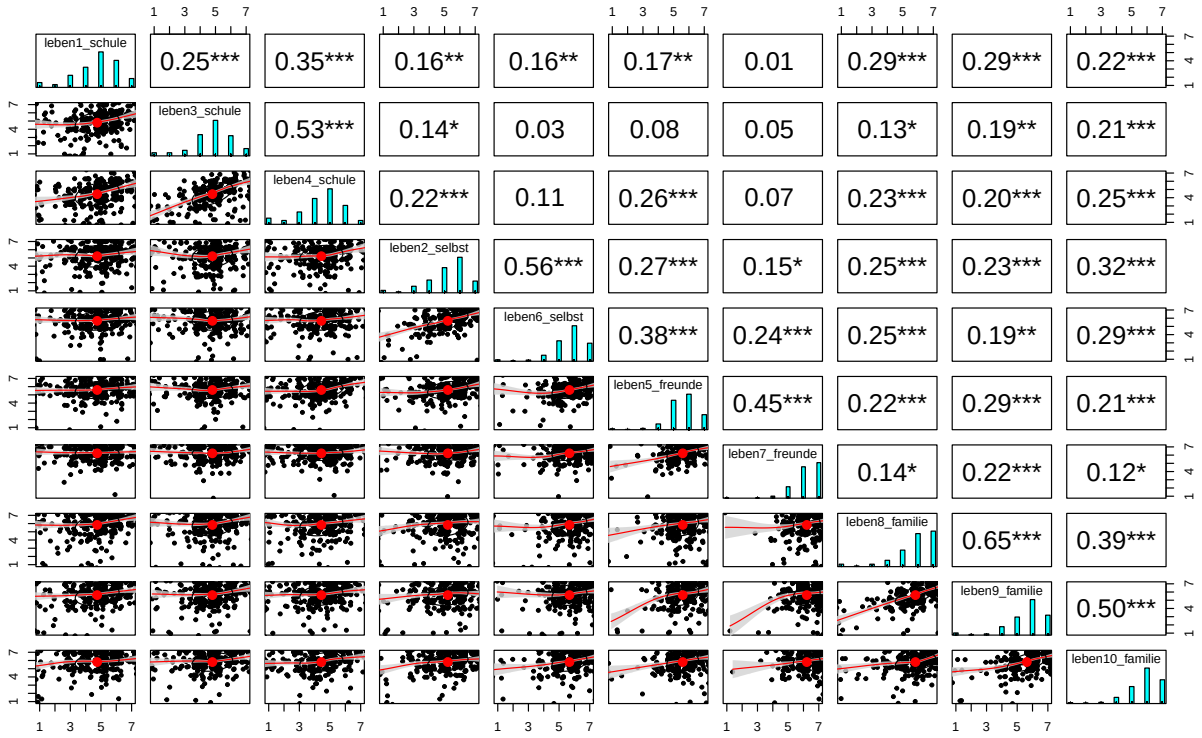
Durch die Entfernung der Outlier hat sich der Datensatz von $n = 276$ auf $n = 255$ reduziert, insgesamt wurden also 21 Personen oder 7,6 % der Daten entfernt. Ohne Outlier zeigen sich insgesamt etwas schwächere korrelative Zusammenhänge und auch kaum mehr offensichtliche Verzerrungen der linearen Zusammenhänge durch Outlier bzw. einflussreiche Datenpunkte (siehe Plots erstellt mit `pairs.panels()`)

Jetzt können wir das bereinigte Datenfile abspeichern, damit wir in den nächsten Kapiteln darauf zurückgreifen können:

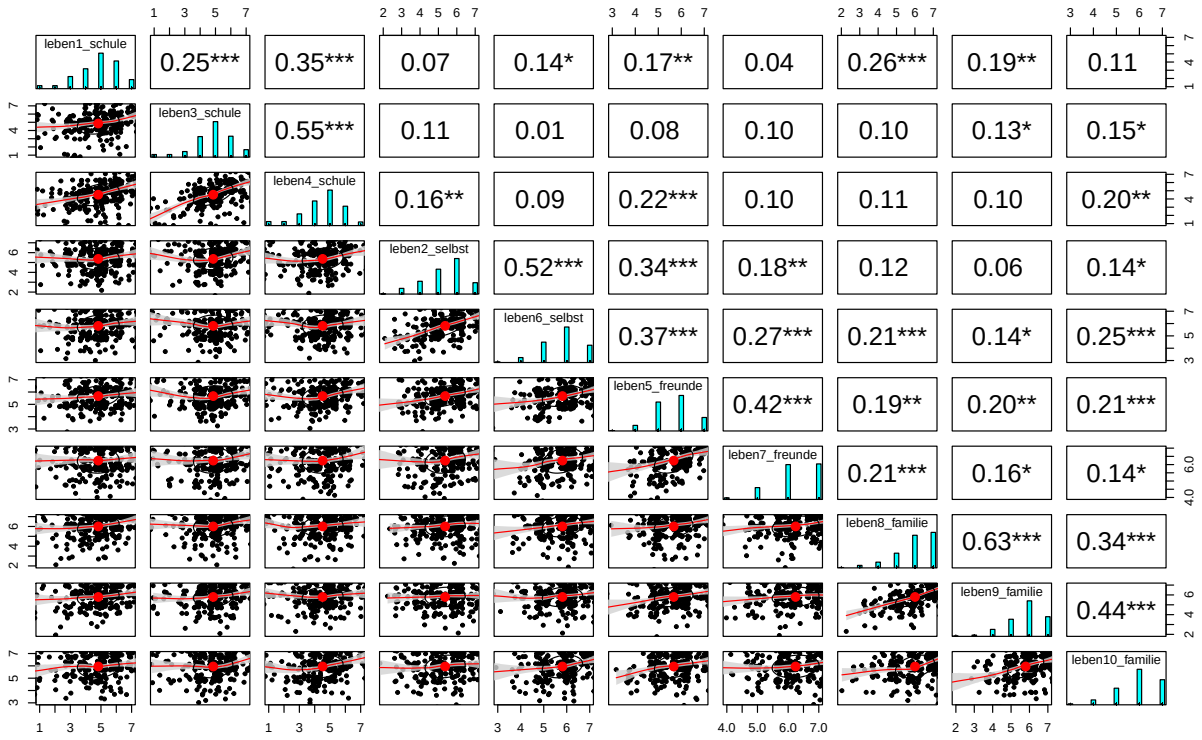
```r
library(readr)
write_csv(
  x = ls_clean,
  file = "data/ls_clean.csv"
)
```

Wir haben die Daten ausserdem auf GitHub gespeichert, dadurch können Sie auch die bereinigten Daten für die kommenden Kapitel direkt herunterladen.

```
pairs.panels(ls, density = FALSE, jiggle = TRUE, ci = TRUE, stars = TRUE)
```




```
pairs.panels(ls_clean, density = FALSE, jiggle = TRUE, ci = TRUE, stars = TRUE)
```



■ Mit dieser Methode haben wir nur univariate Ausreisser detektiert. Es ist jedoch möglich, dass multivariate Ausreisser weiterhin nicht entdeckt werden. Das könnte zum Beispiel ein Datenpunkt sein, der jeweils innerhalb zwei oder drei Standardabweichungen vom Mittelwert liegt, dann aber bivariat angeschaut weit weg liegt von anderen Datenpunkten. Eine Methode, solche Datenpunkte zu detektieren, ist der Mahalanobis-Abstand. In R können wir dafür die Funktion `mahalanobis()` aus dem standardmässig schon installierten Package `stats` verwenden. ■

Bartlett's Test auf Sphärizität

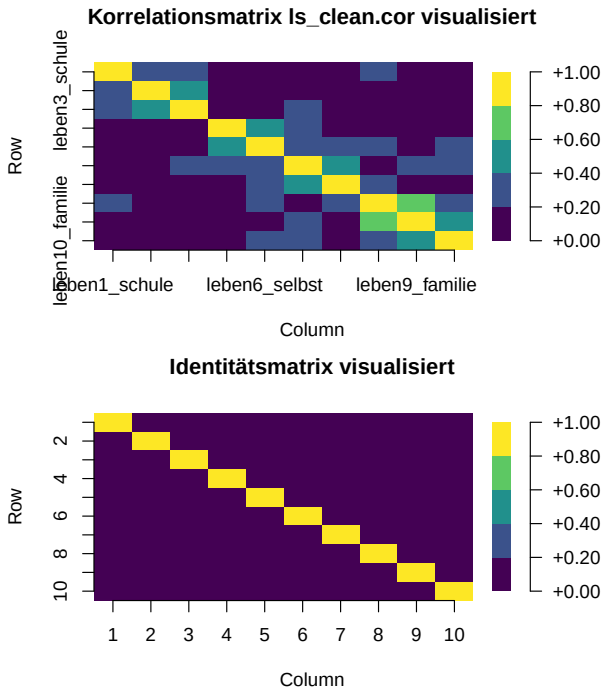
Der Bartlett's Test auf Sphärizität bewertet, ob die Variablen überhaupt miteinander korrelieren oder nicht, indem er die beobachtete Korrelationsmatrix gegen eine "Identitätsmatrix" (eine Matrix mit Einsen entlang der Hauptdiagonalen und Nullen überall sonst) testet. Wenn dieser Test statistisch nicht signifikant ist, sollten faktoranalytische Verfahren grundsätzlich nicht verwendet werden.

```
# Zuerst brauchen wir eine Korrelationsmatrix
ls_clean.cor <- cor(ls_clean)
# Bartlett's Test
cortest.bartlett(ls_clean.cor, n = length(ls$leben1_schule))
```

```
$chisq
[1] 619.2413

$p.value
[1] 5.499974e-102

$df
[1] 45
```

Der Bartlett Test ist signifikant. Es gibt also insgesamt signifikante Interkorrelationen der Items, eine Grundvoraussetzung für die Faktorenanalyse.

Visualisiert bedeutet das, dass der Bartlett Test folgende Matrizen verglichen und Unterschiede festgestellt hat.



KMO-MSA (Kaiser-Meyer-Olkin Measure of Sampling Adequacy)

Das KMO-MSA stellt eine Masszahl für die Bewertung der Eignung der beobachteten Variablen für eine Faktorenanalyse dar. Es prüft unter anderem, ob die partiellen Korrelationen der Variablen (bivariate Korrelationen partialisiert für jeweils alle anderen Variablen) nahe genug an Null liegen, um darauf hinzudeuten, dass den Variablen mindestens ein latenter Faktor zugrunde liegt. Ein höheres KMO-MSA weist auf eine bessere Eignung hin. Das KMO-MSA ist ganz einfach mit der `KMO`-Funktion des `psych`-Packages zu finden.

```r
# Die KMO-Funktion nimmt als Input entweder das Datenframe oder
# eine Korrelationsmatrix. Einfachheitshalber verwenden wir hier
# das Datenframe als input.
KMO(ls_clean)
```

```
Kaiser-Meyer-Olkin factor adequacy
Call: KMO(r = ls_clean)
Overall MSA =  0.68
MSA for each item =
  leben1_schule   leben3_schule   leben4_schule   leben2_selbst   leben6_selbst
           0.73            0.60            0.62            0.66            0.68
 leben5_freunde  leben7_freunde  leben8_familie  leben9_familie leben10_familie
           0.74            0.70            0.67            0.64            0.79
```

Im Output findet man den `Overall MSA`: Einen Kennwert der anzeigt, wie gut sich die Daten (d.h. alle Variablen zusammen) insgesamt für eine Faktorenanalyse eignen. Zusätzlich wird für jedes Item ein MSA ausgegeben.

Nach Kaiser (1975) sind Kennwerte unter 0,5 unakzeptabel (unacceptable), ab 0,5 miserabel (miserable), ab 0,6 mittelmässig (mediocre), ab 0,7 mässig (middling), ab 0,8 gut (meritorious) und ab 0,9 fantastisch (marvelous).

Unsere Daten eignen sich also mit einem Gesamt-KMO-MSA von 0,68 *mittelmässig* bis *mässig* für eine Hauptkomponentenanalyse/Faktorenanalyse. Dies trifft auch für alle einzelnen Variablen zu (MSAs zwischen 0,60 und 0,79).

4. Hauptkomponentenanalyse (PCA)

4.1. Setup

```
pacman::p_load(tidyverse, ggplot2, ggthemes, psych, haven, EFAutilities, knitr)
```

Wir können die im einführenden Kapitel zur Exploratorischen Datenreduktion bereits bereinigten Daten zur Lebenszufriedenheit einlesen, entweder aus dem `data` Ordner oder auch direkt von GitHub:

```
# Aus dem lokalen datenordner
# ls_clean <- read_csv("data/ls_clean.csv")

# Aus GitHub
ls_clean <- read_csv("https://raw.githubusercontent.com/methodenlehre/data/master/statisti
```

4.2. Parallelanalyse

Zuerst führen wir eine Parallelanalyse durch, um die Anzahl der zu extrahierenden Komponenten zu bestimmen. Weil wir die Parallelanalyse zunächst nur für die PCA berechnen, spezifizieren wir das Argument `fa` jetzt als `"pc"` (für principal component). Wenn wir das nicht spezifizieren, wird die Parallelanalyse sowohl für die PCA als auch für die Exploratorische Faktorenanalyse (EFA) zusammen dargestellt werden, was wir hier nicht wollen.

Die Anzahl Iterationen (Zufalls-Samples), die per Default 20 ist, heben wir auf 1000 an, um eine möglichst stabile Lösung zu erhalten.

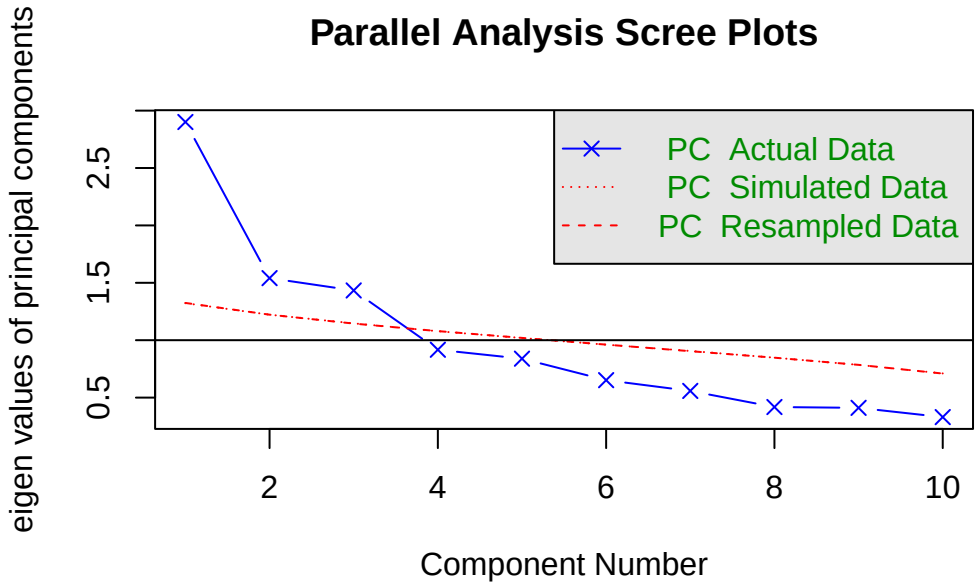
Als Kriterium für die Anzahl Komponenten wählen wir das Quantil `quant = 0.5`, also den Median. Das bedeutet, dass die Eigenwerte der tatsächlichen Komponenten, um als substantielle (zu extrahierende) Hauptkomponenten zu gelten, grösser sein müssen als der Eigenwerts-Median der jeweiligen Komponente aus den simulierten Daten (Median von jeweils 1000 simulierten Eigenwerten).

Das unter `quant` eingestellte Quantil der simulierten Eingenwerteverteilungen bezieht sich allerdings nur auf die im Text ausgegebene Schlussfolgerung, wie viele Komponenten extrahiert

werden sollten (im Sinne eines Vergleichs des jeweiligen empirischen Eigenwerts mit diesem Quantil). Die Voreinstellung (default) für diesen Test ist `quant = 0.95`, ein empirischer Eigenwert wird also nur dann als relevant betrachtet, wenn er den grössten 5 % der simulierten Eigenwerte entspricht.

Die im ausgegeben Plot der Parallelanalyse dargestellte Linie bezieht sich dagegen unabhängig von der `quant`-Einstellung immer auf den Mittelwert der simulierten Eigenwerteverteilungen. Daher stimmt die ausgegebene Schlussfolgerung nicht in jedem Fall mit der nach dem Plot zu ziehenden Schlussfolgerung überein. Da wir `quant = 0.5` verwenden, sollte das für unsere Beispiele aber kaum eine Rolle spielen (da der Median normalerweise sehr nahe am Mittelwert liegt).

```
parallelAnalyse <- fa.parallel(ls_clean, n.iter = 1000, fa = "pc", quant = 0.5)
```




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

Die Parallelanalyse der PCA zeigt deutlich **drei** zu extrahierende Komponenten an, da die vierte Komponente aus random Data einen höheren Eigenwert als die im Datenset gefundene vierte Komponente aufweist. Drei Komponenten würden wir wohl auch nach dem Scree-Kriterium extrahieren, da sich nach der dritten Komponente ein deutlicher "Knick" im Verlauf der Eigenwerte zeigt. Nicht zuletzt sollten auch nach dem Kaiser-Kriterium **drei** Komponenten extrahiert werden (da der Eigenwert der vierten Komponente bereits < 1 ist).

4.3. Unrotierte Lösung mit drei Hauptkomponenten

Zuerst berechnen wir eine unrotierte PCA:

```r
# Unrotierte PCA
principal(ls_clean, nfactors = 3, rotate = "none")
```

```
Principal Components Analysis
Call: principal(r = ls_clean, nfactors = 3, rotate = "none")
Standardized loadings (pattern matrix) based upon correlation matrix
                 PC1   PC2   PC3   h2   u2 com
leben1_schule   0.45  0.42  0.12 0.40 0.60 2.1
leben3_schule   0.41  0.60  0.38 0.68 0.32 2.5
leben4_schule   0.50  0.54  0.45 0.75 0.25 2.9
leben2_selbst   0.50 -0.43  0.39 0.58 0.42 2.9
leben6_selbst   0.57 -0.51  0.20 0.63 0.37 2.2
leben5_freunde  0.62 -0.33  0.22 0.54 0.46 1.8
leben7_freunde  0.48 -0.34  0.09 0.36 0.64 1.9
leben8_familie  0.62  0.08 -0.56 0.71 0.29 2.0
leben9_familie  0.60  0.14 -0.63 0.78 0.22 2.1
leben10_familie 0.58  0.05 -0.33 0.44 0.56 1.6

                       PC1  PC2  PC3
SS loadings           2.90 1.54 1.43
Proportion Var        0.29 0.15 0.14
Cumulative Var        0.29 0.44 0.59
Proportion Explained  0.49 0.26 0.24
Cumulative Proportion 0.49 0.76 1.00

Mean item complexity =  2.2
Test of the hypothesis that 3 components are sufficient.

The root mean square of the residuals (RMSR) is  0.1
 with the empirical chi square  211.22  with prob <  5.6e-35

Fit based upon off diagonal values = 0.85
```

Im Output bekommen wir unter `SS loadings` (Sum of Squared Loadings) zum einen die Eigenwerte der Komponenten (vgl. Scree-Plot Parallelanalyse). Zum anderen die Anteile bzw. die kumulativen Anteile der durch die Komponenten erklärten Varianz, also den Eigenwert dividiert durch die Anzahl Variablen.

In den letzten beiden Zeilen erhält man noch die Anteile bzw. kumulativen Anteile der erklärten Varianz der jeweiligen Komponenten an der insgesamt durch alle Komponenten erklärten Varianz. Z.B. werden durch die drei Komponenten insgesamt 59 % der Gesamtvarianz erklärt (Cumulative Var $PC3 = 0.59$). Also ist die Proportion explained für $PC1 = 0.29/0.59 = 0.49$ und die für $PC2 = 0.15/0.59 = 0.25$.

Des weiteren wird noch ein Chi-Quadrat-Test zum Test dafür, dass drei Komponenten ausreichen, ausgegeben. Wir betrachten diesen hier nicht, da dieser Test bei der PCA umstritten ist, weil er trotz der rein datenreduzierenden und damit deskriptiven Funktion der PCA einen Test der verbleibenden Residual(ko-)varianz in der Population durchführt. Mit dieser Art von Test beschäftigen wir uns im folgenden Kapitel zur ML-EFA.

Bei der unrotierten Lösung sieht man relativ grosse Querladungen der einzelnen Items. Das ist normal: Die Komponenten wurden so extrahiert, dass die erste Komponente möglichst viel erklärt, d.h. die Ladungen aller Variablen auf der ersten Komponente wurden maximiert (hier z.B. keine Ladung kleiner als 0.41).

Die variable `com` im Output steht hier für Komplexität der Items. Je höher die Komplexität, desto mehr laden sie auf verschiedenen Komponenten. Die Berechnung dieser statistischen Grösse behandeln wir hier nicht.

`h2` steht für Kommunalität, also den Anteil der Varianz einer beobachteten Variablen, der durch die extrahierten Komponenten erklärt wird. `u2` steht für Einzigartigkeit (uniqueness), also für den durch die Komponenten unerklärten Varianzanteil der Variablen `(u2 = 1 - h2)`.

Von Hand berechnet:

Um eine etwas grössere Genauigkeit zu erzielen, sind in folgender Tabelle die Ladungen nochmals auf drei Dezimalstellen gerundet.

Tabelle 4.1.: Ladungen auf drei Dezimalstellen gerundet

	PC1	PC2	PC3
leben1_schule	0.453	0.422	0.115
leben3_schule	0.408	0.605	0.383
leben4_schule	0.500	0.545	0.451
leben2_selbst	0.501	-0.429	0.387
leben6_selbst	0.574	-0.510	0.202
leben5_freunde	0.616	-0.334	0.218
leben7_freunde	0.483	-0.338	0.093
leben8_familie	0.625	0.084	-0.563
leben9_familie	0.604	0.140	-0.630
leben10_familie	0.576	0.050	-0.330

Berechnung des Eigenwerts `SS loadings` der *zweiten* Komponente:

$$\begin{aligned} Var(H_2) &= \lambda_{12}^2 + \lambda_{22}^2 + \lambda_{32}^2 + \lambda_{42}^2 + \lambda_{52}^2 + \lambda_{62}^2 + \lambda_{72}^2 + \lambda_{82}^2 + \lambda_{92}^2 + \lambda_{102}^2 \\ &= 0.422^2 + 0.605^2 + 0.545^2 + (-0.429)^2 + (-0.51)^2 + (-0.334)^2 + (-0.338)^2 \\ &\quad + 0.084^2 + 0.14^2 + 0.05^2 \\ &= 1.54 \end{aligned}$$

Berechnung der Kommunalität `h2` des Items `leben2_selbst`:

$$\begin{aligned} \hat{H}(Y_4) &= \lambda_{41}^2 + \lambda_{42}^2 + \lambda_{43}^2 \\ &= 0.501^2 + (-0.429)^2 + 0.387^2 \\ &= 0.58 \end{aligned}$$

Berechnung der Uniqueness `u2` des Items `leben8_familie`:

$$\begin{aligned} \hat{U}(Y_8) &= 1 - (\lambda_{81}^2 + \lambda_{82}^2 + \lambda_{83}^2) \\ &= 1 - [0.625^2 + 0.084^2 + (-0.563)^2] \\ &= 0.29 \end{aligned}$$

4.4. Orthogonale Rotation mit drei Hauptkomponenten

Jetzt rotieren wir die Komponenten mit der orthogonalen Varimax-Rotation. Dabei werden die Komponenten so rotiert, dass die Items möglichst wenige Querladungen aufweisen, also bestmöglich eine Einfachstruktur darstellen und gleichzeitig die Orthogonalität der Lösung erhalten bleibt (keine Interkorrelationen der Komponenten).

Im Gegensatz zu obliquen Rotationen bleiben hier die Summen der Eigenwerte identisch. Auch wenn in der aktuellen Literatur meist oblique Rotationsmethoden empfohlen werden, sind orthogonale Rotationen wegen ihrer Eigenschaft, dass die Komponenten weiterhin als unabhängige Dimensionen interpretiert werden können, attraktiv (insbesondere wenn sich damit eine Einfachstruktur erreichen lässt).

```r
# Rotierte PCA
pca.3.rotiert <- principal(ls_clean, nfactors = 3, rotate = "varimax")

# Per Default ist das principal-Objekt nicht sortiert. Damit wir die
# Items in der Reihenfolge der Faktorenladungen bekommen, müssen wir
# die Funktion `print.psych()` auf das Objekt anwenden und den
# Parameter `sort = TRUE` setzen.
print.psych(pca.3.rotiert, sort = TRUE)
```

```
Principal Components Analysis
Call: principal(r = ls_clean, nfactors = 3, rotate = "varimax")
Standardized loadings (pattern matrix) based upon correlation matrix
                item  RC1   RC3   RC2   h2   u2 com
leben6_selbst      5 0.78  0.13 -0.01 0.63 0.37 1.1
leben2_selbst      4 0.76 -0.05  0.10 0.58 0.42 1.0
leben5_freunde     6 0.70  0.16  0.14 0.54 0.46 1.2
leben7_freunde     7 0.57  0.18  0.02 0.36 0.64 1.2
leben9_familie     9 0.05  0.88  0.07 0.78 0.22 1.0
leben8_familie     8 0.12  0.83  0.08 0.71 0.29 1.1
leben10_familie   10 0.21  0.62  0.14 0.44 0.56 1.3
leben4_schule      3 0.14  0.03  0.85 0.75 0.25 1.1
leben3_schule      2 0.01  0.03  0.82 0.68 0.32 1.0
leben1_schule      1 0.06  0.24  0.58 0.40 0.60 1.4

                       RC1  RC3  RC2
SS loadings           2.08 1.99 1.80
Proportion Var        0.21 0.20 0.18
Cumulative Var        0.21 0.41 0.59
Proportion Explained  0.35 0.34 0.31
Cumulative Proportion 0.35 0.69 1.00

Mean item complexity =  1.1
Test of the hypothesis that 3 components are sufficient.

The root mean square of the residuals (RMSR) is  0.1
 with the empirical chi square  211.22  with prob <  5.6e-35

Fit based upon off diagonal values = 0.85
```

Die Reihenfolge der Darstellung der rotierten Komponenten (RC) ergibt sich aus der Grösse der Eigenwerte/des Varianzanteils nach der Rotation. Jetzt ist der Anteil der erklärten Varianz der drei Komponenten sehr ähnlich, da durch die Rotation das Kriterium der sukzessiv maximalen Varianzaufklärung aufgehoben wurde und die erklärte Varianz mit dem Ziel einer Einfachstruktur möglichst gleichmässig auf die Komponenten verteilt wurde.

Wir können aufgrund folgender Tabelle beispielhaft nochmal einen rotierten Eigenwert von Hand berechnen, diesmal den der ersten Komponente:

Tabelle 4.2.: Rotierte Eigenwerte auf 3 Dezimalstellen gerundet

	RC1	RC3	RC2
leben1_schule	0.057	0.243	0.578
leben3_schule	0.014	0.031	0.823
leben4_schule	0.139	0.027	0.854
leben2_selbst	0.756	-0.052	0.102
leben6_selbst	0.784	0.127	-0.015
leben5_freunde	0.701	0.163	0.143
leben7_freunde	0.569	0.178	0.018
leben8_familie	0.124	0.833	0.076
leben9_familie	0.048	0.880	0.075
leben10_familie	0.206	0.617	0.142

$$\begin{aligned} Var(H_{1rotiert}) &= \lambda_{11}^2 + \lambda_{21}^2 + \lambda_{31}^2 + \lambda_{41}^2 + \lambda_{51}^2 + \lambda_{61}^2 + \lambda_{71}^2 + \lambda_{81}^2 + \lambda_{91}^2 + \lambda_{101}^2 \\ &= 0.057^2 + 0.014^2 + 0.139^2 + 0.756^2 + 0.784^2 + 0.701^2 + 0.569^2 + 0.124^2 \\ &\quad + 0.048^2 + 0.206^2 \\ &= 2.084 \end{aligned}$$

Daraus folgender Anteil der erklärten Varianz der ersten rotierten Komponente:

Proportion Var $= \frac{2.084}{10} = 0.2084$

Die Kommunalität der Items bleibt nach orthogonaler Rotation dagegen unverändert, z.B. für `leben2_selbst`:

$$\begin{aligned} \hat{H}(Y_4) &= \lambda_{41}^2 + \lambda_{42}^2 + \lambda_{43}^2 \\ &= 0.756^2 + (-0.052)^2 + 0.102^2 \\ &= 0.58 \end{aligned}$$

Tatsächlich gibt es jetzt weniger und deutlich schwächere Querladungen. Entsprechend ist auch die `com` (Komplexität) der Items gesunken. Jetzt erst zeigt sich eine Einfachstruktur und damit auch, wie die Komponenten inhaltlich zu interpretieren sind: auf der ersten rotierten Komponente (RC1) laden die Items der Inhaltsbereiche **Selbst** und **Freunde**, die zweite rotierte Komponente (RC2) ist als **Schul-**Komponente zu interpretieren und die dritte (RC3) als **Familien-**Komponente.

Es gibt nur wenige Querladungen. `leben10_familie` hat eine Ladung von 0.21 auf der Selbst/Freunde-Komponente und `leben1_schule` hat eine Ladung von 0.24 auf der Familien-Komponente. Alle anderen Ladungen auf nicht zugehörigen Komponenten sind < 0.20.

4.5. Oblique Rotation mit drei Hauptkomponenten

Jetzt benutzen wir als oblique Rotation eine Oblimin-Rotation. Je stärker die Komponenten (tatsächlich) miteinander korrellieren, desto grösser ist der Vorteil von obliquen Rotationen. Diese können anders als orthogonale Rotationen eine Korrelation der latenten Variablen (Komponenten) selbst abbilden und damit bei den Ladungen (die jetzt nicht mehr als Korrelationen, sondern als Partial-Regressionskoeffizienten zu interpretieren sind) eine noch bessere Einfachstruktur erreichen. Falls die Dimensionen tatsächlich orthogonal sein sollten, so wird sich das in Komponentenkorrelationen nahe Null widerspiegeln. Das ist auch der Grund, warum man orthogonale Rotationen nur in seltenen Fällen wirklich benötigt.

```r
# Oblique Rotation
pca.3.rotiert.obli <- principal(ls_clean, nfactors = 3, rotate = "oblimin")


# Sortiert ausgeben lassen:
print.psych(pca.3.rotiert.obli, sort = TRUE)
```

```
Principal Components Analysis
Call: principal(r = ls_clean, nfactors = 3, rotate = "oblimin")
Standardized loadings (pattern matrix) based upon correlation matrix
                item   TC1   TC3   TC2   h2   u2 com
leben6_selbst      5  0.79  0.05 -0.08 0.63 0.37 1.0
leben2_selbst      4  0.77 -0.14  0.05 0.58 0.42 1.1
leben5_freunde     6  0.69  0.09  0.08 0.54 0.46 1.1
leben7_freunde     7  0.56  0.12 -0.04 0.36 0.64 1.1
leben9_familie     9 -0.05  0.90 -0.01 0.78 0.22 1.0
leben8_familie     8  0.03  0.84 -0.01 0.71 0.29 1.0
leben10_familie   10  0.13  0.60  0.07 0.44 0.56 1.1
leben4_schule      3  0.06 -0.03  0.86 0.75 0.25 1.0
leben3_schule      2 -0.06 -0.01  0.83 0.68 0.32 1.0
leben1_schule      1 -0.02  0.21  0.56 0.40 0.60 1.3

                       TC1  TC3  TC2
SS loadings           2.08 2.00 1.79
Proportion Var        0.21 0.20 0.18
Cumulative Var        0.21 0.41 0.59
Proportion Explained  0.35 0.34 0.30
Cumulative Proportion 0.35 0.70 1.00

 With component correlations of
     TC1  TC3  TC2
```

```
TC1 1.00 0.22 0.18
TC3 0.22 1.00 0.17
TC2 0.18 0.17 1.00

Mean item complexity =  1.1
Test of the hypothesis that 3 components are sufficient.

The root mean square of the residuals (RMSR) is  0.1
 with the empirical chi square  211.22  with prob <  5.6e-35

Fit based upon off diagonal values = 0.85
```

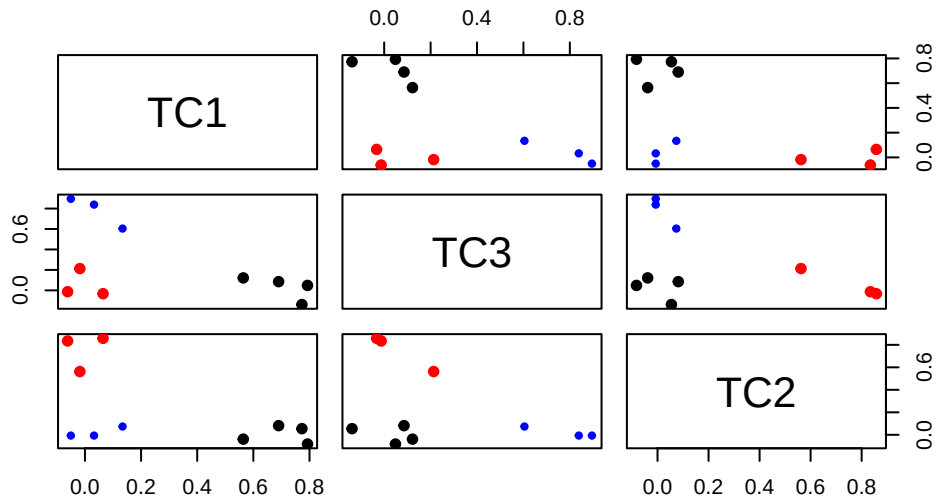
Gibt es noch substantielle Querladungen? - Das einzige Item, das noch eine gewisse Querladung aufweist, ist `leben1_schule`. Dieses bezieht sich direkt auf die *Schulnoten* und zeigt eine Querladung auf der **Familien**-Komponente (TC3). Auch ist die Ladung dieses Items auf der Schulkomponente (TC2) deutlich niedriger als die der anderen beiden zugehörigen Items. Die *Schulnoten* scheinen also auch für die Zufriedenheit für den Bereich Familie eine gewisse Bedeutung zu haben. Das ist inhaltlich durchaus nachvollziehbar.

In Bezug auf die Komponentenkorrelationen zeigt sich, dass alle drei Komponenten relativ gleichmässig und gleichzeitig nicht sonderlich stark miteinander korrelieren ($r_{H_1H_2} = 0.18, r_{H_1H_3} = 0.22, r_{H_2H_3} = 0.17$). Dies kann auf eine übergeordnete Dimension *Lebenszufriedenheit* hinweisen. Damit werden wir uns demnächst bei der konfirmatorischen Faktorenanalyse (CFA) noch weiter beschäftigen.

Visualisierung:

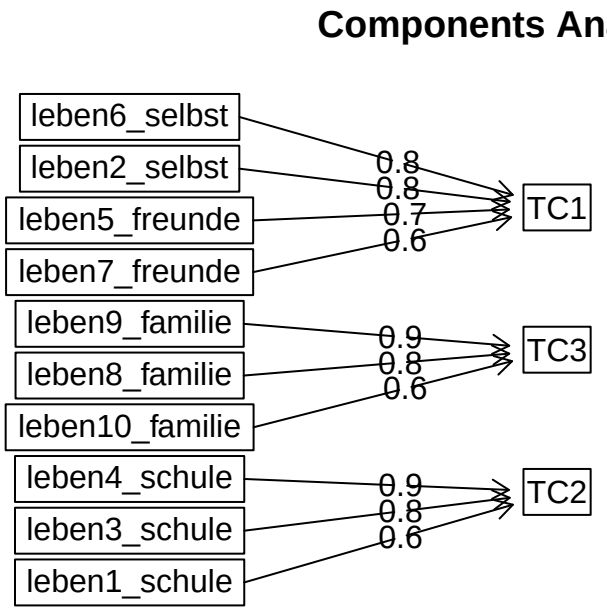
Wie stark die Items auf Komponenten laden, kann mit `factor.plot()` dargestellt werden:

```r
factor.plot(pca.3.rotiert.obli, cut = 0.5)
```



Welche Items "gehören" zu welcher Komponente? Das können wir mit `fa.diagram()` darstellen:

```
fa.diagram(pca.3.rotiert.obli)
```



4.6. Zusammenfassung PCA

Die PCA zeigt also, dass drei Hauptkomponenten insgesamt ca. 60 % der Gesamtvarianz erklären können und dass diese im Vergleich zu den übrigen sieben Komponenten über eine überdurchschnittliche Varianzaufklärung verfügen. Ob dies auch im Sinne einer EFA, die die Korrelationen zwischen den Items **vollständig** durch Faktoren erkären will, ausreicht, wird sich im nächsten Teil zeigen.

Inhaltlich interessant ist die Kombination der Lebenzufriedenheit mit dem Bereich **Selbst** und dem Bereich **Freunde**. Diese beiden Zufriedenheiten scheinen eng miteinander verknüpft zu sein.

4.7. Übung

Für die Übung bearbeiten wir ein Datenset von den 1988 Olympischen Spielen in Seoul. Im Datenset sind die Resultate des olympischen Siebenkampfs (Heptathlon) enthalten. Jede Person hat also insgesamt sieben Resultate. Es handelt sich um Daten von 25 Athletinnen, also um ein sehr kleines n für eine PCA mit 7 Variablen. Das soll uns aber fürs Üben an einem kleinen Beispiel nicht weiter stören. Die Daten sind im Package `HSAUR2` gespeichert.
Folgender Code-Chunk lädt erst das Package und dann die Daten `heptathlon`.

```
pacman::p_load(HSAUR2)
data("heptathlon", package = "HSAUR2")
```

```
heptathlon
```

	hurdles	highjump	shot	run200m	longjump	javelin	run800m
Joyner-Kersee (USA)	12.69	1.86	15.80	22.56	7.27	45.66	128.51
John (GDR)	12.85	1.80	16.23	23.65	6.71	42.56	126.12
Behmer (GDR)	13.20	1.83	14.20	23.10	6.68	44.54	124.20
Sablovskaite (URS)	13.61	1.80	15.23	23.92	6.25	42.78	132.24
Choubenkova (URS)	13.51	1.74	14.76	23.93	6.32	47.46	127.90
Schulz (GDR)	13.75	1.83	13.50	24.65	6.33	42.82	125.79
Fleming (AUS)	13.38	1.80	12.88	23.59	6.37	40.28	132.54
Greiner (USA)	13.55	1.80	14.13	24.48	6.47	38.00	133.65
Lajbnerova (CZE)	13.63	1.83	14.28	24.86	6.11	42.20	136.05
Bouraga (URS)	13.25	1.77	12.62	23.59	6.28	39.06	134.74
Wijnsma (HOL)	13.75	1.86	13.01	25.03	6.34	37.86	131.49
Dimitrova (BUL)	13.24	1.80	12.88	23.59	6.37	40.28	132.54
Scheider (SWI)	13.85	1.86	11.58	24.87	6.05	47.50	134.93

```
Braun (FRG)           13.71     1.83 13.16   24.78     6.12   44.58  142.82
Ruotsalainen (FIN)    13.79     1.80 12.32   24.61     6.08   45.44  137.06
Yuping (CHN)          13.93     1.86 14.21   25.00     6.40   38.60  146.67
Hagger (GB)           13.47     1.80 12.75   25.47     6.34   35.76  138.48
Brown (USA)           14.07     1.83 12.69   24.83     6.13   44.34  146.43
Mulliner (GB)         14.39     1.71 12.68   24.92     6.10   37.76  138.02
Hautenauve (BEL)      14.04     1.77 11.81   25.61     5.99   35.68  133.90
Kytola (FIN)          14.31     1.77 11.66   25.69     5.75   39.48  133.35
Geremias (BRA)        14.23     1.71 12.95   25.50     5.50   39.64  144.02
Hui-Ing (TAI)         14.85     1.68 10.00   25.23     5.47   39.14  137.30
Jeong-Mi (KOR)        14.53     1.71 10.83   26.61     5.50   39.26  139.17
Launa (PNG)           16.42     1.50 11.78   26.16     4.88   46.38  163.43
                     score
Joyner-Kersee (USA)   7291
John (GDR)            6897
Behmer (GDR)          6858
Sablovskaite (URS)    6540
Choubenkova (URS)     6540
Schulz (GDR)          6411
Fleming (AUS)         6351
Greiner (USA)         6297
Lajbnerova (CZE)      6252
Bouraga (URS)         6252
Wijnsma (HOL)         6205
Dimitrova (BUL)       6171
Scheider (SWI)        6137
Braun (FRG)           6109
Ruotsalainen (FIN)    6101
Yuping (CHN)          6087
Hagger (GB)           5975
Brown (USA)           5972
Mulliner (GB)         5746
Hautenauve (BEL)      5734
Kytola (FIN)          5686
Geremias (BRA)        5508
Hui-Ing (TAI)         5290
Jeong-Mi (KOR)        5289
Launa (PNG)           4566
```

```
# Im Datensatz sind die Namen der Athletinnen in den "rownames" gespeichert.
# rownames sind eine R Eigenschaft (ein Attribut von Datenframes), die wir
# bisher nicht kennengelernt haben und auf die wir auch jetzt nicht näher
```

```r
# eingehen wollen, da wir es bevorzugen, eine richtige Variable mit den
# Namen zu haben. Daher:
heptathlon <- heptathlon |>
  rownames_to_column(var = "name")

# Anschauen:
head(heptathlon)
```

```
                name hurdles highjump  shot run200m longjump javelin run800m
1 Joyner-Kersee (USA)   12.69     1.86 15.80   22.56     7.27   45.66  128.51
2          John (GDR)   12.85     1.80 16.23   23.65     6.71   42.56  126.12
3        Behmer (GDR)   13.20     1.83 14.20   23.10     6.68   44.54  124.20
4  Sablovskaite (URS)   13.61     1.80 15.23   23.92     6.25   42.78  132.24
5   Choubenkova (URS)   13.51     1.74 14.76   23.93     6.32   47.46  127.90
6        Schulz (GDR)   13.75     1.83 13.50   24.65     6.33   42.82  125.79
  score
1  7291
2  6897
3  6858
4  6540
5  6540
6  6411
```

Die Variable `score` ist die finale Wertung der Leistung einer Athletin. Diese interessiert uns momentan nicht, wir belassen sie aber für später im Datensatz.

Das Datenframe hat also 7 Variablen, die uns im Moment interessieren:

- `hurdles`: Hürdenlauf
- `highjump`: Hochsprung
- `shot`: Kugelstossen
- `run200m`: 200m Lauf
- `longjump`: Weitsprung
- `javelin`: Speerwurf
- `run800m`: 800m Lauf

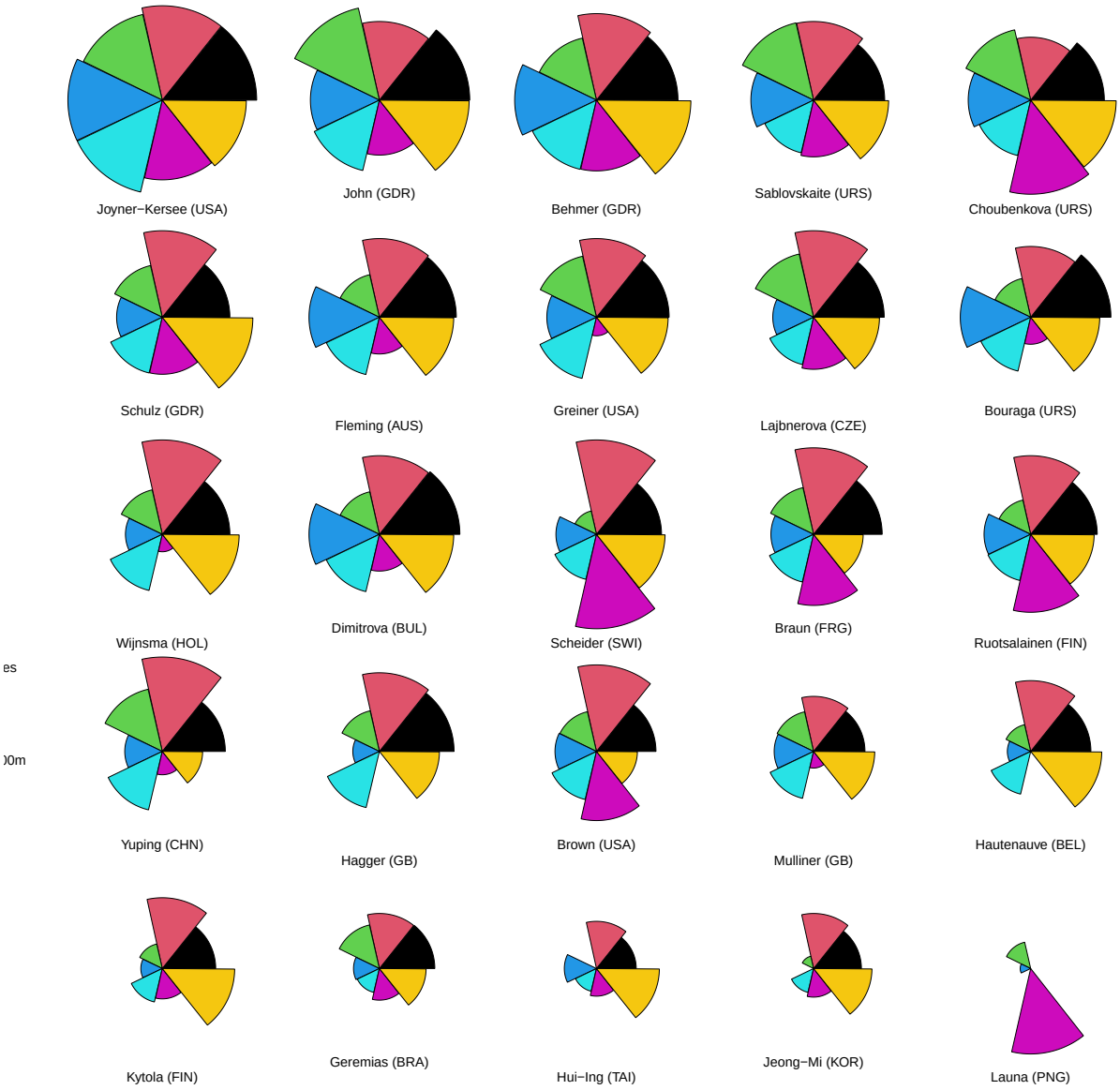
Visualisierung

Wir visualisieren die Daten, um uns einen Überblick zu verschaffen. Dies geht aber nicht so einfach, weil sich die Skalen offensichtlich stark unterscheiden. Zusätzlich sind bei manchen Sportarten kleine Werte erwünscht (`run200`) und bei anderen grosse Werte (`longjump`). Folgender Code-Chunk stellt sicher, dass für alle Sportarten grosse Werte wünschenswert sind:

```
# Wir subtrahieren den Wert von "falsch kodierten" Variablen vom Maximalwert
# dieser Variable:
heptathlon <- heptathlon |>
  mutate(
    hurdles = max(hurdles) - hurdles,
    run200m = max(run200m) - run200m,
    run800m = max(run800m) - run800m
  )
# Die Werte der veränderten Variablen lassen sich jetzt zwar nicht mehr so
# einfach interpretieren, dafür zeigen sie nun alle in dieselbe Richtung.
```

Jetzt können wir uns die Daten in einem Sternenplot anschauen:

```
par(mar = c(4, 4, 0.1, 0.1))
heptathlon |>
  select(-name, -score) |>
  stars(
    draw.segments = TRUE,
    key.loc = c(-0.7, 5),
    nrow = 5,
    scale = TRUE,
    labels = heptathlon$name
  )
```



Aufgabe 1

■

■

Im Datenframe gibts einen Outlier. Die Werte dieser Athletin unterscheiden sich stark von denen aller anderen Teilnehmerinnen. Entfernen Sie den Outlier basierend auf dem `starplot` von oben.

4.7.0.0.1. * Solution

Der Outlier ist Launa. Sie ist sehr gut im Speerwerfen, hat aber sonst sehr kleine Werte.

```
heptathlon <- heptathlon |>
  filter(name != "Launa (PNG)")
```

Aufgabe 2

■

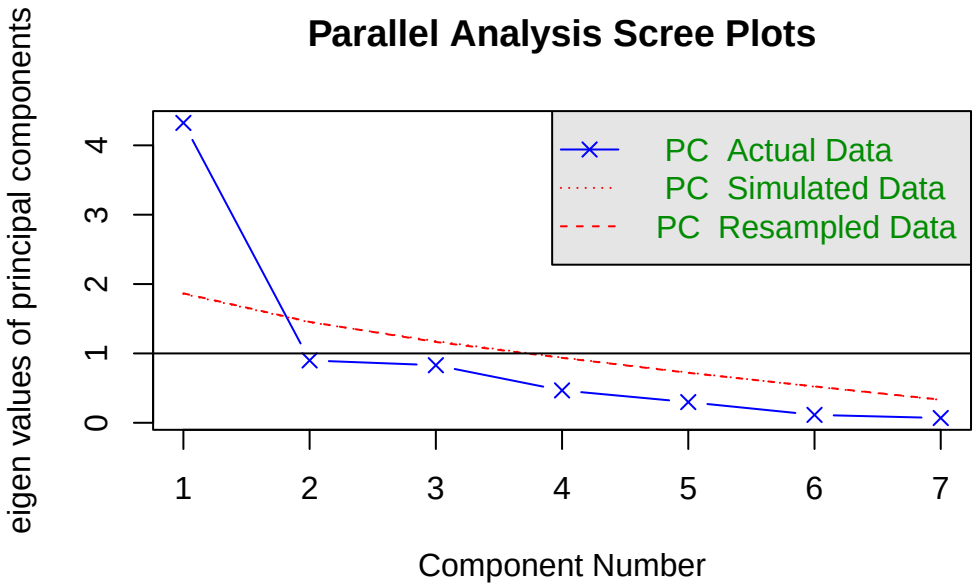
■

Das Ziel dieser Übung ist die Reduktion der 7 Scores auf möglichst *eine* Dimension, um eine informative Rangreihenfolge der Athletinnen zu erstellen, die die Leistung über alle sieben Disziplinen hinweg möglichst optimal widerspiegeln soll. In einem ersten Schritt möchten wir wissen, ob sich diese Daten überhaupt auf weniger bzw. auf eine einzige Dimension reduzieren lassen.

Führen Sie dafür eine Parallelanalyse (für PCA) durch und erklären Sie, wie viele Hauptkomponenten angebracht sind.

4.7.0.0.1. * Solution

```
library(psych) # falls nicht mehr geladen
heptathlon |>
  select(-name, -score) |>
  fa.parallel(fa = "pc", n.iter = 1000, quant = 0.5)
```




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

Die Parallelanalyse empfiehlt eine Lösung mit nur einer Hauptkomponente. Das bedeutet für uns, dass mit einer Hauptkomponente bereits genügend Varianz eingefangen wird, um einen guten Siebenkampf-Score zu berechnen, es aus empirischer Sicht also nicht nötig ist, mehrere Disziplin-Dimensionen zu betrachten (man könnte natürlich auch dann einen Gesamtscore berechnen, wenn sich herausstellen sollte, dass man eigentlich mehrere Komponenten benötigt - dann würde dieser Gesamtscore einfach keine homogene Siebenkampf-Fähigkeit widerspiegeln).

Aufgabe 3

■

■

Berechnen Sie eine PCA mit *einer* Hauptkomponente über die sieben Disziplinen.

Wieviel Prozent Varianz wird mit dieser Hauptkomponente erklärt? (Output)

Berechnen Sie ausserdem den Eigenwert dieser Komponente von Hand und daraus dann nochmals die % erklärte Varianz. (Tipp: die Kommunalitäten `h2` ensprechen jetzt den quadrierten Ladungen auf der ersten und einzigen Komponente!)

4.7.0.0.1. * Solution

```r
# Das letzte Argument könnte wir uns auch sparen, da eine einzige
# Komponente nicht rotiert werden kann
pca_heptathlon <- heptathlon |>
  select(-name, -score) |>
  principal(factors = 1, rotate = "none")

# Ergebnisobjekt drucken, um Ladungen nach Grösse sortieren zu können
print.psych(pca_heptathlon, sort = TRUE)
```

```
Principal Components Analysis
Call: principal(r = select(heptathlon, -name, -score), rotate = "none",
    factors = 1)
Standardized loadings (pattern matrix) based upon correlation matrix
         V  PC1   h2   u2 com
longjump 5 0.94 0.88 0.12   1
hurdles  1 0.94 0.88 0.12   1
run200m  4 0.89 0.79 0.21   1
shot     3 0.84 0.70 0.30   1
highjump 2 0.65 0.43 0.57   1
run800m  7 0.63 0.40 0.60   1
javelin  6 0.50 0.25 0.75   1

                PC1
SS loadings    4.32
Proportion Var 0.62

Mean item complexity =  1
Test of the hypothesis that 1 component is sufficient.

The root mean square of the residuals (RMSR) is  0.1
 with the empirical chi square  9.81  with prob <  0.78

Fit based upon off diagonal values = 0.97
```

Mit der ersten Komponente werden 62% der Varianz erklärt. Dies lässt sich direkt aus dem Output (`Proportion Var`) ablesen.

Für den Eigenwert (`SS Loadings`) von Komponente 1 quadrieren wir die Ladung jeder Variablen auf Komponente 1 und summieren diese Werte anschliessend. Oder wir nehmen die Kommunalitäten `h2` und summieren diese! Von Hand lässt sich in diesem Fall auch mit R als "Taschenrechner" interpretieren. In der Prüfung müssen Sie das allerdings mit einem normalen Taschenrechner rechnen…

```r
# Erste Möglichkeit:
# Um genauere Werte zu bekommen, extrahieren wir die Ladungen und
# runden sie auf 4 Stellen hinter dem Komma:
loadings <- round(pca_heptathlon$loadings[, 1], 4)
# Das Quadrieren können wir dann auch vektorisiert machen...
loadings_squared <- loadings^2
# Und wenn wir schon dabei sind...
eigenvalue <- sum(loadings_squared)
eigenvalue |> round(2)
```

```
[1] 4.32
```

```r
# Und daraus dann...
percent_explained <- eigenvalue / 7 * 100
round(percent_explained, 2)
```

```
[1] 61.77
```

```r
# Zweite Möglichkeit (wirklich von Hand):
loadings # Ladungen ausgeben
```

```
 hurdles highjump     shot  run200m longjump  javelin  run800m
  0.9365   0.6540   0.8369   0.8881   0.9377   0.5038   0.6298
```

```r
pctexplained_vonhand <- (0.9365^2 + 0.654^2 + 0.8369^2 +
  0.8881^2 + 0.9377^2 + 0.5038^2 + 0.6298^2) / 7 * 100
round(pctexplained_vonhand, 2)
```

```
[1] 61.77
```

```r
# Dritte Möglichkeit (über die Kommunalitäten h2):
pctexplained_ueberh2 <- sum(pca_heptathlon$communality) / 7 * 100
round(pctexplained_ueberh2, 2)
```

```
[1] 61.77
```

Der Eigenwert der ersten Komponente ist also 4.32 und sie erklärt 61.77 % der Gesamtvarianz.

Vertiefung: Aufgabe 4

■

■

a) Extrahieren Sie die `scores` der ersten Hauptkomponente.

b) Wer ist die Gewinnerin des Heptathlons nach unserer Methode?

c) Schauen Sie ausserdem, wie stark die `scores` mit der offiziellen Score aus dem Datenframe `heptathlon` zusammenhängt.

4.7.0.0.1. * Solution

4.7.0.0.1.1. * a) Extraktion der Scores

```
pc1_scores <- pca_heptathlon$scores[, 1]
pc1_scores
```

```
 [1]  2.288003481  1.513916923  1.407268194  0.619493399  0.723043464
 [6]  0.460948428  0.458533216  0.304539034  0.183506561  0.251196423
[11]  0.104697557  0.517465086 -0.001449975 -0.052508930 -0.100449355
[16] -0.111817913 -0.317178053 -0.363988434 -0.904582980 -0.879208346
[21] -1.018691636 -1.332495085 -1.876159087 -1.874081972
```

4.7.0.0.1.2. * b) Gewinnerin?

Grundsätzlich müssen wir dafür nur die Person (`name`) mit dem grössten Wert auf `pc1_scores` identifizieren. Dafür können wir die Funktion `which.max()` verwenden.

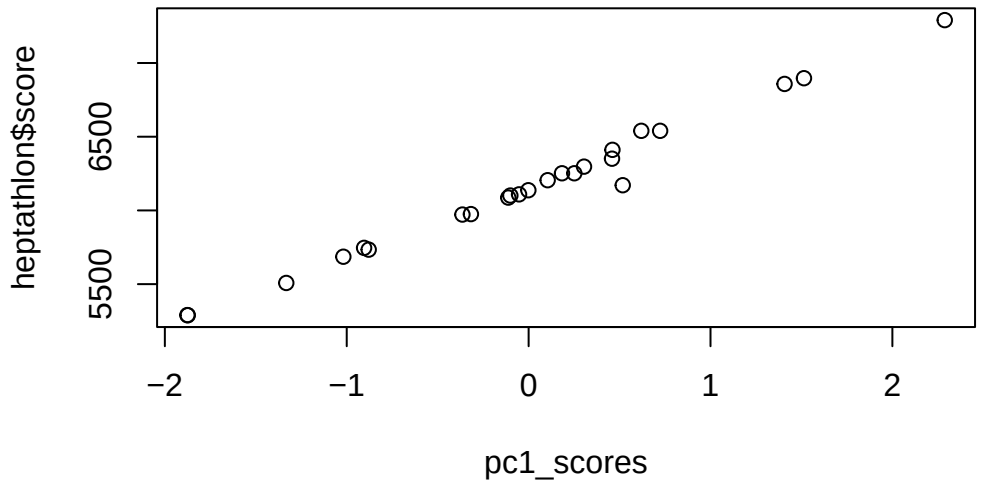
```
heptathlon$name[which.max(pc1_scores)]
```

```
[1] "Joyner-Kersee (USA)"
```

Jackie Joyner-Kersee aus den USA hat den höchsten Siebenkampf-Score nach der PCA-Methode.

4.7.0.0.1.3. * c) Streudiagramm und Korrelation zwischen den PCA-Scores und den Original-Scores

```
plot(pc1_scores, heptathlon$score)
```




```
cor(pc1_scores, heptathlon$score)
```

```
[1] 0.9931168
```

So wie es scheint haben die Verantwortlichen für die Bewertung der Teilnehmerinnen eine ähnliche Methode angewendet. Auf jeden Fall kommen wir zu sehr ähnlichen Resultaten, was durch die geringen Abweichungen von der Geraden im Streudiagramm sowie durch die Korrelation nahe 1 veranschaulicht wird.

5. Exploratorische Faktorenanalyse (EFA)

5.1. Setup

```
pacman::p_load(tidyverse, ggplot2, ggthemes, psych, haven, EFAutilities, knitr)
```

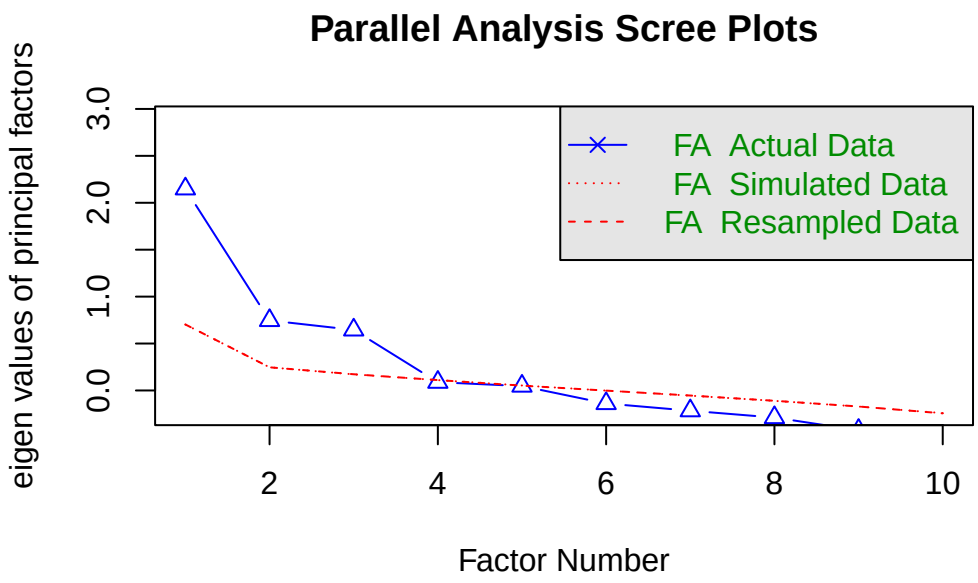
Wir können wieder die im einführenden Kapitel zur Exploratorischen Datenreduktion bereits bereinigten Daten zur Lebenszufriedenheit einlesen, entweder aus dem `data` Ordner oder auch direkt von GitHub:

```
# Aus dem lokalen Datenordner
# ls_clean <- read_csv("data/ls_clean.csv")

# Aus GitHub
ls_clean <- read_csv("https://raw.githubusercontent.com/methodenlehre/data/master/statisti
```

5.2. Parallelanalyse

Zuerst führen wir eine Parallelanalyse durch. Weil wir die Parallelanalyse jetzt für die Faktorenanalyse berechnen (und nicht wie im vorigen Kapitel für die Hauptkomponentenanalyse), spezifizieren wir jetzt `fa = "fa"`. Ausserdem benutzen wir das Argument `fm = "ml"`, damit die Parallelanalyse auf einer **Maximum-Likelihood**-Faktorenanalyse basiert.

```
parallelAnalyse <- fa.parallel(ls_clean,
  n.iter = 1000, fa = "fa",
  fm = "ml", quant = 0.5
)
```




```
Parallel analysis suggests that the number of factors =  3  and the number of components =  1
```

Die Parallelanalyse zur ML-EFA zeigt, dass auch hier drei Faktoren extrahiert werden sollten. Der Eigenwert des vierten Faktors liegt knapp unter dem Median der 1000 simulierten vierten Eigenwerte einer Zufallsmatrix.

5.3. ML-EFA mit drei unrotierten Faktoren

```r
# Unrotierte EFA mit 3 Faktoren
efa.3.unrotiert <- fa(ls_clean, fm = "ml", nfactors = 3, rotate = "none")
print.psych(efa.3.unrotiert, sort = TRUE)
```

```
Factor Analysis using method =  ml
Call: fa(r = ls_clean, nfactors = 3, rotate = "none", fm = "ml")
Standardized loadings (pattern matrix) based upon correlation matrix
                item  ML1   ML2   ML3   h2   u2 com
leben4_schule      3 0.83 -0.42 -0.07 0.87 0.13 1.5
leben3_schule      2 0.55 -0.22 -0.11 0.36 0.64 1.4
leben1_schule      1 0.43  0.00 -0.01 0.18 0.82 1.0
leben10_familie   10 0.41  0.33  0.06 0.28 0.72 2.0
leben9_familie     9 0.48  0.73 -0.20 0.81 0.19 1.9
leben8_familie     8 0.43  0.57 -0.04 0.51 0.49 1.9
leben6_selbst      5 0.27  0.20  0.70 0.60 0.40 1.5
```

```
leben2_selbst      4 0.27  0.06  0.59 0.43 0.57 1.4
leben5_freunde     6 0.37  0.14  0.40 0.32 0.68 2.2
leben7_freunde     7 0.23  0.16  0.30 0.17 0.83 2.5

                        ML1  ML2  ML3
SS loadings            2.09 1.29 1.15
Proportion Var         0.21 0.13 0.12
Cumulative Var         0.21 0.34 0.45
Proportion Explained   0.46 0.28 0.25
Cumulative Proportion  0.46 0.75 1.00

Mean item complexity =  1.7
Test of the hypothesis that 3 factors are sufficient.

df null model =  45  with the objective function =  2.29 with Chi Square =  571.23
df of  the model are 18  and the objective function was  0.16

The root mean square of the residuals (RMSR) is  0.04
The df corrected root mean square of the residuals is  0.06

The harmonic n.obs is  255 with the empirical chi square  36.34  with prob <  0.0064
The total n.obs was  255  with Likelihood Chi Square =  39.98  with prob <  0.0021

Tucker Lewis Index of factoring reliability =  0.895
RMSEA index =  0.069  and the 90 % confidence intervals are  0.04 0.098
BIC =  -59.76
Fit based upon off diagonal values = 0.97
Measures of factor score adequacy
                                                 ML1  ML2  ML3
Correlation of (regression) scores with factors  0.95 0.92 0.84
Multiple R square of scores with factors         0.89 0.84 0.71
Minimum correlation of possible factor scores    0.79 0.68 0.42
```

Insgesamt können mit drei Faktoren 45 % der Gesamtvarianz erklärt werden. Die Eigenwerte entsprechen (anders als bei der PCA) hier übrigens **nicht** denen der Parallelanalyse. Das hängt damit zusammen, dass die Eigenwerte in der EFA auch von der Anzahl der extrahierten Faktoren abhängen, in der PCA dagegen nicht.

Der Anpassungstest für das Modell ("Test of the hypothesis that 3 factors are sufficient.") zeigt ein $\chi^2 = 39.98, p = 0.002$. Somit muss die Nullhypothese, dass drei Faktoren ausreichen, um die Korrelationen zwischen den Items vollständig zu erklären, abgelehnt werden. Auch die weiteren Fit-Indizes, die wir erst bei der CFA genauer kennenlernen werden, zeigen keinen besonders

guten Fit an. Aus diesem Grund werden wir weiter unten auch noch eine 4-Faktor-Lösung betrachten.

5.4. ML-EFA mit drei rotierten Faktoren

Zunächst betrachten wir aber eine rotierte Lösung für drei Faktoren. Eine orthogonale Rotation lassen wir diesmal aus und betrachten gleich eine schiefwinklige Rotation. Anders als oben bei der PCA benutzen wir hier keine Oblimin-Rotation, sondern eine oblique Geomin-Rotation. Der Grund ist, dass die Oblimin-Rotation im Package `EFAutilities`, das uns weiter unten die ML-geschätzten Standardfehler und Konfidenzintervalle der Ladungen und Faktorkorrelationen liefern soll, nicht enthalten ist. Die oblique Geomin-Rotation liefert aber sehr ähnliche Ergebnisse wie eine Oblimin-Rotation.

```r
# Rotierte EFA mit 3 Faktoren
efa.3.rotiert <- fa(ls_clean, fm = "ml", nfactors = 3, rotate = "geominQ")
print.psych(efa.3.rotiert, sort = TRUE)
```

```
Factor Analysis using method =  ml
Call: fa(r = ls_clean, nfactors = 3, rotate = "geominQ", fm = "ml")
Standardized loadings (pattern matrix) based upon correlation matrix
                item   ML2   ML3   ML1   h2   u2 com
leben9_familie     9  0.92 -0.06 -0.02 0.81 0.19 1.0
leben8_familie     8  0.69  0.08  0.00 0.51 0.49 1.0
leben10_familie   10  0.43  0.16  0.10 0.28 0.72 1.4
leben6_selbst      5  0.01  0.79 -0.09 0.60 0.40 1.0
leben2_selbst      4 -0.08  0.67  0.03 0.43 0.57 1.0
leben5_freunde     6  0.10  0.49  0.10 0.32 0.68 1.2
leben7_freunde     7  0.11  0.36  0.01 0.17 0.83 1.2
leben4_schule      3 -0.04  0.01  0.94 0.87 0.13 1.0
leben3_schule      2  0.05 -0.05  0.60 0.36 0.64 1.0
leben1_schule      1  0.16  0.06  0.34 0.18 0.82 1.5

                       ML2  ML3  ML1
SS loadings           1.61 1.52 1.40
Proportion Var        0.16 0.15 0.14
Cumulative Var        0.16 0.31 0.45
Proportion Explained  0.36 0.33 0.31
Cumulative Proportion 0.36 0.69 1.00

 With factor correlations of
     ML2  ML3  ML1
```

```
ML2 1.00 0.28 0.19
ML3 0.28 1.00 0.23
ML1 0.19 0.23 1.00

Mean item complexity =  1.1
Test of the hypothesis that 3 factors are sufficient.

df null model =  45  with the objective function =  2.29 with Chi Square =  571.23
df of  the model are 18  and the objective function was  0.16

The root mean square of the residuals (RMSR) is  0.04
The df corrected root mean square of the residuals is  0.06

The harmonic n.obs is  255 with the empirical chi square  36.34  with prob <  0.0064
The total n.obs was  255  with Likelihood Chi Square =  39.98  with prob <  0.0021

Tucker Lewis Index of factoring reliability =  0.895
RMSEA index =  0.069  and the 90 % confidence intervals are  0.04 0.098
BIC =  -59.76
Fit based upon off diagonal values = 0.97
Measures of factor score adequacy
                                                ML2  ML3  ML1
Correlation of (regression) scores with factors   0.92 0.87 0.94
Multiple R square of scores with factors          0.86 0.75 0.88
Minimum correlation of possible factor scores     0.71 0.50 0.77
```

Wie bei der PCA ergeben sich hier ein Familien-Faktor (ML2), eine Selbst- und Freunde-Faktor (ML3) und ein Schul-Faktor (ML1). Die Faktorkorrelationen sind ähnlich wie bei der PCA relativ gleichmässig: ($r_{ML_1ML_2} = 0.19, r_{ML_1ML_3} = 0.23, r_{ML_2ML_3} = 0.28$).

5.4.1. Konfidenzintervalle und Standardfehler

Mit der Funktion `fa()` aus `psych` ist es nicht möglich, Standardfehler und Konfidenzintervalle für die Ladungen einer Maximum-Likelihood-EFA zu berechnen. Mit der Funktion `efa()` aus dem Package `EFAutilities` ist dies dagegen möglich. Wie oben werden mit der ML-Methode (`fm = "ml"`) 3 Faktoren (`factors = 3`) extrahiert und dann eine oblique Geomin-Rotation durchgeführt (`rtype = "oblique", rotation = "geomin"`). Ausserdem werden mit `LConfid = c(.95, .90)` zwei Konfidenzkoeffizienten spezifiziert: der erste (0.95) bezieht sich auf die Konfidenzintervalle der Faktorladungen und -korrelationen, der zweite (0.90) ausschliesslich auf den Fit-Index RMSEA, den wir erst beim Thema CFA genauer betrachten. Alle uns hier interessierenden Konfidenzintervalle sind also 95 %-Konfidenzintervalle.

Da der Output von `EFAutilities::efa()` unübersichtlich ist und wir uns hier nur für das Feature "Standardfehler und CIs" der Ladungen interessieren, haben wir den Output so formatiert, dass eine Tabelle mit den Ladungen zusammen mit ihren Standardfehlern (SEs) sowie eine weitere mit den Ladungen zusammen mit ihren Konfidenzintervallen (CIs) ausgegeben wird:

```
efa.ci.rotated <- EFAutilities::efa(ls_clean,
  fm = "ml", factors = 3,
  rtype = "oblique",
  rotation = "geomin",
  LConfid = c(.95, .90)
)
```

Tabelle 5.1.: Ladungen mit Standardfehlern

	F1	F2	F3
leben1_schule	0.34 [se=0.07]	0.16 [se=0.08]	0.06 [se=0.07]
leben3_schule	0.6 [se=0.09]	0.05 [se=0.06]	-0.05 [se=0.06]
leben4_schule	0.94 [se=0.11]	-0.04 [se=0.02]	0.01 [se=0.02]
leben2_selbst	0.03 [se=0.04]	-0.08 [se=0.06]	0.67 [se=0.06]
leben6_selbst	-0.09 [se=0.05]	0.01 [se=0.02]	0.79 [se=0.06]
leben5_freunde	0.1 [se=0.06]	0.1 [se=0.07]	0.49 [se=0.08]
leben7_freunde	0.01 [se=0.06]	0.11 [se=0.08]	0.36 [se=0.09]
leben8_familie	0 [se=0.04]	0.69 [se=0.07]	0.08 [se=0.06]
leben9_familie	-0.02 [se=0.03]	0.92 [se=0.07]	-0.06 [se=0.03]
leben10_familie	0.1 [se=0.06]	0.43 [se=0.06]	0.16 [se=0.07]

Tabelle 5.2.: Ladungen mit Konfidenzintervallen

	F1	F2	F3
leben1_schule	0.34 [0.197; 0.484]	0.16 [0.01; 0.319]	0.06 [-0.087; 0.199]
leben3_schule	0.6 [0.418; 0.777]	0.05 [-0.056; 0.162]	-0.05 [-0.167; 0.063]
leben4_schule	0.94 [0.725; 1.148]	-0.04 [-0.081; 0.007]	0.01 [-0.035; 0.059]
leben2_selbst	0.03 [-0.039; 0.099]	-0.08 [-0.195; 0.027]	0.67 [0.552; 0.785]
leben6_selbst	-0.09 [-0.195; 0.013]	0.01 [-0.032; 0.043]	0.79 [0.666; 0.914]
leben5_freunde	0.1 [-0.018; 0.226]	0.1 [-0.039; 0.237]	0.49 [0.331; 0.652]
leben7_freunde	0.01 [-0.12; 0.133]	0.11 [-0.043; 0.263]	0.36 [0.192; 0.532]
leben8_familie	0 [-0.078; 0.069]	0.69 [0.548; 0.833]	0.08 [-0.044; 0.197]
leben9_familie	-0.02 [-0.07; 0.032]	0.92 [0.788; 1.052]	-0.06 [-0.114; -0.01]
leben10_familie	0.1 [-0.019; 0.224]	0.43 [0.307; 0.545]	0.16 [0.013; 0.303]

In beiden Tabelle ist jetzt F1 der **Schul**-Faktor, F2 der **Familien**-Faktor, und F3 der **Selbst- und Freunde**-Faktor. Die Konfidenzintervalle aller Ladungen von Items auf einem *zugehörigen* Faktor sollten den Wert 0 *nicht* beinhalten und damit **signifikant** sein, während die Konfidenzintervalle aller Ladungen von Items auf einem *nicht zugehörigen* Faktor den Wert 0 möglichst beinhalten sollten und damit **nicht signifikant** sein. Das ist auch weitgehend der Fall: Ausnahmen sind nur die Querladungen von `leben1_schule` auf F2 (Familie), `leben10_familie` auf F3 (Selbst und Freunde) sowie die sehr kleine (und negative) Querladung von `leben9_familie` auf F3 (Selbst und Freunde). Die CIs dieser Ladungen beinhalten den Wert 0 *nicht*, sind daher also als signifikant zu werten.

Die folgende Tabelle stellt die Faktorkorrelationen mit den zugehörigen Konfidenzintervallen dar. Alle Faktorkorrelationen sind signifikant, da das CI den Wert 0 jeweils nicht enthält.

Anmerkung: Wegen der Darstellung in Matrixform sind 1) alle Korrelationen der Faktoren F1, F2, und F3 untereinander doppelt enthalten; sowie 2) die Korrelationen der Faktoren mit sich selbst enthalten (alle gleich 1, mit CIs [1; 1]).

Tabelle 5.3.: Faktor-Korrelationen mit Konfidenzintervallen

	F1	F2	F3
F1	1 [1; 1]	0.19 [0.052; 0.329]	0.23 [0.083; 0.375]
F2	0.19 [0.052; 0.329]	1 [1; 1]	0.28 [0.13; 0.412]
F3	0.23 [0.083; 0.375]	0.28 [0.13; 0.412]	1 [1; 1]

Die direkte Ausgabe der Standardfehler sowie der Unter- und Obergrenzen der Konfidenzintervalle der Faktorladungen und -korrelationen erhält man folgendermassen:

```
efa.ci.rotated$rotatedse # Standardfehler der rotierten Ladungen
```

```
            F1         F2         F3
MV1  0.07322554 0.07896028 0.07279309
MV2  0.09157621 0.05556776 0.05878569
MV3  0.10780396 0.02246087 0.02412278
MV4  0.03522485 0.05666072 0.05938855
MV5  0.05311170 0.01920790 0.06309381
MV6  0.06231546 0.07030587 0.08196293
MV7  0.06442134 0.07815184 0.08697373
MV8  0.03744344 0.07262669 0.06157491
MV9  0.02606356 0.06726436 0.02650636
MV10 0.06194039 0.06048471 0.07405735
```

```
efa.ci.rotated$rotatedlow # Untere Grenzen CIs der rotierten Ladungen
```

```
             F1           F2          F3
MV1   0.19676509  0.009500119 -0.08670372
MV2   0.41834568 -0.056311443 -0.16718855
MV3   0.72500345 -0.081483388 -0.03539032
MV4  -0.03939991 -0.194653592  0.55190872
MV5  -0.19524613 -0.031847705  0.66623737
MV6  -0.01838522 -0.038737504  0.33080255
MV7  -0.11986408 -0.043241628  0.19155936
MV8  -0.07784971  0.548365993 -0.04430160
MV9  -0.06987032  0.788381701 -0.11380595
MV10 -0.01853494  0.307443979  0.01287337
```

```
efa.ci.rotated$rotatedupper # Obere Grenzen CIs der rotierten Ladungen
```

```
            F1          F2          F3
MV1  0.48380394 0.319018744  0.198639952
MV2  0.77731783 0.161510191  0.063247116
MV3  1.14758719 0.006561603  0.059169224
MV4  0.09867898 0.027452350  0.784707563
MV5  0.01294790 0.043445871  0.913560575
MV6  0.22588691 0.236856427  0.652091332
MV7  0.13266293 0.263107970  0.532490095
MV8  0.06892589 0.833057377  0.197067593
```

```
MV9  0.03229695 1.052053166 -0.009902929
MV10 0.22426692 0.544539683  0.303172836
```

```
efa.ci.rotated$Philow # Untere Grenzen CIs der Faktorkorrelationen
```

```
           F1         F2         F3
F1 1.00000000 0.05174141 0.08306906
F2 0.05174141 1.00000000 0.13040653
F3 0.08306906 0.13040653 1.00000000
```

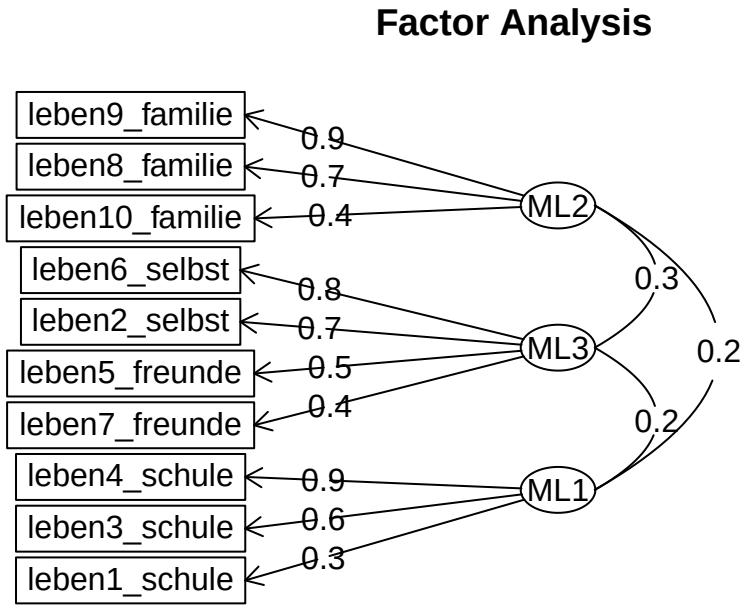
```
efa.ci.rotated$Phiupper # Obere Grenzen CIs der Faktorkorrelationen
```

```
          F1        F2        F3
F1 1.0000000 0.3293499 0.3748086
F2 0.3293499 1.0000000 0.4117507
F3 0.3748086 0.4117507 1.0000000
```

Visualisierung:

Welche Items laden auf welche Faktoren? Hier sieht man auch die Korrelationen zwischen den Faktoren:

```
fa.diagram(efa.3.rotiert, cut = 0)
```



5.5. ML-EFA mit vier unrotierten Faktoren

Unrotierte exploratorische Faktorenanalyse mit 4 Faktoren:

```
efa.4.unrotiert <- fa(ls_clean, fm = "ml", nfactors = 4, rotate = "none")
print.psych(efa.4.unrotiert, sort = TRUE)
```

```
Factor Analysis using method =  ml
Call: fa(r = ls_clean, nfactors = 4, rotate = "none", fm = "ml")
Standardized loadings (pattern matrix) based upon correlation matrix
                item  ML1   ML3   ML4   ML2   h2    u2 com
leben5_freunde     6 0.83 -0.01  0.00  0.56 1.00 0.005 1.8
leben6_selbst      5 0.83  0.00  0.00 -0.56 1.00 0.005 1.8
leben2_selbst      4 0.52  0.01  0.11 -0.16 0.31 0.687 1.3
leben7_freunde     7 0.42  0.07 -0.04  0.13 0.20 0.799 1.3
leben9_familie     9 0.21  0.77 -0.42  0.06 0.81 0.188 1.7
leben8_familie     8 0.25  0.60 -0.29 -0.01 0.51 0.492 1.8
leben10_familie   10 0.28  0.44 -0.09 -0.03 0.28 0.723 1.8
leben1_schule      1 0.19  0.33  0.21  0.03 0.19 0.808 2.4
leben4_schule      3 0.19  0.47  0.73  0.12 0.80 0.198 2.0
leben3_schule      2 0.06  0.40  0.48  0.06 0.39 0.605 2.0

                       ML1  ML3  ML4  ML2
SS loadings           2.08 1.64 1.08 0.69
Proportion Var        0.21 0.16 0.11 0.07
Cumulative Var        0.21 0.37 0.48 0.55
Proportion Explained  0.38 0.30 0.20 0.13
Cumulative Proportion 0.38 0.68 0.87 1.00

Mean item complexity =  1.8
Test of the hypothesis that 4 factors are sufficient.

df null model =  45  with the objective function =  2.29 with Chi Square =  571.23
df of  the model are 11  and the objective function was  0.06

The root mean square of the residuals (RMSR) is  0.02
The df corrected root mean square of the residuals is  0.05

The harmonic n.obs is  255 with the empirical chi square  12.53  with prob <  0.33
The total n.obs was  255  with Likelihood Chi Square =  14.32  with prob <  0.22

Tucker Lewis Index of factoring reliability =  0.974
```

```
RMSEA index =  0.034  and the 90 % confidence intervals are  0 0.079
BIC =  -46.63
Fit based upon off diagonal values = 0.99
Measures of factor score adequacy
                                                 ML1  ML3  ML4  ML2
Correlation of (regression) scores with factors   1.00 0.92 0.90 1.00
Multiple R square of scores with factors          1.00 0.85 0.81 0.99
Minimum correlation of possible factor scores     0.99 0.70 0.62 0.98
```

Insgesamt können mit vier Faktoren 55 % der Gesamtvarianz erklärt werden. Der Anpassungstest für das Modell ("Test of the hypothesis that 4 factors are sufficient.") zeigt ein $\chi^2 = 14.32, p = 0.22$. Somit kann die Nullhypothese, dass vier Faktoren ausreichen, beibehalten werden. Auch die weiteren Fit-Indizes sind deutlich besser als bei der 3-Faktor-Lösung.

Allerdings zeigt sich hier ein sogenannter "Heywood case". Als solcher wird eine negative geschätzte Varianz oder eine geschätzte Korrelation > 1 bezeichnet, die bei instabilen Faktorstrukturen auftreten können. Instabile Faktorstrukturen können sich in der EFA insbesondere dann ergeben, wenn einer oder mehrere Faktoren nur durch zwei Items repräsentiert sind. Das ist hier der Fall (s.u.).

5.5.1. ML-EFA mit vier rotierten Faktoren

Geomin-rotierte exploratorische Faktorenanalyse mit 4 Faktoren:

```r
efa.4.rotiert <- fa(ls_clean, fm = "ml", nfactors = 4, rotate = "geominQ")
print.psych(efa.4.rotiert, sort = TRUE)
```

```
Factor Analysis using method =  ml
Call: fa(r = ls_clean, nfactors = 4, rotate = "geominQ", fm = "ml")
Standardized loadings (pattern matrix) based upon correlation matrix
                item   ML3   ML4   ML1   ML2   h2    u2 com
leben9_familie     9  0.92 -0.02 -0.07  0.00 0.81 0.188 1.0
leben8_familie     8  0.70  0.01  0.05  0.00 0.51 0.492 1.0
leben10_familie   10  0.43  0.12  0.13  0.02 0.28 0.723 1.4
leben4_schule      3 -0.04  0.90  0.00  0.03 0.80 0.198 1.0
leben3_schule      2  0.05  0.63 -0.04 -0.06 0.39 0.605 1.0
leben1_schule      1  0.16  0.35  0.06  0.03 0.19 0.808 1.5
leben6_selbst      5  0.02 -0.03  1.00 -0.02 1.00 0.005 1.0
leben2_selbst      4 -0.04  0.09  0.47  0.15 0.31 0.687 1.3
leben5_freunde     6  0.00  0.00 -0.01  1.00 1.00 0.005 1.0
leben7_freunde     7  0.09 -0.01  0.12  0.35 0.20 0.799 1.4
```

```
                        ML3  ML4  ML1  ML2
SS loadings            1.59 1.37 1.32 1.20
Proportion Var         0.16 0.14 0.13 0.12
Cumulative Var         0.16 0.30 0.43 0.55
Proportion Explained   0.29 0.25 0.24 0.22
Cumulative Proportion  0.29 0.54 0.78 1.00

 With factor correlations of
     ML3  ML4  ML1  ML2
ML3 1.00 0.20 0.21 0.25
ML4 0.20 1.00 0.13 0.23
ML1 0.21 0.13 1.00 0.40
ML2 0.25 0.23 0.40 1.00

Mean item complexity =  1.2
Test of the hypothesis that 4 factors are sufficient.

df null model =  45  with the objective function =  2.29 with Chi Square =  571.23
df of  the model are 11  and the objective function was  0.06

The root mean square of the residuals (RMSR) is  0.02
The df corrected root mean square of the residuals is  0.05

The harmonic n.obs is  255 with the empirical chi square  12.53  with prob <  0.33
The total n.obs was  255  with Likelihood Chi Square =  14.32  with prob <  0.22

Tucker Lewis Index of factoring reliability =  0.974
RMSEA index =  0.034  and the 90 % confidence intervals are  0 0.079
BIC =  -46.63
Fit based upon off diagonal values = 0.99
Measures of factor score adequacy
                                                ML3  ML4  ML1  ML2
Correlation of (regression) scores with factors   0.92 0.91 1.00 1.00
Multiple R square of scores with factors          0.85 0.83 0.99 1.00
Minimum correlation of possible factor scores     0.71 0.67 0.99 0.99
```

Die Lösung zeigt eine klare Einfachstruktur. Keine Ladung auf einem nicht-zugehörigen Faktor ist > 0.16. Die Einfachstruktur deckt sich auch mit der inhaltlichen Ausrichtung der Lebenszufriedenheitsbereiche: jetzt haben alle Bereiche (Familie, Schule, Selbst, Freunde) ihren eigenen Faktor.

Die Faktorkorrelationen der vier Faktoren sind jetzt nicht mehr so gleichmässig wie noch bei

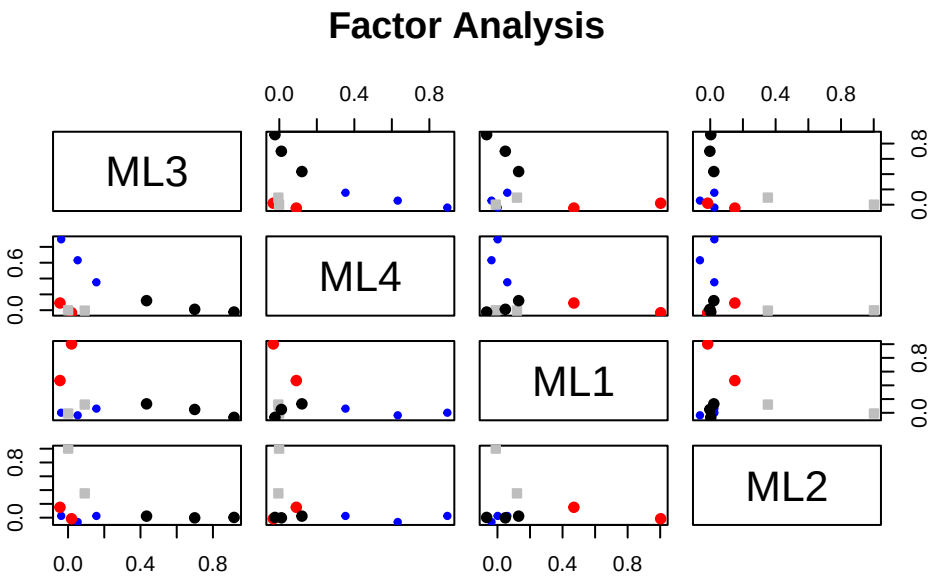
der 3-Komponenten-Lösung: die grösste Korrelation zeigt sich zwischen dem **Selbst**- und dem **Freunde**-Faktor: $r_{ML_1 ML_2} = 0.40$ und die geringste zwischen dem **Selbst**- und dem **Schul**-Faktor: $r_{ML_1 ML_4} = 0.13$.

Auf die zusätzliche Berechnung von Standardfehlern und Konfidenzintervallen mit `EFAutilities::efa()` verzichten wir hier.

Visualisierung:

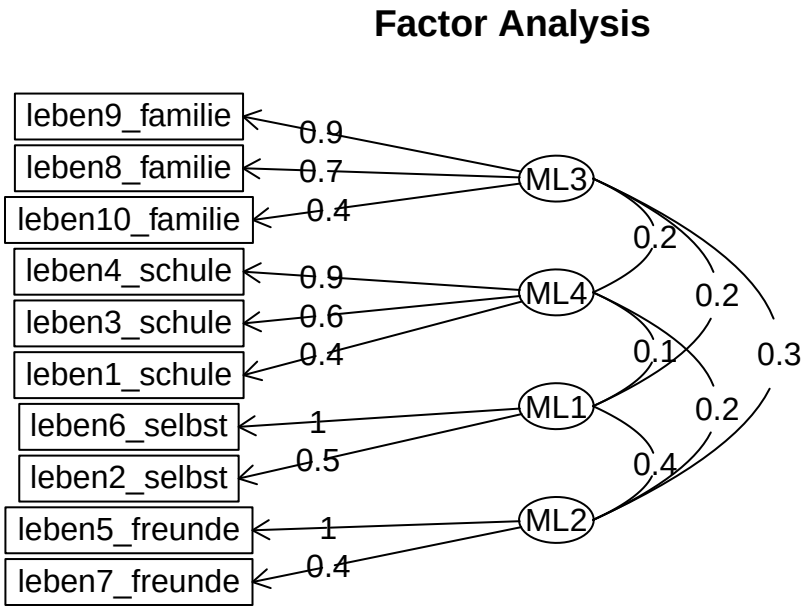
Faktorladung je Item graphisch dargestellt:

```
factor.plot(efa.4.rotiert)
```



Welche Items laden auf welche Faktoren?

```
fa.diagram(efa.4.rotiert, cut = 0)
```



5.6. Zusammenfassung ML-EFA

In der ML-EFA zeigten sich laut Parallelanalyse auch *drei* Faktoren. Allerdings ist der Chi-Quadrat-Test zur Frage, ob drei Faktoren ausreichen, signifikant. Dies spricht für eine 4-Faktor-Lösung, bei der sich der Selbst-Freunde-Faktor in zwei Faktoren aufspaltet, die die Inhaltsbereiche *Selbst* und *Freunde* repräsentieren. Obwohl sich Selbst- und Freunde-Items also viel Varianz teilen und die Parallelanalyse deren Gruppierung auf einem Faktor bevorzugt, müssen diese Inhaltsbereiche im Sinne einer vollständigen Repräsentation der Zusammenhänge separat betrachtet werden.

5.7. Übung

Datenframe mit 25 Persönlichkeits-Items und die drei Variablen `gender` (Geschlecht), `education` (Ausbildung) und `age` (Alter). Die Daten sind im Package `psych` gespeichert und können mit folgendem Chunk geladen werden:

```
data("bfi", package = "psych")
head(bfi)
```

```
      A1 A2 A3 A4 A5 C1 C2 C3 C4 C5 E1 E2 E3 E4 E5 N1 N2 N3 N4 N5 O1 O2 O3 O4
61617  2  4  3  4  4  2  3  3  4  4  3  3  3  4  4  3  4  2  2  3  3  6  3  4
```

```
61618  2  4  5  2  5  5  4  4  3  4  1  1  6  4  3  3  3  3  5  5  4  2  4  3
61620  5  4  5  4  4  4  5  4  2  5  2  4  4  4  5  4  5  4  2  3  4  2  5  5
61621  4  4  6  5  5  4  4  3  5  5  5  3  4  4  4  2  5  2  4  1  3  3  4  3
61622  2  3  3  4  5  4  4  5  3  2  2  2  5  4  5  2  3  4  4  3  3  3  4  3
61623  6  6  5  6  5  6  6  6  1  3  2  1  6  5  6  3  5  2  2  3  4  3  5  6
      O5 gender education age
61617  3      1        NA  16
61618  3      2        NA  18
61620  2      2        NA  17
61621  5      2        NA  17
61622  3      1        NA  17
61623  1      2         3  21
```

Nehmen wir an, wir hätten vergessen, wie viele und welche Persönlichkeitsdimensionen mit dem Persönlichkeitstest abgefragt wurden. Oder noch besser: Wir stellen uns vor, wir wären die Entwicklerinnen und Entwickler eines neuen Persönlichkeitsmodells und hätten uns diese 25 Items einfallen lassen, deren Dimensionaliät wir nun gerne mit einer exploratorische Faktorenanalyse untersuchen möchten.

Wir haben natürlich eine gewisse inhaltliche Vorstellung (und haben den Items deshalb verschiedene Präfixe - A, C, E, N, O für die Variablennamen verordnet) und nehmen an, dass die Items durch weniger zugrunde liegende Dimensionen repräsentiert werden können. Mit anderen Worten: dass die Interkorrelationen der Items auf ihre Zugehörigkeit zu weniger Dimensionen (als Items) zurückgeführt werden können (evtl. 5?).

Aufgabe 1

a) Wir haben genügend theoretischen Hintergrund anzunehmen, dass eine EFA mit diesen Daten angebracht ist. Testen Sie trotzdem erst auf Sphärizität und berechnen Sie das Kaiser-Meyer-Olkin Measure of Sampling Adequacy (KMO-MSA). Interpretieren Sie die Resultate. *Tipp: Verwenden Sie nur die 25 Columns der Persönlichkeits-Items.*

Lösung

Bartlett Test und KMO-MSA

```
# Zuerst speichern wir ein neues Datenframe mit den 25 Items
bfi25 <- bfi %>%
  select(-gender, -education, -age)
cortest.bartlett(bfi25)
```

```
R was not square, finding R from data
```

```
$chisq
[1] 20163.79

$p.value
[1] 0

$df
[1] 300
```

```
KMO(bfi25)
```

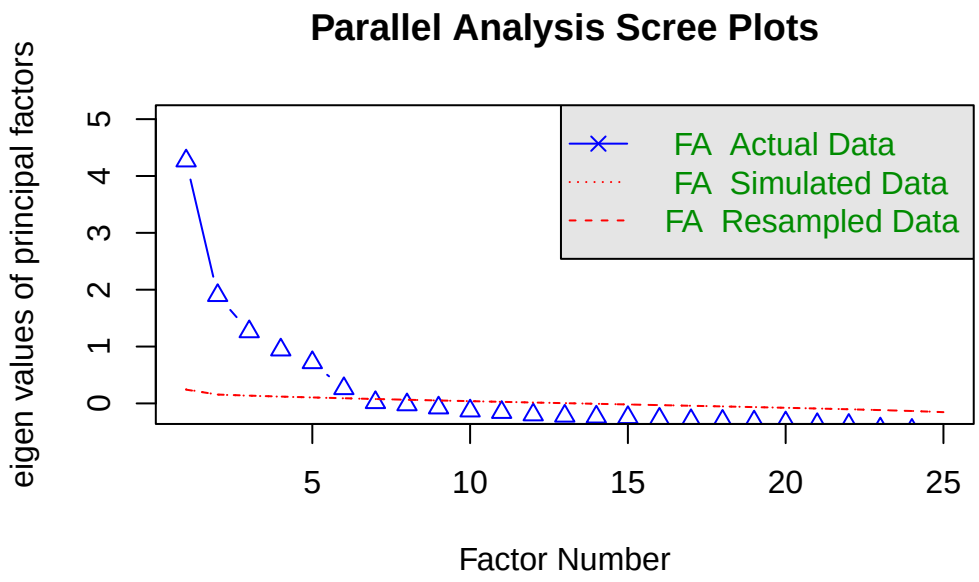
```
Kaiser-Meyer-Olkin factor adequacy
Call: KMO(r = bfi25)
Overall MSA =  0.85
MSA for each item =
  A1   A2   A3   A4   A5   C1   C2   C3   C4   C5   E1   E2   E3   E4   E5   N1
0.74 0.84 0.87 0.87 0.90 0.83 0.79 0.85 0.82 0.86 0.83 0.88 0.89 0.87 0.89 0.78
  N2   N3   N4   N5   O1   O2   O3   O4   O5
0.78 0.86 0.88 0.86 0.85 0.78 0.84 0.76 0.76
```

Bartlett's Test auf Sphärizität deutet stark darauf hin, dass die Items eine Korrelationsstruktur haben. Das Bedeutet, dass sie miteinander zusammenhängen. Mit einem Overall KMO-MSA von 0.85 haben wir eine gute KMO-MSA. Beide Werte deuten darauf hin, dass eine EFA gerechtfertigt ist.

b) Führen Sie eine Parallelanalyse durch, um zu ermitteln, wie viele Faktoren für die exploratorische Faktorenanalyse benötigt werden, um die Interkorrelationen der Items zu erklären.

Lösung

```
fa.parallel(bfi25, fa = "fa", fm = "ml", n.iter = 1000, quant = 0.5)
```




```
Parallel analysis suggests that the number of factors =  6  and the number of components =
```

Die Ergebnisse der Parallelanalyse zeigen, dass wir 6 latente Faktoren berücksichtigen sollten, die den Interkorrelationen der Items zu Grunde liegen.

Aufgabe 2

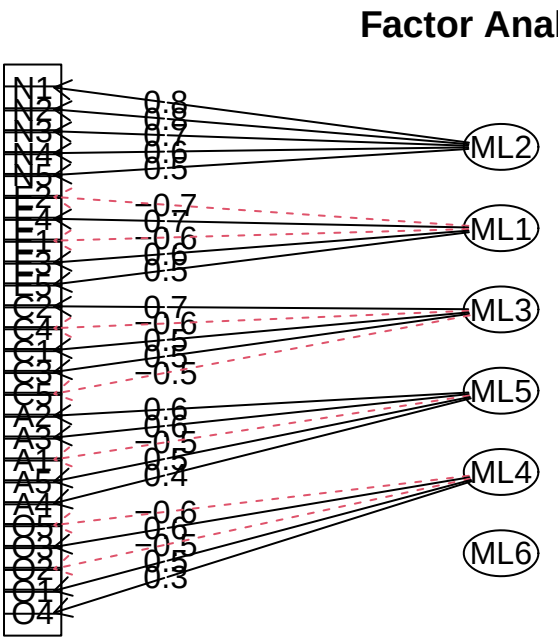
Rechnen Sie eine EFA der entsprechenden Anzahl Faktoren (aus *Aufgabe b*). Berechnen Sie diese gleich mit einer orthogonalen Rotation (eine unrotierte Lösung wird nicht zwingend benötigt). Welche Items werden welchem Faktor zugeordnet?

Visualisieren Sie dazu die EFA mit der Funktion `fa.diagram()`.

Wofür stehen die *roten* Pfeile im Diagramm?

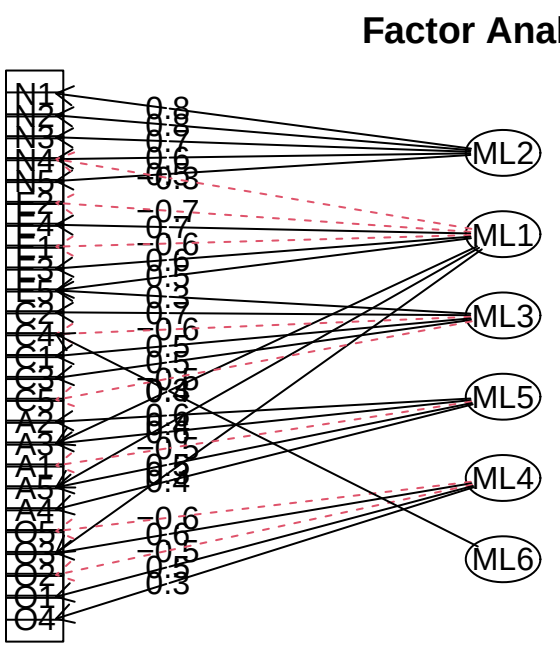
Lösung

```r
# Wir wählen als orthogonale Rotation die varimax Rotation
bfi.fa6 <- fa(bfi25, fm = "ml", nfactors = 6, rotate = "varimax")
fa.diagram(bfi.fa6, sort = T)
```



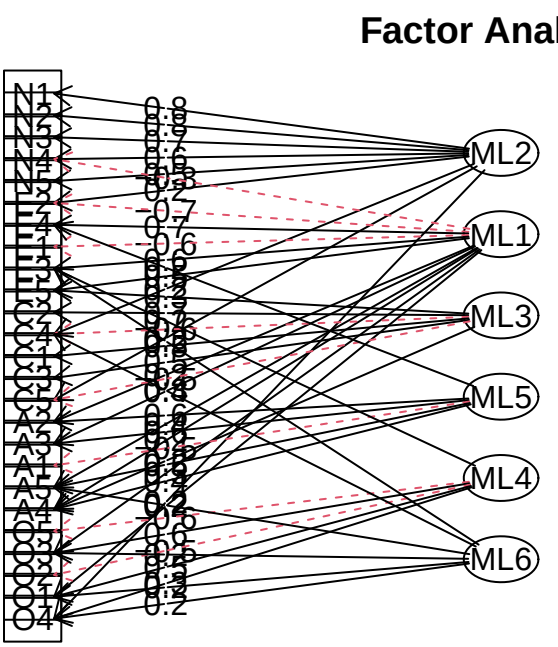
Die roten Pfeile im Diagramm stehen für inhaltlich umgekehrt formulierte Items. Diese Laden negativ auf dem entsprechenden Faktor. Wir gehen jedenfalls schwer davon aus, dass dem so ist. Streng genommen müssten wir das jetzt noch am Itemwortlaut überprüfen!
Eigenartig: Obwohl die Parallelanalyse 6 Faktoren bevorzugt, weist das Diagramm dem 6. Faktor keine Items zu. Die Voreinstellung von `fa.diagram()` ist, dass nur die grösste Ladung eines Items (also auf nur einem Faktor) dargestellt wird (`simple = TRUE`). Wir können dieses Argumente auch aus `FALSE` setzen, um wenigstens ein bisschen zu sehen, wie die Ladungen auf Faktor 6 aussehen.

```
fa.diagram(bfi.fa6, sort = TRUE, simple = FALSE)
```



Das Ergebnis zeigt nun *eine* Ladung von Faktor 6 an, und zwar auf Item `C4`. Die anderen Ladungen wurden durch die Voreinstellung `cut = 0.3` unterdrückt. Diese Voreinstellung führt dazu, dass grundsätzlich nur Ladungen mit einem Betrag > 0.3 dargestellt werden. Wir könnten nun mit `cut = 0` alle Ladungen (aller Items auf allen Faktoren) anzeigen lassen, dann wäre aber kaum mehr etwas zu erkennen (probieren Sie es aus). Stattdessen setzen wir - damit es nicht total unübersichtlich wird - neben `simple = FALSE` noch `cut = 0.2`.

```
fa.diagram(bfi.fa6, sort = TRUE, simple = FALSE, cut = 0.2)
```



Der Plot soll uns die rotierte Faktorstruktur möglichst im Sinne einer Einfachstruktur darstellen, daher ist `simple = TRUE` sehr sinnvoll. Es ist aber auch gut zu wissen, dass es nur wenige Querladungen der Faktoren 1-5 auf den anderen Faktoren gibt. Das konnten wir nur mit `simple = FALSE` veranschaulichen. Der 6. Faktor scheint irgendwie überflüssig zu sein: es gibt nur ein Item, dessen Ladungsbetrag > 0.3 ist (und dieses lädt gerade mal 0.31, wie Sie sehen können, wenn Sie sich die vollständigen Ergebnisse der Analyse ausgeben lassen).
Jetzt fällt uns auch (plötzlich) wieder ein, dass es sich um einen Fragebogen für die Big-5 Persönlichkeitseigenschaften handelt! Dieser 6. Eigenwert mag zwar in den Daten relativ gross sein, er entspricht aber keinem interpretierbaren Faktor. Bei der EFA (wie bei der PCA) ist die Interpretierbarkeit der (rotierten) Ladungsstruktur ein wichtiges Kriterium für die Anzahl der zu extrahierenden Faktoren, und es kommt nicht selten vor, dass diesem Kriterium Vorrang vor dem Ergebnis der Parallelanalyse gegeben wird. Auch nach dem visuellen Scree-Plot-Kriterium könnte man hier übrigens ggf. für 5 statt für 6 Faktoren plädieren (wegen des relativ grossen Sprungs vom 5. zum 6. Eigenwert).

Aufgabe 3

Für die Aufgabe 3 gehen wir davon aus, dass die Daten auf 5 Faktoren reduzierbar sind.

a) Ist mit 5 orthogonalen Faktoren eine Einfachstruktur vorhanden?

Lösung

Wir führen also eine weitere EFA mit nur 5 Faktoren durch. Diesmal lassen wir uns mit `print.psych()` alle Ergebnissee ausgeben:

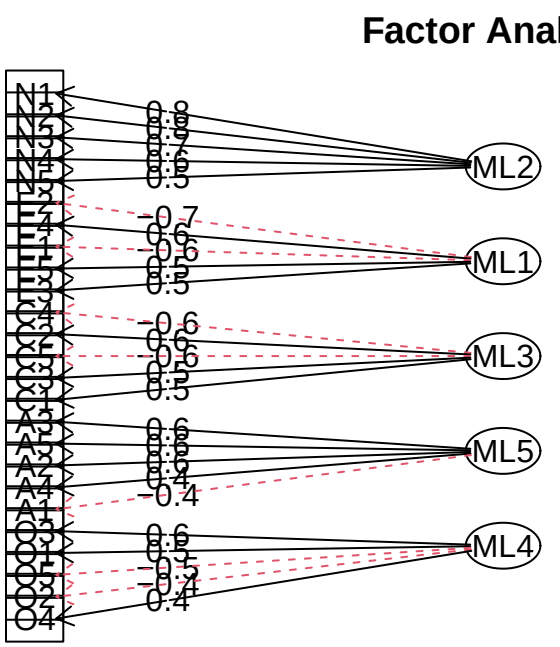
```
bfi.fa5 <- fa(bfi25, fm = "ml", nfactors = 5, rotate = "varimax")
print.psych(bfi.fa5, digits = 3)
```

```
Factor Analysis using method =  ml
Call: fa(r = bfi25, nfactors = 5, rotate = "varimax", fm = "ml")
Standardized loadings (pattern matrix) based upon correlation matrix
      ML2    ML1    ML3    ML5    ML4    h2    u2  com
A1  0.113  0.037  0.002 -0.364 -0.058 0.150 0.850 1.27
A2  0.038  0.179  0.151  0.584  0.065 0.401 0.599 1.37
A3  0.019  0.274  0.107  0.648  0.067 0.511 0.489 1.44
A4 -0.052  0.154  0.233  0.441 -0.102 0.286 0.714 1.97
A5 -0.113  0.340  0.078  0.585  0.080 0.483 0.517 1.79
C1 -0.002  0.041  0.523  0.053  0.207 0.321 0.679 1.34
C2  0.076 -0.001  0.621  0.137  0.130 0.427 0.573 1.22
C3 -0.020  0.006  0.547  0.132  0.004 0.317 0.683 1.12
C4  0.230 -0.086 -0.629 -0.021 -0.094 0.465 0.535 1.36
C5  0.276 -0.187 -0.565 -0.061  0.033 0.435 0.565 1.74
E1  0.033 -0.587  0.038 -0.123 -0.079 0.369 0.631 1.14
E2  0.231 -0.679 -0.094 -0.149 -0.054 0.548 0.452 1.39
E3  0.014  0.482  0.063  0.336  0.301 0.441 0.559 2.58
E4 -0.104  0.601  0.084  0.370 -0.045 0.519 0.481 1.80
E5  0.048  0.486  0.314  0.125  0.228 0.405 0.595 2.40
N1  0.800  0.094 -0.037 -0.218 -0.084 0.705 0.295 1.21
N2  0.782  0.052 -0.024 -0.203 -0.025 0.657 0.343 1.15
N3  0.716 -0.078 -0.081 -0.018  0.004 0.525 0.475 1.05
N4  0.558 -0.356 -0.187 -0.012  0.063 0.478 0.522 1.99
N5  0.522 -0.183 -0.048  0.107 -0.135 0.338 0.662 1.51
O1 -0.003  0.180  0.105  0.096  0.521 0.324 0.676 1.40
O2  0.174 -0.011 -0.113  0.112 -0.434 0.244 0.756 1.62
O3  0.016  0.259  0.070  0.165  0.611 0.473 0.527 1.54
O4  0.216 -0.220 -0.024  0.145  0.375 0.257 0.743 2.67
O5  0.085 -0.010 -0.074  0.013 -0.511 0.274 0.726 1.10

                       ML2   ML1   ML3   ML5   ML4
SS loadings          2.668 2.254 1.967 1.947 1.518
Proportion Var       0.107 0.090 0.079 0.078 0.061
Cumulative Var       0.107 0.197 0.276 0.353 0.414
```

```
Proportion Explained  0.258 0.218 0.190 0.188 0.147
Cumulative Proportion 0.258 0.475 0.665 0.853 1.000

Mean item complexity =  1.6
Test of the hypothesis that 5 factors are sufficient.

df null model =  300  with the objective function =  7.228 with Chi Square =  20163.79
df of  the model are 185  and the objective function was  0.628

The root mean square of the residuals (RMSR) is  0.03
The df corrected root mean square of the residuals is  0.038

The harmonic n.obs is  2762 with the empirical chi square  1474.696  with prob <  1.29e-199
The total n.obs was  2800  with Likelihood Chi Square =  1749.883  with prob <  1.39e-252

Tucker Lewis Index of factoring reliability =  0.8721
RMSEA index =  0.055  and the 90 % confidence intervals are  0.0526 0.0573
BIC =  281.469
Fit based upon off diagonal values = 0.979
Measures of factor score adequacy
                                                 ML2   ML1   ML3   ML5   ML4
Correlation of (regression) scores with factors   0.927 0.867 0.857 0.845 0.824
Multiple R square of scores with factors          0.859 0.752 0.734 0.714 0.679
Minimum correlation of possible factor scores     0.718 0.504 0.468 0.428 0.358
```

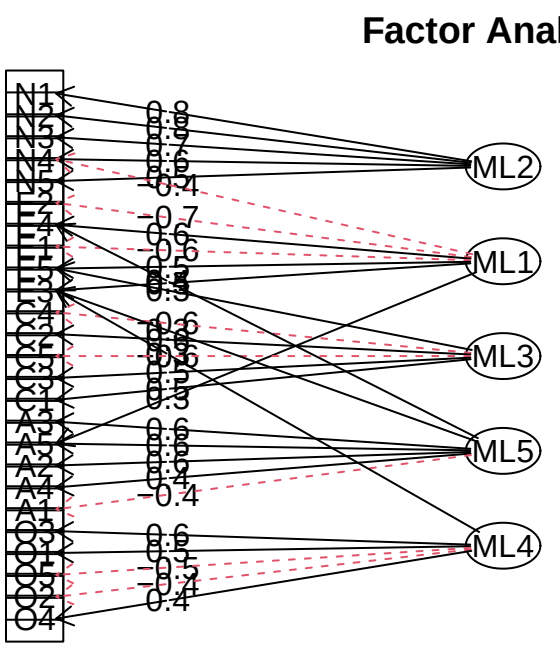
Und das Diagramm mit Einfachstruktur:

```
fa.diagram(bfi.fa5, sort = TRUE)
```



Wie bereits oben korrespondieren die unterschiedlichen Variablen-Präfixe mit jeweils einem Faktor, was auch auf eine klare inhaltliche Strukturierung hinweist (N = Neuroticism, E = Extraversion, C = Conscientiousness, A = Agreeableness, O = Openness). Jetzt schauen wir nochmal mit `simple = FALSE`, wie häufig substantielle Querladungen (hier via default als > 0.3 definiert) auftreten:

```
fa.diagram(bfi.fa5, simple = FALSE, sort = TRUE)
```



Es gibt nur sehr vereinzelte Querladungen und diese liegen alle zwischen 0.3 und 0.4. Letzteres kann man sehen, indem man noch ein Diagramm mit `simple = FALSE` und `cut = 0.4` anfordert - da gibt es dann nämlich keine Querladungen mehr. Alternativ kann man das natürlich auch in der Ladungsmatrix oben sehen.
Korrelationen zwischen den Faktoren werden in den Diagrammen übrigens nicht angezeigt, weil sie wegen der orthogonalen Rotation nicht zugelassen wurden, also per Definition = 0 sind. Eine oblique Rotation würde die Einfachstruktur noch weiter fördern, dann würden sich Korrelationen zwischen den Faktoren ergeben und auch im Diagramm erscheinen. Probieren Sie es aus!

Berechnen Sie ausserdem von Hand folgende Werte:

b) die Kommunalität des Items `C4`

Lösung

Kommunalität von Item `C4`

```r
## Entweder direkt aus dem Objekt gezogen:
# Zeile 9 aus der Ladungsmatrix ist die Zeile von Item C4
c4 <- bfi.fa5$loadings[9, ] # Ladungen von C4 extrahieren
c4_squared <- c4^2 # Ladungen quadrieren
c4_communality <- sum(c4_squared) # Summe ergibt Kommunalität
round(c4_communality, 3) # vgl. mit h2 aus Output!
```

```
[1] 0.465
```

```r
## Oder von Hand aus dem Output oben:
round(c4, 3) # Ladungen anzeigen
```

```
   ML2    ML1    ML3    ML5    ML4
 0.230 -0.086 -0.629 -0.021 -0.094
```

```r
0.230^2 + (-0.086)^2 + (-0.629)^2 + (-0.021)^2 + (-0.094)^2
```

```
[1] 0.465214
```

Die Kommunalität des Items C4 ist 0.465.

c) die Uniqueness des Items O3

Lösung

Uniqueness von Item O3

```r
# Uniqueness ist definiert als: 1 - Kommunalität.
## Direkt aus dem Objekt gezogen (zuerst berechnen wir die Kommunalität):
o3 <- bfi.fa5$loadings[23, ]
o3_squared <- o3^2
o3_communality <- sum(o3_squared)

# Jetzt die Uniqueness:
o3_unique <- 1 - o3_communality
round(o3_unique, 3) # vgl. mit u2 aus Output!
```

```
[1] 0.527
```

```
## Oder von Hand:
round(o3, 3) # Ladungen anzeigen
```

```
  ML2   ML1   ML3   ML5   ML4
0.016 0.259 0.070 0.165 0.611
```

```
1 - (0.016^2 + 0.259^2 + 0.070^2 + 0.165^2 + 0.611^2)
```

```
[1] 0.527217
```

Die Uniqueness des Items `O3` ist 0.527.

d) den (rotierten) Eigenwert von Faktor 3 (ML3)

Lösung

Eigenwert von Faktor 3 (ML3)

```
# Eigenwert ist definiert als: Sum of Squared Loadings (eines Faktors)
## Ladungen für MR3 aus dem Objekt gezogen:
ML3 <- bfi.fa5$loadings[, 3]
ML3_squared <- ML3^2
ML3_eigenvalue <- sum(ML3_squared)
round(ML3_eigenvalue, 3) # vgl. mit SS loadings aus Output!
```

```
[1] 1.967
```

```
## Oder von Hand:
round(ML3, 3) # Ladungen anzeigen
```

```
    A1     A2     A3     A4     A5     C1     C2     C3     C4     C5     E1
 0.002  0.151  0.107  0.233  0.078  0.523  0.621  0.547 -0.629 -0.565  0.038
    E2     E3     E4     E5     N1     N2     N3     N4     N5     O1     O2
-0.094  0.063  0.084  0.314 -0.037 -0.024 -0.081 -0.187 -0.048  0.105 -0.113
    O3     O4     O5
 0.070 -0.024 -0.074
```

```
  0.002^2 + 0.151^2 + 0.107^2 + 0.233^2 + 0.078^2 + 0.523^2 + 0.621^2 +
    0.547^2 + (-0.629)^2 + (-0.565)^2 + 0.038^2 + (-0.094)^2 + 0.063^2 +
    0.084^2 + 0.314^2 + (-0.037)^2 + (-0.024)^2 + (-0.081)^2 + (-0.187)^2 +
    (-0.048)^2 + 0.105^2 + (-0.113)^2 + 0.070^2 + (-0.024)^2 + (-0.074)^2
```

```
[1] 1.968298
```

```
  # Kleiner Rundungsfehler hier!
```

Der Eigenwert (nach Varimax-Rotation) von Faktor `ML3` ist 1.967.

! Wichtig

Beachten Sie, dass diese Berechnungen nur dann voll interpretierbar sind, wenn die Faktoren unkorrelliert sind. Dies ist per Definition bei unrotierten und orthogonal rotierten Faktoren der Fall (also auch bei der hier verwendeten Varimax-Rotation).
Während Kommunalität und Uniqueness eines Items sich nicht zwischen unrotierter und orthogonal rotierter Lösung unterscheiden, unterscheiden sich die Eigenwerte der Faktoren in dieser Hinsicht: in der rotierten Lösung sind sich die Eigenwerte viel ähnlicher als in der unrotierten Anfangslösung, die auf sukzessiver Varianzmaximierung beruht. Die **Summe** der Eigenwerte ist aber invariant zwischen den beiden Lösungen.

Teil III.

Modelle mit latenten Variablen

6. Konfirmatorische Faktorenanalyse (CFA)

6.1. Setup

Das Beispieldaten-Setup für die CFA ist identisch mit dem Setup der Kapitel zu PCA und EFA, bis auf eine kleine Änderung der Variablennamen.

Packages laden, Daten einlesen und aufbereiten:

```
pacman::p_load(tidyverse, ggplot2, ggthemes, psych, haven, EFAutilities, knitr, lavaan, se
```

```
# Daten einlesen
data <- read_sav("https://github.com/methodenlehre/data/blob/master/beispieldaten.sav?raw=

# Datenframe mit nur den Items zur Lebenszufriedenheit erstellen
ls <- data |>
  select(num_range("leben", 1:10)) |>
  drop_na()
```

Die Daten der Lebenszufriedenheit bestehen aus 10 Items, welche sich auf verschiedene Aspekte/Bereiche der Lebenszufriedenheit beziehen. Die Jugendlichen wurden gefragt:

"Wie zufrieden bist Du..."

Tabelle 6.1.

	Fragen
leben1	mit deinen Schulnoten?
leben2	mit deinem Aussehen und deiner Erscheinung?
leben3	mit der Beziehung zu deinen Lehrern?
leben4	mit allem, was mit der Schule zu tun hat?
leben5	mit allem, was mit deiner Beziehung zu anderen Menschen zu tun hat?
leben6	mit deiner Person?
leben7	mit der Beziehung zu deinen Freunden?
leben8	mit der Beziehung zu deinen Eltern?
leben9	mit dem Zusammenleben mit den anderen Familienmitgliedern?
leben10	mit dem Lebensstandard und dem Ansehen deiner Familie?

Geantwortet wurde auf einer 7-stufigen Skala von 1 = "überhaupt nicht zufrieden" bis 7 = "sehr zufrieden".

Die Fragen zur Lebenszufriedenheit können in vier Aspekte/Bereiche gegliedert werden:

Familie: Items 8, 9 und 10

Schule: Items 1, 3 und 4

Selbst: Items 2 und 6

Freunde: Items 7 und 5

Entsprechend ändern wir jetzt die Namen der Variablen, damit die Zuordnung etwas klarer wird. Um die Reihenfolge der Variablen mit ihrer inhaltlichen Ausrichtung in Übereinstimmung zu bringen, ändern wir die Reihenfolge noch mit `select()`.

```
ls <- ls |>
  rename(
    fam1 = leben8,
    fam2 = leben9,
    fam3 = leben10,
    schule1 = leben1,
    schule2 = leben3,
    schule3 = leben4,
    selbst1 = leben2,
    selbst2 = leben6,
    freund1 = leben5,
    freund2 = leben7
```

```
  ) |>
  select(
    schule1, schule2, schule3, selbst1, selbst2,
    freund1, freund2, fam1, fam2, fam3
  )
```

```
summary(ls)
```

```
   schule1         schule2         schule3         selbst1
Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
1st Qu.:4.000   1st Qu.:4.000   1st Qu.:4.000   1st Qu.:5.000
Median :5.000   Median :5.000   Median :5.000   Median :5.000
Mean   :4.754   Mean   :4.797   Mean   :4.435   Mean   :5.214
3rd Qu.:6.000   3rd Qu.:6.000   3rd Qu.:5.000   3rd Qu.:6.000
Max.   :7.000   Max.   :7.000   Max.   :7.000   Max.   :7.000
   selbst2         freund1         freund2          fam1
Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
1st Qu.:5.000   1st Qu.:5.000   1st Qu.:6.000   1st Qu.:5.000
Median :6.000   Median :6.000   Median :6.000   Median :6.000
Mean   :5.678   Mean   :5.594   Mean   :6.217   Mean   :5.833
3rd Qu.:6.000   3rd Qu.:6.000   3rd Qu.:7.000   3rd Qu.:7.000
Max.   :7.000   Max.   :7.000   Max.   :7.000   Max.   :7.000
     fam2            fam3
Min.   :1.000   Min.   :1.000
1st Qu.:5.000   1st Qu.:5.000
Median :6.000   Median :6.000
Mean   :5.627   Mean   :5.808
3rd Qu.:6.000   3rd Qu.:7.000
Max.   :7.000   Max.   :7.000
```

6.1.1. Ausreisser beseitigen

Wie in der Übung zur exploratorischen Faktorenanalyse/Hauptkomponentenanalyse entfernen wir alle Personen aus dem Datensatz, die auf mindestens einer der zehn Lebenszufriedenheitsvariablen einen Wert ausserhalb von +/- 3 SD vom Mittelwert aufweisen. Auch für die ML-Schätzung bei der CFA gilt die Annahme einer multivariaten Normalverteilung, die insbesondere durch starke Ausreisserwerte verletzt werden kann. Durch die Entfernung von Ausreisserwerten wollen wir ausserdem sicherstellen, dass einflussreiche Datenpunkte möglichst keine Rolle für die Faktorlösung spielen.

```r
# Definition einer Funktion `keep`, die nur Datenpunkte auswählt, die zwischen
# +/- 3 Standardabweichungen einer Variablen liegen
keep <- function(x) {
  mx <- mean(x) # Wir speichern den Mittelwert der Variable
  sd3 <- 3 * sd(x) # Hier speichern wir die Standardabweichung * 3
  between(x, left = mx - sd3, right = mx + sd3)
}

# Wir nutzen die Funktion filter(). Wir behalten alle Rows (Personen), die keinen Wert hab
# der stärker vom Variablen-Mittelwert abweicht als 3 sd's.
ls_clean <- ls |>
  filter(rowMeans(sapply(ls, keep)) == 1)
```

6.2. `lavaan`

Lavaan ist ein kostenloses Open-Source-Paket für die latente Variablenmodellierung in R. Man kann Lavaan verwenden, um eine Vielzahl von statistischen Modellen zu schätzen. Zum Beispiel Pfadanalysen, Strukturgleichungsmodelle und eben auch konfirmatorische Faktorenanalysen (CFA).

Der Name `lavaan` kommt von *la*tent *va*riable *an*alysis.

In dieser Übung beschränken wir uns vor allem auf die `cfa()`-Funktion von `lavaan`.

Die Berechnung einer CFA mit `lavaan` besteht aus zwei Schritten. Zuerst muss ein Modell definiert werden, dann kann das Modell mit der `cfa()`-Funktion geschätzt werden. Diese Funktion nimmt als Input unsere Daten und unsere Modelldefinition.

6.2.1. Modell definieren

Modelldefinitionen in `lavaan` folgen alle der selben Syntax.

In der Syntax sind gewisse Zeichen (Operatoren) vordefiniert und es werden auch bestimmte Voreinstellungen vorgenommen, die man ggf. überschreiben muss. Z.B. wird per default die Skalierung der latenten Variablen über die Fixierung (auf den Wert 1) des Ladungsparameters des jeweils ersten Indikators (manifeste Variable) für eine bestimmte latente Variable erreicht. Ausserdem müssen die Ladungsparameter des Modells nicht explizit definiert werden, genauso wenig wie die Varianzparameter der Residualvariablen der manifesten Variablen.

`=~` bedeutet, dass die zu bildende latente Variable (hier z.B. `Faktor1`, Name frei wählbar) links von dem Operator durch alle Variablen rechts davon definiert wird. Die manifesten (gemessenen) Variablen auf der rechten Seite werden mit einem `+` separiert. Diese müssen im Datenframe vorhanden sein.

Ein Beispiel für sechs Items, welche durch zwei Faktoren (latente Variablen) erklärt werden:

```
bsp_model <- "
# Die Reihenfolge, mit der die Faktoren definiert werden, spielt keine Rolle.
Faktor1 =~ var1 + var2 + var3
Faktor2 =~ var4 + var5 + var6
# Die Reihenfolge der manifesten Variablen ist nur für die Fixierung einer Ladung
# pro Faktor von Bedeutung. Hier werden die Ladungen von `var1` und `var4` auf
# den Wert 1 fixiert.

# Zudem haben wir Kommentare im String, die von lavaan ignoriert werden.
"
```

Eine weitere Voreinstellung ist, dass für alle latenten Variablen in einer CFA automatisch eine Faktor-Kovarianz spezifiziert wird. Der Operator für die Spezifikation einer Kovarianz ist `~~`. D.h. wir könnten in diesem Beispiel auch noch eine weitere Zeile mit `Faktor1 ~~ Faktor2` hinzufügen, ohne dass sich an der Modelldefinition etwas ändern würde.

6.2.2. Modell schätzen

Ein Modell kann mit folgender Syntax geschätzt werden: `cfa(model = bsp_model, data = dataframe)`

Die direkte Ausführung dieser Syntax ergibt allerdings nur einen sehr begrenzten Output, der nur die Anzahl der geschätzten Parameter, der Anzahl Beobachtungen sowie die Chi-Quadrat-Statistik enthält.

Daher muss das Ergebnis von `cfa()` zunächst in einem Output-Objekt (z.B. `fit_bsp_model`) abgespeichert und der ausführliche Output (mit den Parameterschätzern) mit `summary(fit_bsp_model)` extrahiert werden. Für die `summary()`-Funktion gibt es einige zusätzliche Argumente: `fit.measures = TRUE` gibt beispielsweise eine Reihe globaler Fitindizes (z.B. SRMR, RMSEA, CLI, TLI) sowie Informationskriterien (z.B. AIC und BIC) aus, und mit `standardized = TRUE` erhalten wir neben den unstandardisierten Parameterschätzern auch die standardisierten Parameterschätzer.

6.3. CFA zu Lebenszufriedenheitsbereichen

In diesem Kapitel fitten wir unterschiedlich komplexe CFA-Modelle auf unsere `ls`-Daten und vergleichen die Modelle anschliessend miteinander. Zunächst schauen wir uns theoriegeleitet ein 4-Faktor-Modell an, dann als Alternative dazu ein (sparsameres, da weniger Parameter schätzendes) 3-Faktor-Modell. In einem Vertiefungsteil schauen wir uns dann noch eine alternative Formulierung des 3-Faktor-Modells an, um zu zeigen, dass das 3-Faktor-Modell im 4-Faktor-Modell genestet ist (und damit ein Modellvergleich mittels LR-Test möglich ist). Die Modelle in diesem Kapitel sind Modelle mit Faktoren erster Ordnung. In Kapitel 6.3.4 betrachten wir dann noch ein Modell mit einem zusätzlichen Gesamt-Lebenszufriedenheitsfaktor zweiter Ordnung.

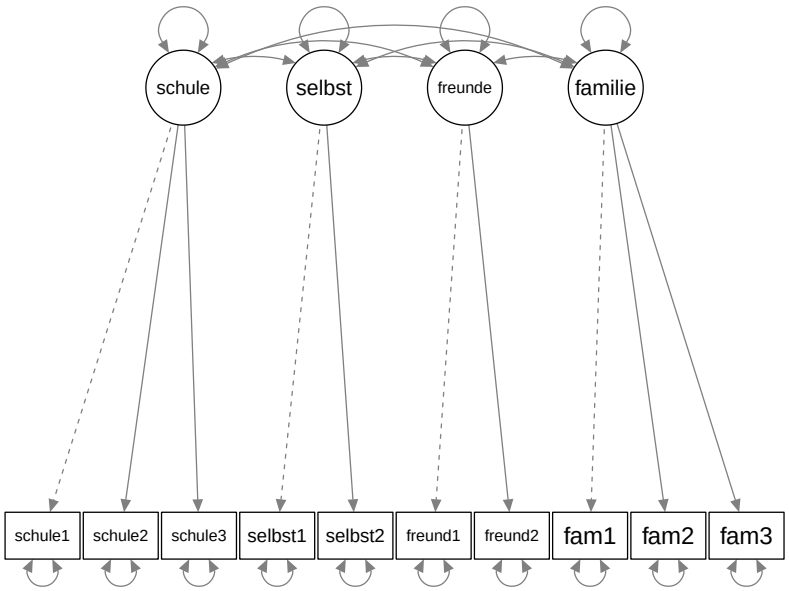
6.3.1. CFA mit vier Faktoren

Aus theoretischen Gründen schätzen wir zunächst ein Modell mit vier Faktoren. Für jeden inhaltlich definierten Lebenszufriedenheitsbereich (Schule, Selbst, Freunde, Familie) definieren wir einen Faktor. Ausserdem postulieren wir wie üblich eine Einfachstruktur, d.h. alle potentiellen Querladungen werden auf den Wert 0 restringiert (indem manifeste Variablen in der Modelldefinition nur in der Gleichung des *zugehörigen* Faktor auftauchen).

6.3.1.1. Modell definieren

```
model_4f <- "
# Der erste Faktor bezieht sich auf den Bereich `schule`
schule =~ schule1 + schule2 + schule3

# Der zweite Faktor bezieht sich auf den Bereich `selbst`
selbst =~ selbst1 + selbst2

# Der dritte Faktor bezieht sich auf den Bereich `freunde`
freunde =~ freund1 + freund2

# Der vierte Faktor bezieht sich auf den Bereich `familie`
familie =~ fam1 + fam2 + fam3
"
```

Das Modell kann man sich folgendermassen vorstellen:



Typische Aufgabe: Schritt für Schritt

In diesem Abschnitt schauen wir uns zusammen die Struktur des CFA-Modells genauer an. Dabei hilft uns die obige Visualisierung des Modells. Fragen zur Struktur des Modells (e.g. Freiheitsgrade des Modells oder Anzahl latenter Variablen) eignen sich sehr gut, um das Verständnis von CFA-Modellen zu überprüfen.

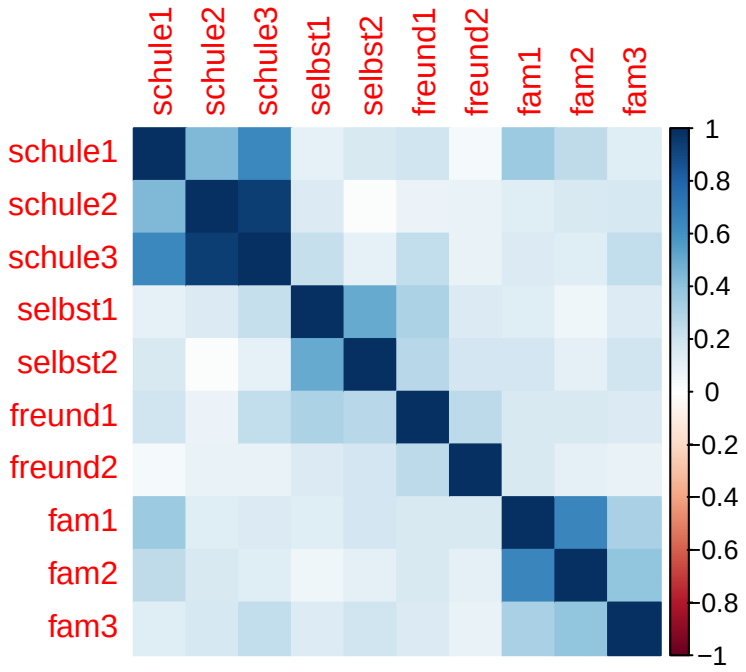
Wie viele und welche manifeste Variablen hat das Modell?

10 manifeste Variablen: schule1, schule2, schule3, selbst1, selbst2, freund1, freund2, fam1, fam2, fam3

Wie viele Informationen (n_{Info}) enthält die Varianz-Kovarianz-Matrix der manifesten Variablen?

$n_{Info} = \frac{p \cdot (p+1)}{2} = \frac{10 \cdot 11}{2} = 55$

Das könnte man auch ganz einfach zählen, wenn man sich die Varianz-Kovarianz Matrix der manifesten Variablen ansieht und dann die Einträge in dieser Matrix zählt.



Beachte dabei, dass wie immer jeder Kovarianz-Wert doppelt vorkommt (die Matrix ist symmetrisch). Wir dürfen aber nur jeweils einen davon zählen. Hier zählen wir das untere Dreieck der Matrix (und die Diagonale, auf der sich die Varianz-Werte befinden).

	schul1	schul2	schul3	slbst1	slbst2	frend1	frend2	fam1	fam2	fam3
schule1	1									
schule2	2	11								
schule3	3	12	20							
selbst1	4	13	21	28						
selbst2	5	14	22	29	35					
freund1	6	15	23	30	36	41				
freund2	7	16	24	31	37	42	46			
fam1	8	17	25	32	38	43	47	50		
fam2	9	18	26	33	39	44	48	51	53	
fam3	10	19	27	34	40	45	49	52	54	55

Wie viele und welche latenten Variablen hat das Modell?

14 latente Variablen: 4 Faktoren (familie, schule, selbst, freunde) und 10 Residualvariablen der manifesten Variablen (ohne Namen)

Wie viele und welche Parameter müssen geschätzt werden? (n_{Par})

6 Faktorladungen

10 Varianzen der Residualvariablen der manifesten Variablen

4 Varianzen der latenten Faktoren

6 Kovarianzen zwischen den latenten Faktoren

$n_{Par} = 6 + 10 + 4 + 6 = 26$

Wie viele Freiheitsgrade besitzt das Modell?

$df = n_{Info} - n_{Par} = 55 - 26 = 29$

6.3.1.2. Modell schätzen

```r
fit_mod4f <- cfa(model_4f, data = ls_clean)
summary(fit_mod4f, fit.measures = TRUE, standardized = TRUE)
```

```
lavaan 0.6.17 ended normally after 40 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        26

  Number of observations                           255

Model Test User Model:

  Test statistic                                51.433
  Degrees of freedom                                29
  P-value (Chi-square)                           0.006

Model Test Baseline Model:

  Test statistic                               583.039
  Degrees of freedom                                45
  P-value                                        0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    0.958
  Tucker-Lewis Index (TLI)                       0.935

Loglikelihood and Information Criteria:
```

```
  Loglikelihood user model (H0)              -3427.760
  Loglikelihood unrestricted model (H1)      -3402.044

  Akaike (AIC)                                6907.520
  Bayesian (BIC)                              6999.593
  Sample-size adjusted Bayesian (SABIC)       6917.166

Root Mean Square Error of Approximation:

  RMSEA                                          0.055
  90 Percent confidence interval - lower         0.029
  90 Percent confidence interval - upper         0.079
  P-value H_0: RMSEA <= 0.050                    0.341
  P-value H_0: RMSEA >= 0.080                    0.045

Standardized Root Mean Square Residual:

  SRMR                                           0.053

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  schule =~
    schule1           1.000                               0.572    0.420
    schule2           1.431    0.257    5.572    0.000    0.819    0.638
    schule3           1.995    0.403    4.947    0.000    1.141    0.858
  selbst =~
    selbst1           1.000                               0.730    0.669
    selbst2           0.940    0.156    6.028    0.000    0.685    0.784
  freunde =~
    freund1           1.000                               0.661    0.791
    freund2           0.614    0.117    5.235    0.000    0.406    0.531
  familie =~
    fam1              1.000                               0.782    0.742
    fam2              1.061    0.129    8.233    0.000    0.830    0.846
    fam3              0.598    0.084    7.136    0.000    0.467    0.510
```

```
Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  schule ~~
    selbst            0.075    0.039    1.947    0.052    0.180    0.180
    freunde           0.111    0.039    2.862    0.004    0.293    0.293
    familie           0.098    0.040    2.427    0.015    0.219    0.219
  selbst ~~
    freunde           0.296    0.059    5.049    0.000    0.614    0.614
    familie           0.142    0.052    2.735    0.006    0.249    0.249
  freunde ~~
    familie           0.178    0.048    3.700    0.000    0.344    0.344

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .schule1           1.528    0.147   10.414    0.000    1.528    0.824
   .schule2           0.978    0.137    7.130    0.000    0.978    0.593
   .schule3           0.469    0.209    2.245    0.025    0.469    0.265
   .selbst1           0.656    0.100    6.575    0.000    0.656    0.552
   .selbst2           0.295    0.076    3.865    0.000    0.295    0.386
   .freund1           0.262    0.079    3.301    0.001    0.262    0.375
   .freund2           0.419    0.047    8.948    0.000    0.419    0.718
   .fam1              0.498    0.079    6.336    0.000    0.498    0.449
   .fam2              0.274    0.077    3.574    0.000    0.274    0.285
   .fam3              0.622    0.061   10.240    0.000    0.622    0.740
    schule            0.327    0.109    3.001    0.003    1.000    1.000
    selbst            0.532    0.120    4.452    0.000    1.000    1.000
    freunde           0.437    0.095    4.594    0.000    1.000    1.000
    familie           0.611    0.109    5.593    0.000    1.000    1.000
```

Interpretation des Modells

Eine korrekte Interpretation des `R` Outputs ist ein wichtiger Bestandteil der Lernziele für die CFA.

Wie sehen die Faktorladungen aus? Sind alle signifikant?

Alle Faktorladungen sind signifikant ($p < 0.001$). Die standardisierten Ladungen (`Std.all`) schwanken zwischen 0.420 (`schule =~ schule1`) und 0.858 (`schule =~ schule3`).

Sind die Kovarianzen/Korrelationen zwischen den Faktoren signifikant?

Bis auf `schule ~~ selbst` ($p = 0.052$) sind alle Kovarianzen signifikant. Die stärkste Kovarianz/Korrelation findet sich mit 0.296/0.614 zwischen `freunde ~~ selbst`, also zwischen den beiden Faktoren, die in der EFA zusammen auf einem Faktor geladen haben.

Sind die Faktorvarianzen sowie Varianzen der Residualvariablen signifikant?

Ja, alle Residualvarianzen sind signifikant ($p < 0.05$).

Wie gross sind die Kommunalitäten der manifesten Variablen?

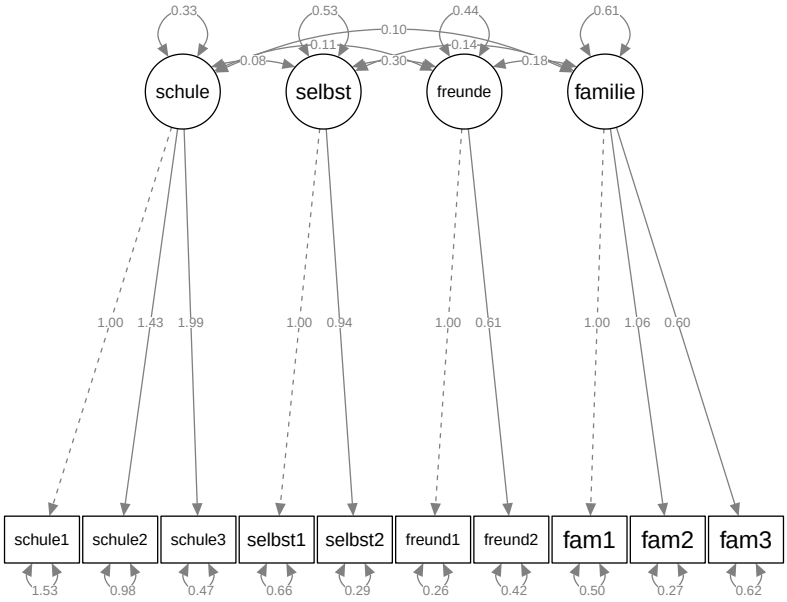
Die Kommunalitäten erhalten wir entweder über eine Quadrierung der zu einer manifesten Variablen gehörenden standardisierten Ladung oder über die Differenz 1 minus die standardisierte Residualvarianz (Uniqueness) einer manifesten Variablen.

Die höchste Kommunalität ist daher die von `schule3` mit $0.858^2 = 1 - 0.265 \approx 0.735$ und die niedrigste die von `schule1` mit $0.420^2 = 1 - 0.824 \approx 0.176$.

6.3.1.3. Modell visualisieren

Wir visualisieren Strukturgleichungsmodelle mit dem Package `semPlot`.

```
semPaths(fit_mod4f, "par",
  weighted = FALSE, nCharNodes = 7, shapeMan = "rectangle",
  sizeMan = 8, sizeMan2 = 5
)
```



6.3.1.4. Lokaler Fit: Vergleich der empirischen mit der vom Modell implizierten Varianz-Kovarianz-Matrix

Lokaler Fit bezieht sich auf die Frage, wie gut das Modell einzelne beobachtete Varianzen und Kovarianzen abbildet/repräsentiert. Wir vergleichen dafür die vom Modell implizierte Varianz-Kovarianz-Matrix mit der empirischen Varianz-Kovarianz-Matrix (der gemessenen Variablen).

Die Funktion `lavInspect()` aus dem Package `lavaan` ermöglicht die Extraktion von Informationen aus einem lavaan-Objekt. Mit dem Argument `what` spezifiziert man, welche Informationen extrahiert werden sollen. Der Wert `sampstat` dieses Arguments steht für "sample statistics", also für die empirische Varianz-Kovarianz-Matrix:

```
lavInspect(fit_mod4f, what = "sampstat")
```

```
$cov
        schul1 schul2 schul3 slbst1 slbst2 frend1 frend2  fam1  fam2  fam3
schule1  1.855
schule2  0.444  1.648
schule3  0.640  0.944  1.771
selbst1  0.104  0.156  0.235  1.189
selbst2  0.167  0.015  0.105  0.500  0.765
freund1  0.193  0.088  0.249  0.312  0.271  0.699
freund2  0.043  0.095  0.098  0.151  0.183  0.268  0.584
fam1     0.369  0.134  0.159  0.136  0.189  0.170  0.168 1.110
fam2     0.258  0.162  0.137  0.067  0.118  0.164  0.116 0.656 0.962
fam3     0.136  0.173  0.242  0.144  0.198  0.160  0.099 0.326 0.394 0.841
```

Die vom Modell implizierte Varianz-Kovarianz-Matrix erhält man mit `what = 'implied'`:

```
lavInspect(fit_mod4f, what = "implied")
```

```
$cov
        schul1 schul2 schul3 slbst1 slbst2 frend1 frend2  fam1  fam2  fam3
schule1  1.855
schule2  0.468  1.648
schule3  0.653  0.934  1.771
selbst1  0.075  0.108  0.150  1.189
selbst2  0.071  0.101  0.141  0.500  0.765
freund1  0.111  0.159  0.221  0.296  0.278  0.699
freund2  0.068  0.097  0.136  0.182  0.171  0.268  0.584
```

```
fam1     0.098  0.140  0.196  0.142  0.134  0.178  0.109 1.110
fam2     0.104  0.149  0.208  0.151  0.142  0.189  0.116 0.649 0.962
fam3     0.059  0.084  0.117  0.085  0.080  0.106  0.065 0.365 0.388 0.841
```

Je kleiner die Unterschiede zwischen diesen beiden Matrizen, desto besser passt das Modell auf die Daten, d.h. desto näher kommen die aus den geschätzten Parametern zurückgerechneten Varianzen und Kovarianzen an die empirischen Varianzen und Kovarianzen heran.

```
# Differenz der Matrizen berechnen und Ergebnis (Residualmatrix)
# in einem Objekt abspeichern
residualmatrix <-
  lavInspect(fit_mod4f, what = "sampstat")$cov -
  lavInspect(fit_mod4f, what = "implied")$cov
residualmatrix
```

```
        schul1 schul2 schul3 slbst1 slbst2 frend1 frend2   fam1   fam2   fam3
schule1  0.000
schule2 -0.024  0.000
schule3 -0.013  0.009  0.000
selbst1  0.028  0.049  0.085  0.000
selbst2  0.096 -0.086 -0.036  0.000  0.000
freund1  0.082 -0.070  0.028  0.016 -0.007  0.000
freund2 -0.025 -0.002 -0.038 -0.031  0.012  0.000  0.000
fam1     0.271 -0.007 -0.037 -0.006  0.055 -0.008  0.059  0.000
fam2     0.154  0.013 -0.071 -0.084 -0.023 -0.024  0.001  0.007  0.000
fam3     0.077  0.089  0.125  0.059  0.118  0.054  0.034 -0.040  0.006  0.000
```

Eine direkte Extraktion der Residualmatrix ist mit dem Argument `what = "resid"` möglich:

```
residualmatrix2 <- lavInspect(fit_mod4f, what = "resid")$cov
```

Um das ganze in diesem Dokument nicht doppelt darzustellen, zeigen wir lediglich, dass der Inhalt der beiden Objekte `residualmatrix` und `residualmatrix2` exakt gleich ist:

```
all.equal(residualmatrix, residualmatrix2)
```

```
[1] TRUE
```

Für die lokale Fit-Diagnose besonders relevant ist die Varianz-Kovarianz-Matrix der standardisierten Residuen. Diese extrahieren wir nicht mit `lavInspect()`, sondern direkt mit `resid()`.

```r
resid(fit_mod4f, type = "standardized")$cov
```

```
        schul1 schul2 schul3 slbst1 slbst2 frend1 frend2   fam1   fam2   fam3
schule1  0.000
schule2 -0.640  0.000
schule3 -1.035  2.314  0.000
selbst1  0.336  0.698  1.529  0.000
selbst2  1.437 -1.738 -1.196  0.000  0.000
freund1  1.331 -1.659  1.254  1.235 -0.901  0.000
freund2 -0.421 -0.044 -0.843 -1.100  0.680  0.000  0.000
fam1     3.298 -0.106 -0.764 -0.126  1.645 -0.266  1.560  0.000
fam2     2.081  0.243 -2.382 -2.080 -1.010 -1.338  0.017  2.679  0.000
fam3     1.064  1.357  1.960  1.080  2.829  1.386  0.892 -2.809  0.790  0.000
```

6.3.1.4.1. * Interpretation der Residuen

Betrachten Sie die Sample-(Ko-)Varianzmatrix und die Implizierte (Ko-)Varianzmatrix. Welches sind die drei stärksten Abweichungen zwischen beiden Matrizen? Werden die ensprechenden Varianzen/Kovarianzen vom Modell unter- oder überschätzt?

Dazu betrachten wir die unstandardisierte Residualmatrix.

`fam1 ~~ schule1 = 0.271`

`fam2 ~~ schule1 = 0.154`

`fam3 ~~ schule3 = 0.125`

Bei den drei grössten Abweichungen sind die empirischen Kovarianzen grösser als die vom Modell implizierten. Sie werden also durch das Modell unterschätzt, daher sind die zugehörigen Residualkovarianzen alle positiv. Inhaltlich erklärbar sind diese Abweichungen dadurch, dass im Modell keine Querladungen zugelassen sind, also z.B. Item `schule1` (Schulnoten) nicht auf dem Familien-Faktor laden „darf".

Welche sind signifikant?

Dazu benötigen wir die standardisierte Residualmatrix. Dort schauen wir, welche Werte (absolut) grösser sind als unsere z-verteilte Prüfgrösse ($\geq 2,58 \mathrel{\widehat{=}} p \leq 0.01$).

Nämlich:

`fam1 ~~ schule1 = 3.298`

```
fam3 ~~ selbst2 = 2.829

fam3 ~~ fam1 = -2.809

fam2 ~~ fam1 = 2.679
```

Interessanterweise beziehen sich zwei der vier signifikanten Residualkovarianzen auf solche innerhalb des Familien-Faktors!

6.3.1.5. Globaler Fit

$CFI = 0.958$ und $NNFI/TLI = 0.935$ (siehe Output unter 3.2) nehmen Werte knapp unter empfohlenen Cut-Off-Kriterien für einen guten Fit von 0.97 bzw. 0.95 an.

Der $RMSEA = 0.055$ liegt zwar knapp über dem Cut-Off für einen guten Model Fit von 0.05, ist aber nicht signifikant ($p = 0.341$) grösser als dieser, und gleichzeitig signifikant kleiner als der Cut-off von 0.08, ab dem der Model Fit als schlecht gilt ($p = 0.045$). Das 90 %-CI des RMSEA von $[0.029; 0.079]$ beinhaltet dementsprechend den Wert 0.05, nicht aber den Wert 0.08.

Auch nach dem $SRMR = 0.053$ ist der Model Fit als gut (< 0.08) zu beurteilen.

6.3.1.5.1. * CFI und NNFI/TLI sowie die Informationskriterien AIC und BIC von Hand berechnet

Wie in der Vorlesung erwähnt, sind diese Formeln prüfungsrelevant.

$$\begin{aligned} CFI &= 1 - \frac{\chi_t^2 - df_t}{\chi_i^2 - df_i} \\ &= 1 - \frac{51.433 - 29}{583.039 - 45} \\ &= 1 - \frac{22.433}{538.039} \\ &= 0.958 \end{aligned}$$

$$\begin{aligned} NNFI/TLI &= \left(\frac{\chi_i^2}{df_i} - \frac{\chi_t^2}{df_t}\right) / \left(\frac{\chi_i^2}{df_i} - 1\right) \\ &= \left(\frac{583.039}{45} - \frac{51.433}{29}\right) / \left(\frac{583.039}{45} - 1\right) \\ &= \frac{12.956 - 1.774}{12.956 - 1} \\ &= 0.935 \end{aligned}$$

$$
\begin{aligned}
AIC &= \chi^2 + 2 \cdot t \\
&= 51.433 + 2 \cdot 26 \\
&= 103.433
\end{aligned}
$$

$$
\begin{aligned}
BIC &= \chi^2 + \ln(n) \cdot t \\
&= 51.433 + 26 \cdot \ln(255) \\
&= 51.433 + 26 \cdot 5.541 \\
&= 195.499
\end{aligned}
$$

AIC und BIC unterscheiden sich von den von `lavaan` berechneten Werten, da letztere auf Basis der -2 Log-Likelihood und nicht auf der Basis von Chi-Quadrat berechnet werden. Für Modellvergleiche (s.u.) spielt es aber keine Rolle, welche dieser beiden Model-Fit-Statistiken man als Ausgangspunkt für die Berechnung von AIC und BIC nimmt. Da sich Chi-Quadrat und -2 Log-Likelihood nur um eine zu vernachlässigende Konstante voneinander unterscheiden, weist z.B. der AIC zweier Modelle immer dieselbe Differenz unabhängig von der Berechnungsart auf.

6.3.2. CFA mit drei Faktoren

6.3.2.1. Modell definieren

Wir definieren hier ein Modell mit drei Faktoren, da wir bei der EFA gesehen haben, dass ein solches möglicherweise ein sparsameres Alternativmodell sein könnte.

```
model_3f <- "
# Der erste Faktor ist `schule`
schule =~ schule1 + schule2 + schule3

# Als zweiten Faktor kombinieren wir jetzt `selbst` und `freunde`
selbstfreunde =~ selbst1 + selbst2 + freund1 + freund2

# Zum Schluss die `familie`
familie =~ fam1 + fam2 + fam3
"
```

6.3.2.2. Modell schätzen

```
fit_mod3f <- cfa(model_3f, data = ls_clean)
summary(fit_mod3f, fit.measures = TRUE, standardized = TRUE)
```

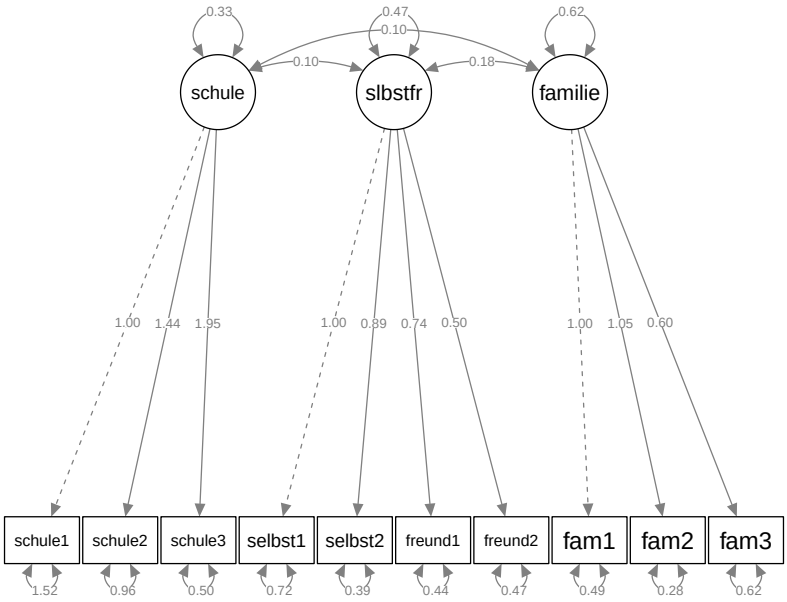
```
lavaan 0.6.17 ended normally after 39 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        23

  Number of observations                           255

Model Test User Model:

  Test statistic                                79.522
  Degrees of freedom                                32
  P-value (Chi-square)                           0.000

Model Test Baseline Model:

  Test statistic                               583.039
  Degrees of freedom                                45
  P-value                                        0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    0.912
  Tucker-Lewis Index (TLI)                       0.876

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -3441.804
  Loglikelihood unrestricted model (H1)      -3402.044

  Akaike (AIC)                                6929.609
  Bayesian (BIC)                              7011.058
  Sample-size adjusted Bayesian (SABIC)       6938.142

Root Mean Square Error of Approximation:

  RMSEA                                          0.076
```

```
  90 Percent confidence interval - lower         0.055
  90 Percent confidence interval - upper         0.098
  P-value H_0: RMSEA <= 0.050                    0.021
  P-value H_0: RMSEA >= 0.080                    0.408

Standardized Root Mean Square Residual:

  SRMR                                           0.064

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  schule =~
    schule1           1.000                               0.577    0.424
    schule2           1.436    0.258    5.574    0.000    0.828    0.645
    schule3           1.954    0.393    4.971    0.000    1.127    0.847
  selbstfreunde =~
    selbst1           1.000                               0.686    0.629
    selbst2           0.889    0.127    7.006    0.000    0.610    0.698
    freund1           0.741    0.110    6.722    0.000    0.509    0.608
    freund2           0.502    0.092    5.465    0.000    0.344    0.450
  familie =~
    fam1              1.000                               0.787    0.747
    fam2              1.048    0.128    8.191    0.000    0.824    0.840
    fam3              0.596    0.084    7.131    0.000    0.469    0.511

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  schule ~~
    selbstfreunde     0.103    0.039    2.616    0.009    0.261    0.261
    familie           0.102    0.041    2.468    0.014    0.225    0.225
  selbstfreunde ~~
    familie           0.178    0.051    3.507    0.000    0.331    0.331

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .schule1           1.522    0.147   10.379    0.000    1.522    0.821
   .schule2           0.962    0.139    6.926    0.000    0.962    0.584
```

```
.schule3          0.501    0.206    2.429    0.015    0.501    0.283
.selbst1          0.718    0.087    8.236    0.000    0.718    0.604
.selbst2          0.393    0.057    6.907    0.000    0.393    0.513
.freund1          0.441    0.051    8.572    0.000    0.441    0.630
.freund2          0.466    0.046   10.154    0.000    0.466    0.797
.fam1             0.491    0.079    6.189    0.000    0.491    0.443
.fam2             0.283    0.077    3.700    0.000    0.283    0.295
.fam3             0.621    0.061   10.222    0.000    0.621    0.739
 schule           0.333    0.110    3.019    0.003    1.000    1.000
 selbstfreunde    0.470    0.103    4.571    0.000    1.000    1.000
 familie          0.619    0.110    5.607    0.000    1.000    1.000
```

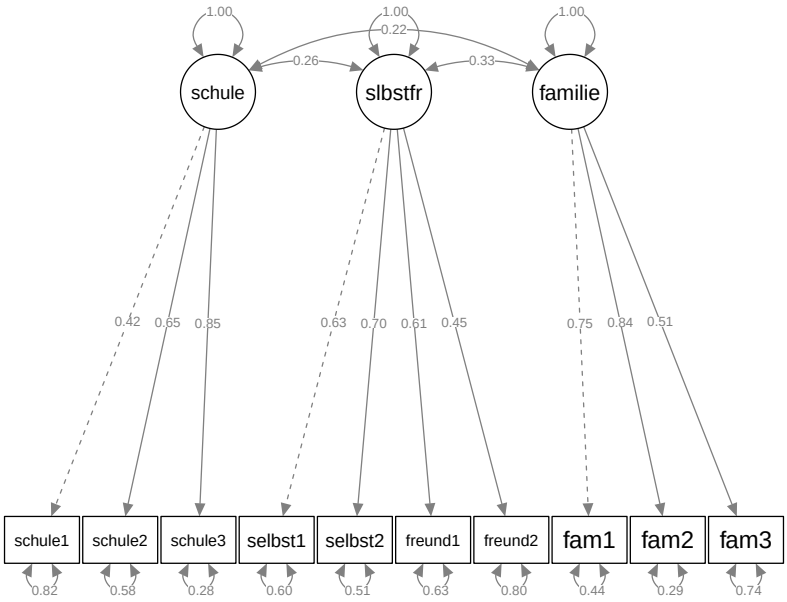
6.3.2.3. Modell visualisieren

```
semPaths(fit_mod3f, "par",
  weighted = FALSE, nCharNodes = 7, shapeMan = "rectangle",
  sizeMan = 8, sizeMan2 = 5
)
```



Und jetzt mit standardisierten Parameterschätzern:

```
semPaths(fit_mod3f, "std",
  weighted = FALSE, nCharNodes = 7, shapeMan = "rectangle",
  sizeMan = 8, sizeMan2 = 5
)
```



6.3.2.4. Globaler Fit

$CFI = 0.912$ und $NNFI/TLI = 0.876$ sind jetzt deutlich niedriger als die empfohlenen Cut-Off-Kriterien.

Der $RMSEA = 0.076$ ist jetzt auch signifikant grösser als der Cut-Off von 0.05 ($p = 0.021$) und nicht mehr signifikant kleiner als 0.08 ($p = 0.408$). Das 90 %-CI des RMSEA von $[0.055; 0.098]$ beinhaltet dementsprechend den Wert 0.05 nicht, den Wert 0.08 aber schon.

Lediglich nach dem $SRMR = 0.064$ ist der Model Fit noch als gut (< 0.08) zu beurteilen.

6.3.2.4.1. * CFI und NNFI/TLI sowie die Informationskriterien AIC und BIC von Hand berechnet

$$
\begin{aligned}
CFI &= 1 - \frac{\chi^2_t - df_t}{\chi^2_i - df_i} \\
&= 1 - \frac{79.522 - 32}{583.039 - 45} \\
&= 1 - \frac{47.522}{538.039} \\
&= 0.912
\end{aligned}
$$

$$
\begin{aligned}
NNFI/TLI &= \left(\frac{\chi^2_i}{df_i} - \frac{\chi^2_t}{df_t}\right) / \left(\frac{\chi^2_i}{df_i} - 1\right) \\
&= \left(\frac{583.039}{45} - \frac{79.522}{32}\right) / \left(\frac{583.039}{45} - 1\right) \\
&= \frac{12.956 - 2.485}{12.956 - 1} \\
&= 0.876
\end{aligned}
$$

$$
\begin{aligned}
AIC &= \chi^2 + 2 \cdot t \\
&= 79.522 + 2 \cdot 23 \\
&= 125.522
\end{aligned}
$$

$$
\begin{aligned}
BIC &= \chi^2 + t \cdot \ln(n) \\
&= 79.522 + 23 \cdot \ln(255) = 79.522 + 23 \cdot 5.541 \\
&= 206.965
\end{aligned}
$$

i Vertiefung: Alternative Spezifikation des 3-Faktor-Modells

Ein 3-Faktor-Modell ist alternativ spezifizierbar, indem bestimmte Parameter des 4-Faktor-Modells restringiert werden. Damit können wir zeigen, dass das 3-Faktor-Modell im 4-Faktor-Modell genestet ist.
Um aus dem 4-Faktor-Modell ein 3-Faktor-Modell mit einem kombinierten Selbst- und Freunde-Faktor zu machen, muss durch Parameterrestriktionen dafür gesorgt werden, dass die Faktoren `freunde` und `selbst` sich wie ein einziger Faktor verhalten. Damit diese zwei Faktoren zu einem werden, muss zum einen die Kovarianz zwischen den beiden Faktoren auf den Wert 1 gesetzt werden (Syntax siehe unten). Damit eine Kovarianz von 1 wirklich eine exakte Gleichheit der beiden Faktoren bedeutet, müssen auch die Varianzen der beiden latenten Variablen gleich 1 sein (damit wird dann auch die *Korrelation* zwischen den latenten Variablen 1). Die einfachste Möglichkeit, um das zu erreichen, ist die Festlegung der Varianzen aller latenten Variablen auf den Wert 1 mit dem `cfa()`-Argument `std.lv = TRUE`, das gleichzeitg dafür sorgt, dass auch die Ladungen der ersten

manifesten Variablen jedes Faktors frei geschätzt werden. Dieser Aspekt wird also nicht in der *Modelldefinition*, sondern erst bei der *Modellschätzung* festgelegt!
Zum anderen müssen die Zusammenhänge der Faktoren mit allen anderen Faktoren für beide Faktoren (`freunde` und `selbst`) identisch sein. Dies können wir erreichen, indem wir die Parameter der Faktoren-Kovarianzen *benennen* (d.h. explizit spezifizieren) und gleichzeitig den beiden gleichzusetzenden Parametern denselben Namen geben. Z.B. soll die Kovarianz von `freunde` und `schule` genau gleich geschätzt werden wie die Kovarianz zwischen `selbst` und `schule`. Wenn wir beide zu schätzende Parameter mit `a` benennen, erkennt lavaan, dass für beide Kovarianzen derselbe Wert geschätzt werden soll.
Dieses Modell nennen wir `model_4f_res`.

Modell definieren

```
model_4f_res <- "
schule  =~ schule1 + schule2 + schule3
selbst  =~ selbst1 + selbst2
freunde =~ freund1 + freund2
familie =~ fam1 + fam2 + fam3

# Restriktion Kovarianz
freunde ~~ 1 * selbst

# Gleichheitsrestriktionen: Kovarianzen mit gleichbenannten Parametern
freunde ~~ a * schule
selbst  ~~ a * schule
freunde ~~ b * familie
selbst  ~~ b * familie
"
```

Modell schätzen

```
fit_mod4f_res <- cfa(model_4f_res, std.lv = TRUE, data = ls_clean)
summary(fit_mod4f_res, fit.measures = TRUE, standardized = TRUE)
```

```
lavaan 0.6.17 ended normally after 26 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        25
  Number of equality constraints                     2
```

```
  Number of observations                           255

Model Test User Model:

  Test statistic                                79.522
  Degrees of freedom                               32
  P-value (Chi-square)                          0.000

Model Test Baseline Model:

  Test statistic                               583.039
  Degrees of freedom                                45
  P-value                                       0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    0.912
  Tucker-Lewis Index (TLI)                       0.876

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -3441.804
  Loglikelihood unrestricted model (H1)      -3402.044

  Akaike (AIC)                                6929.609
  Bayesian (BIC)                              7011.058
  Sample-size adjusted Bayesian (SABIC)       6938.142

Root Mean Square Error of Approximation:

  RMSEA                                          0.076
  90 Percent confidence interval - lower         0.055
  90 Percent confidence interval - upper         0.098
  P-value H_0: RMSEA <= 0.050                    0.021
  P-value H_0: RMSEA >= 0.080                    0.408

Standardized Root Mean Square Residual:

  SRMR                                           0.064

Parameter Estimates:
```

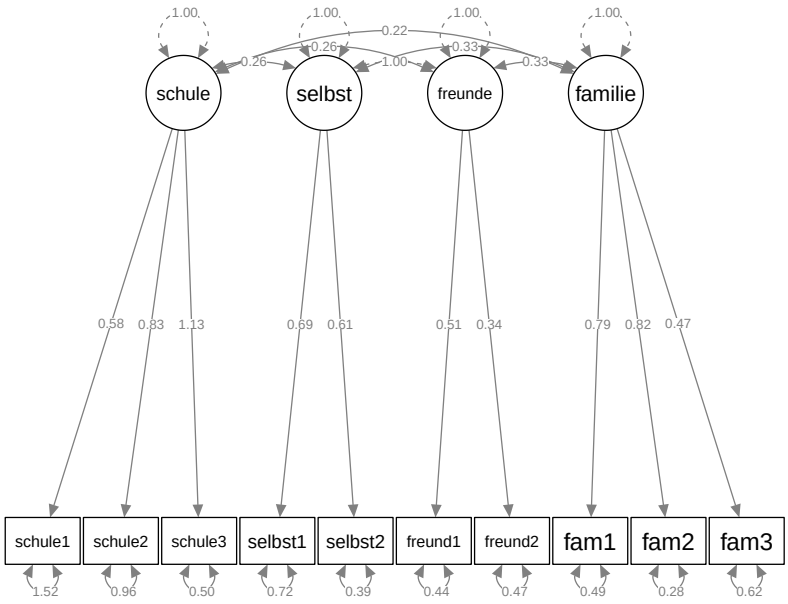
```
  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  schule =~
    schule1           0.577    0.096    6.037    0.000    0.577    0.424
    schule2           0.828    0.097    8.499    0.000    0.828    0.645
    schule3           1.127    0.111   10.109    0.000    1.127    0.847
  selbst =~
    selbst1           0.686    0.075    9.141    0.000    0.686    0.629
    selbst2           0.610    0.060   10.126    0.000    0.610    0.698
  freunde =~
    freund1           0.509    0.058    8.829    0.000    0.509    0.608
    freund2           0.344    0.054    6.378    0.000    0.344    0.450
  familie =~
    fam1              0.787    0.070   11.214    0.000    0.787    0.747
    fam2              0.824    0.066   12.461    0.000    0.824    0.840
    fam3              0.469    0.060    7.783    0.000    0.469    0.511

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  selbst ~~
    freunde           1.000                              1.000    1.000
  schule ~~
    freunde    (a)    0.261    0.081    3.216    0.001    0.261    0.261
    selbst     (a)    0.261    0.081    3.216    0.001    0.261    0.261
  freunde ~~
    familie    (b)    0.331    0.077    4.304    0.000    0.331    0.331
  selbst ~~
    familie    (b)    0.331    0.077    4.304    0.000    0.331    0.331
  schule ~~
    familie           0.225    0.077    2.922    0.003    0.225    0.225

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .schule1           1.522    0.147   10.379    0.000    1.522    0.821
   .schule2           0.962    0.139    6.926    0.000    0.962    0.584
   .schule3           0.501    0.206    2.429    0.015    0.501    0.283
   .selbst1           0.718    0.087    8.236    0.000    0.718    0.604
```

```
.selbst2          0.393    0.057    6.907    0.000    0.393    0.513
.freund1          0.441    0.051    8.572    0.000    0.441    0.630
.freund2          0.466    0.046   10.154    0.000    0.466    0.797
.fam1             0.491    0.079    6.189    0.000    0.491    0.443
.fam2             0.283    0.077    3.700    0.000    0.283    0.295
.fam3             0.621    0.061   10.222    0.000    0.621    0.739
 schule           1.000                               1.000    1.000
 selbst           1.000                               1.000    1.000
 freunde          1.000                               1.000    1.000
 familie          1.000                               1.000    1.000
```

Modell visualisieren

Mit standardisierten Parameterschätzern:

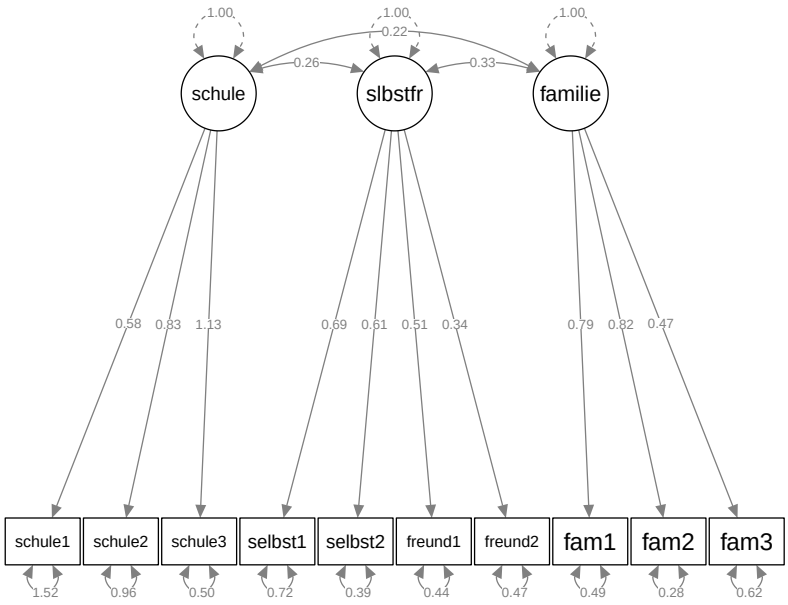
```
semPaths(fit_mod4f_res, "std", "par",
  weighted = FALSE, nCharNodes = 7,
  shapeMan = "rectangle", sizeMan = 8, sizeMan2 = 5
)
```



Um bei allen Parametern exakt die gleichen standardisierten Parameterschätzer zu erhalten wie im "normal" definierten 3-Faktor-Modell oben, muss letzteres auch mit der Option `std.lv = TRUE` geschätzt werden. Sonst ergeben sich hier aufgrund der unter-

schiedlichen Festlegung der Skalierung der latenten Variablen kleine Unterschiede bei den Ladungen und bei den Residualvarianzen.

```
fit_mod3f_alt <- cfa(model_3f, std.lv = TRUE, data = ls_clean)
semPaths(fit_mod3f_alt, "std", "par",
  weighted = FALSE, nCharNodes = 7,
  shapeMan = "rectangle", sizeMan = 8, sizeMan2 = 5
)
```



Modellvergleich von 3-Faktor-Modell und alternativem 3-Faktor-Modell

Ein Modellvergleich sollte jetzt zeigen, dass das 3-Faktor-Modell und das restringierte 4-Faktor-Modell gleich gut fitten, weil sie identisch sind.
Wir testen das:

```
anova(fit_mod3f, fit_mod4f_res)
```

```
Chi-Squared Difference Test

             Df    AIC    BIC  Chisq Chisq diff RMSEA Df diff Pr(>Chisq)
fit_mod3f    32 6929.6 7011.1 79.522
fit_mod4f_res 32 6929.6 7011.1 79.522 2.2919e-10     0       0
```

Das ist der Fall! (Die sehr kleine Zahl bei `Chisq diff` kommt durch minimale numerische Ungenauigkeiten bei der ML-Schätzung zu Stande.)

6.3.3. Modellvergleich 3-Faktor-Modell und 4-Faktor-Modell

Ausserdem wollen wir noch das 3-Faktor-Modell gegen das ursprüngliche 4-Faktormodell testen.

Dieser Test überprüft, ob die Nullhypothese, dass das 3-Faktor-Modell *nicht* schlechter als das 4-Faktor-Modell auf die Daten passt, aufrechterhalten werden kann oder abgelehnt werden muss.

```
anova(fit_mod3f, fit_mod4f)
```

```
Chi-Squared Difference Test

          Df    AIC    BIC  Chisq Chisq diff  RMSEA Df diff Pr(>Chisq)
fit_mod4f 29 6907.5 6999.6 51.433
fit_mod3f 32 6929.6 7011.1 79.522     28.089 0.1811       3  3.479e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Der Vergleich ist signifikant. Das 3-Faktormodell passt also signifikant schlechter auf die Daten als das 4-Faktor-Modell.

Diese Schlussfolgerung ergibt sich auch aufgrund der oben bereits von Hand berechneten Informationskriterien AIC und BIC:

Beim AIC ergab sich für das 4-Faktor-Modell ein Wert von 103.433 und für das 3-Faktor-Modell ein Wert von 125.522 (respektive die von `lavaan` ausgegebenen AIC-Werte von 6907.52 und 6929.609). Nach dem AIC ist also das 4-Faktor-Modell zu bevorzugen (kleinerer AIC).

Beim BIC ergibt sich ein ähnliches Bild: von Hand gerechnet beim 4-Faktor-Modell BIC = 195.499 und beim 3-Faktor-Modell BIC = 206.965 (von `lavaan` ausgegebene Werte: 6999.593 und 7011.058). Auch nach dem (im Vergleich zum AIC) eher sparsamere Modelle (mit weniger Parametern) bevorzugenden BIC sollte also das 4-Faktor-Modell ausgewählt werden.

6.3.4. Modell mit Lebenszufriedenheitsfaktor zweiter Ordnung

Wie wir sehen können, sind die vier Faktoren im 4-Faktor-Modell alle positiv untereinander korreliert und diese Korrelationen sind mit einer Ausnahme ($r_{SchuleSelbst} = 0.18,\ p = 0.052$) auch signifikant. Dies und die Tatsache, dass die Lebenszufriedenheit in der psychologischen Forschung nicht nur bereichsspezifisch, sondern auch (und sogar vorwiegend) global konzeptualisiert wird, lassen vermuten, dass ein übergeordneter Faktor (higher order factor, 2nd order factor) *Lebenszufriedenheit* existiert, der die Korrelationen der bereichsspezifischen Lebenszufriedenheitsfaktoren erklären kann.

In der CFA können wir einen solchen Faktor zweiter Ordnung modellieren. Ein weiterer Faktor scheint das Modell zunächst komplizierter zu machen, doch das kann täuschen: Tatsächlich werden in einem Modell mit *einem* übergeordneten Faktor insgesamt *zwei* Parameter weniger geschätzt als im Modell mit vier Faktoren: Anstelle von sechs Kovarianzen zwischen den Faktoren werden jetzt vier Ladungen (bzw. drei Ladungen und eine Varianz des Faktors zweiter Ordnung) geschätzt.

6.3.4.1. Modell definieren

```
model_4f_2order <- "
schule  =~ schule1 + schule2 + schule3
selbst  =~ selbst1 + selbst2
freunde =~ freund1 + freund2
familie =~ fam1 + fam2 + fam3

# Übergeordneter Faktor Zufriedenheit
zufriedenheit =~ schule + selbst + freunde + familie
"
```

6.3.4.2. Modell schätzen

```
fit_mod4f_2order <- cfa(model_4f_2order, ls_clean)
summary(fit_mod4f_2order, fit.measures = TRUE, standardized = TRUE)
```

```
lavaan 0.6.17 ended normally after 67 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        24
```

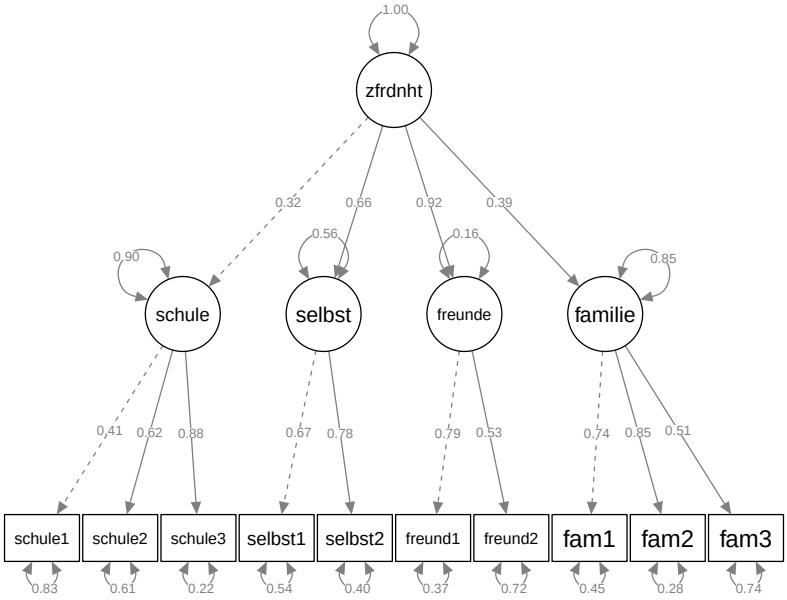
```
  Number of observations                           255

Model Test User Model:

  Test statistic                                53.372
  Degrees of freedom                               31
  P-value (Chi-square)                          0.008

Model Test Baseline Model:

  Test statistic                               583.039
  Degrees of freedom                                45
  P-value                                       0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    0.958
  Tucker-Lewis Index (TLI)                       0.940

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -3428.730
  Loglikelihood unrestricted model (H1)      -3402.044

  Akaike (AIC)                                6905.459
  Bayesian (BIC)                              6990.450
  Sample-size adjusted Bayesian (SABIC)       6914.364

Root Mean Square Error of Approximation:

  RMSEA                                          0.053
  90 Percent confidence interval - lower         0.027
  90 Percent confidence interval - upper         0.077
  P-value H_0: RMSEA <= 0.050                    0.387
  P-value H_0: RMSEA >= 0.080                    0.030

Standardized Root Mean Square Residual:

  SRMR                                           0.060

Parameter Estimates:

  Standard errors                             Standard
```

```
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  schule =~
    schule1           1.000                               0.555    0.407
    schule2           1.439    0.261    5.508    0.000    0.799    0.622
    schule3           2.118    0.449    4.713    0.000    1.175    0.883
  selbst =~
    selbst1           1.000                               0.736    0.675
    selbst2           0.924    0.153    6.040    0.000    0.680    0.777
  freunde =~
    freund1           1.000                               0.662    0.792
    freund2           0.612    0.117    5.229    0.000    0.405    0.530
  familie =~
    fam1              1.000                               0.780    0.741
    fam2              1.068    0.131    8.136    0.000    0.833    0.849
    fam3              0.595    0.084    7.106    0.000    0.465    0.507
  zufriedenheit =~
    schule            1.000                               0.322    0.322
    selbst            2.720    0.985    2.763    0.006    0.661    0.661
    freunde           3.397    1.250    2.717    0.007    0.917    0.917
    familie           1.686    0.661    2.550    0.011    0.386    0.386

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .schule1           1.547    0.148   10.481    0.000    1.547    0.834
   .schule2           1.011    0.139    7.278    0.000    1.011    0.613
   .schule3           0.390    0.231    1.689    0.091    0.390    0.220
   .selbst1           0.647    0.100    6.442    0.000    0.647    0.544
   .selbst2           0.303    0.075    4.020    0.000    0.303    0.396
   .freund1           0.261    0.080    3.277    0.001    0.261    0.373
   .freund2           0.420    0.047    8.958    0.000    0.420    0.719
   .fam1              0.501    0.079    6.317    0.000    0.501    0.452
   .fam2              0.268    0.078    3.429    0.001    0.268    0.279
   .fam3              0.625    0.061   10.256    0.000    0.625    0.743
   .schule            0.276    0.095    2.915    0.004    0.896    0.896
   .selbst            0.305    0.091    3.369    0.001    0.563    0.563
   .freunde           0.070    0.106    0.661    0.509    0.159    0.159
   .familie           0.518    0.097    5.360    0.000    0.851    0.851
    zufriedenheit     0.032    0.021    1.536    0.125    1.000    1.000
```

6.3.4.3. Modell visualisieren

Hier jetzt gleich mit standardisierten Parametern:

```
semPaths(fit_mod4f_2order, "std",
  weighted = FALSE, nCharNodes = 7,
  shapeMan = "rectangle", sizeMan = 8, sizeMan2 = 5
)
```



Die Parameterschätzer zeigen, dass die stärkste standardisierte Ladung der Bereichsfaktoren (Faktoren erster Ordnung) diejenige des Bereichs *Freunde* ist, während die niedrigste diejenige des Bereichs *Schule* ist. Das Ladungsmuster zeigt sich insgesamt relativ heterogen.

6.3.4.4. Model Fit und Modellvergleiche

CFI = 0.958 und NNFI/TLI = 0.94 sind sehr ähnlich wie beim 4-Faktor-Modell. Der RMSEA = 0.053 ist noch etwas niedriger als bei letzterem und nicht signifikant grösser als 0.05 (90 %-CI = [0.027; 0.077]). Der SRMR = 0.06 fällt dagegen etwas höher als beim 4-Faktor-Modell aus.

AIC = 101.372 und BIC = 186.356 (nach der Eid-Formel mit Chi-Quadrat berechnet) sind beide niedriger als die entsprechenden Werte des 4-Faktor-Modells (AIC = 103.433 und BIC = 195.499). Nach diesen Informationskriterien ist also das Modell mit Faktor zweiter Ordnung gegenüber dem 4-Faktor-Modell zu bevorzugen.

Jetzt vergleichen wir noch alle drei Modelle (in aufsteigender Reihenfolge der Anzahl geschätzter Parameter: 3-Faktor-Modell, 4-Faktor-Modell mit Faktor zweiter Ordnung, 4-Faktor-Modell) über sequentielle Likelihood-Ratio-Tests. Die Voraussetzung für die Gültigkeit dieser Tests ist wie wir wissen die Nestung der eingeschränkten Modelle (mit weniger Parameter) in den Modellen mit weniger Restriktionen (mit mehr Parametern). Für das 3-Faktor- und das 4-Faktor-Modell haben wir die Nestung oben nachgewiesen. Aber auch das 4-Faktor-Modell mit Faktor zweiter Ordnung ist im 4-Faktor-Modell genestet: alle Modelle, die die gleichen vier Faktoren erster Ordnung haben, sind im 4-Faktor-Modell genestet, da es sich bei diesem um ein gesättigtes Modell auf der Ebene der latenten Variablen handelt.

```
anova(fit_mod3f, fit_mod4f_2order, fit_mod4f)
```

```
Chi-Squared Difference Test

                Df    AIC    BIC  Chisq Chisq diff   RMSEA Df diff Pr(>Chisq)
fit_mod4f       29 6907.5 6999.6 51.433
fit_mod4f_2order 31 6905.5 6990.4 53.372     1.9395 0.00000       2     0.3792
fit_mod3f       32 6929.6 7011.1 79.522    26.1494 0.31405       1   3.16e-07

fit_mod4f
fit_mod4f_2order
fit_mod3f       ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Der Modellvergleich zeigt, dass das 4-Faktor-Modell mit Faktor zweiter Ordnung *nicht* signifikant schlechter auf die Daten passt als das weniger sparsame 4-Faktor-Modell ($\Delta\chi^2 = 1.9395$, $p = 0.3792$). Der zweite Test zeigt dagegen einen signifikanten Modellvergleich des 3-Faktor-Modells mit dem 4-Faktor-Modell mit Faktor zweiter Ordnung an ($\Delta\chi^2 = 26.1494$, $p = 0$). Das 3-Faktor-Modell passt also auch hier signifikant schlechter.

Das beste dieser drei Modelle ist also das 4-Faktor-Modell mit Faktor zweiter Ordnung. Dieses weist auch die jeweils niedrigsten von `lavaan` ausgegebenen Informationskriterien aller Modelle auf (AIC = 6905.5, BIC = 6990.4).

6.4. Übung

Wir nehmen wieder Daten aus dem *Big Five Inventory* (bfi) mit 2800 Datenpunkten ($n = 2800$)

Laden Sie die Daten mit folgendem Code-Chunk:

```
pacman::p_load(tidyverse, lavaan, psychTools, semPlot)
data("bfi", package = "psychTools")
```

Aufgabe 1

a) Definieren Sie ein Modell mit 5 sinnvollen Faktoren und lassen Sie die Faktoren frei miteinander kovariieren/korrelieren. Verwenden Sie dafür nur die Variablen, die einer Frage (Item) im *bfi*-Fragebogen entsprechen (also ohne die demographischen Angaben). Geben Sie den Faktoren brauchbare Namen, das hilft bei der Interpretation. Dafür schauen Sie sich am Besten die Dokumentation des Datenframes an. Tipp: `?psychTools::bfi`

Lösung

Modell definieren

```
bfi_5F <- "
Agreeableness     =~ A1 + A2 + A3 + A4 + A5
Conscientiousness =~ C1 + C2 + C3 + C4 + C5
Extraversion      =~ E1 + E2 + E3 + E4 + E5
Neuroticism       =~ N1 + N2 + N3 + N4 + N5
Openness          =~ O1 + O2 + O3 + O4 + O5
"
```

Aus der Itemstruktur sowie der Dokumentation der Daten wird klar, dass jede der Komponente der *Big Five* durch einen eigenen Faktor dargestellt werden sollte. Welche Items zu welchem Faktor gehören, ist dank der Namensgebung leicht zu erkennen. Jeder Faktor hat hier fünf manifeste Variablen. Hier die Items mit Kennzeichnung der inhaltlich umgekehrt formulierten Items mit (RE).
Agreeableness:

- A1: Am indifferent to the feelings of others. (RE)
- A2: Inquire about others' well-being.
- A3: Know how to comfort others.
- A4: Love children.
- A5: Make people feel at ease.

Conscientiousness:

- C1: Am exacting in my work.
- C2: Continue until everything is perfect.
- C3: Do things according to a plan.
- C4: Do things in a half-way manner. (RE)
- C5: Waste my time. (RE)

Extraversion:

- E1: Don't talk a lot. (RE)
- E2: Find it difficult to approach others. (RE)
- E3: Know how to captivate people.
- E4: Make friends easily.
- E5: Take charge.

Neuroticism:

- N1: Get angry easily.
- N2: Get irritated easily.
- N3: Have frequent mood swings.
- N4: Often feel blue.
- N5: Panic easily.

Openness:

- O1: Am full of ideas.
- O2: Avoid difficult reading material. (RE)
- O3: Carry the conversation to a higher level.
- O4: Spend time reflecting on things.
- O5: Will not probe deeply into a subject. (RE)

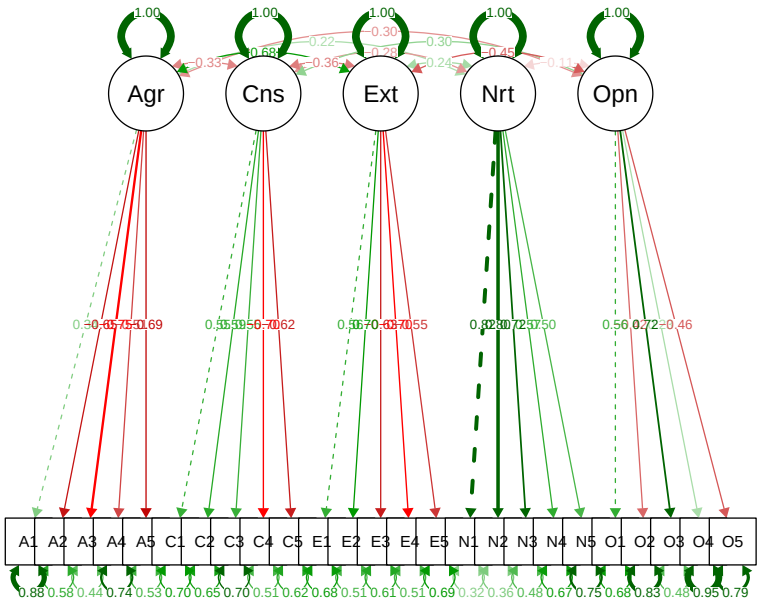
b) Fitten Sie dieses Modell und stellen Sie das Modell mit der Funktion `semPaths()` dar. Es sollen die standardisierten Parameterschätzer dargestellt werden. Diese Funktion finden

Sie im Package `semPlot`. Lassen Sie sich das Modell inkl. Fit-Indizes und standardisierter Parameterschätzer ausgeben. Wie ist der Model-Fit zu beurteilen? Wie sind die (standardisierten) Faktorladungen zu beurteilen? Welche Korrelationen weisen die Persönlichkeitsfaktoren auf?

Lösung

Modell fitten, darstellen und Ergebnisse ausgeben:

```
fit_bfi_5F <- cfa(data = bfi, model = bfi_5F) # Fitten

semPaths(fit_bfi_5F, "std") # Darstellen
```




```
summary(fit_bfi_5F, fit.measures = TRUE, standardized = TRUE) # Ausgeben
```

```
lavaan 0.6.17 ended normally after 55 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        60

                                                  Used       Total
  Number of observations                          2436        2800
```

```
Model Test User Model:

  Test statistic                                4165.467
  Degrees of freedom                                 265
  P-value (Chi-square)                             0.000

Model Test Baseline Model:

  Test statistic                               18222.116
  Degrees of freedom                                 300
  P-value                                          0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                      0.782
  Tucker-Lewis Index (TLI)                         0.754

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -99840.238
  Loglikelihood unrestricted model (H1)      -97757.504

  Akaike (AIC)                               199800.476
  Bayesian (BIC)                             200148.363
  Sample-size adjusted Bayesian (SABIC)      199957.729

Root Mean Square Error of Approximation:

  RMSEA                                            0.078
  90 Percent confidence interval - lower           0.076
  90 Percent confidence interval - upper           0.080
  P-value H_0: RMSEA <= 0.050                      0.000
  P-value H_0: RMSEA >= 0.080                      0.037

Standardized Root Mean Square Residual:

  SRMR                                             0.075

Parameter Estimates:

  Standard errors                             Standard
```

```
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
                     Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  Agreeableness =~
    A1                  1.000                               0.484    0.344
    A2                 -1.579    0.108  -14.650    0.000   -0.764   -0.648
    A3                 -2.030    0.134  -15.093    0.000   -0.983   -0.749
    A4                 -1.564    0.115  -13.616    0.000   -0.757   -0.510
    A5                 -1.804    0.121  -14.852    0.000   -0.873   -0.687
  Conscientiousness =~
    C1                  1.000                               0.680    0.551
    C2                  1.148    0.057   20.152    0.000    0.781    0.592
    C3                  1.036    0.054   19.172    0.000    0.705    0.546
    C4                 -1.421    0.065  -21.924    0.000   -0.967   -0.702
    C5                 -1.489    0.072  -20.694    0.000   -1.012   -0.620
  Extraversion =~
    E1                  1.000                               0.920    0.564
    E2                  1.226    0.051   23.899    0.000    1.128    0.699
    E3                 -0.921    0.041  -22.431    0.000   -0.847   -0.627
    E4                 -1.121    0.047  -23.977    0.000   -1.031   -0.703
    E5                 -0.808    0.039  -20.648    0.000   -0.743   -0.553
  Neuroticism =~
    N1                  1.000                               1.300    0.825
    N2                  0.947    0.024   39.899    0.000    1.230    0.803
    N3                  0.884    0.025   35.919    0.000    1.149    0.721
    N4                  0.692    0.025   27.753    0.000    0.899    0.573
    N5                  0.628    0.026   24.027    0.000    0.816    0.503
  Openness =~
    O1                  1.000                               0.635    0.564
    O2                 -1.020    0.068  -14.962    0.000   -0.648   -0.418
    O3                  1.373    0.072   18.942    0.000    0.872    0.724
    O4                  0.437    0.048    9.160    0.000    0.277    0.233
    O5                 -0.960    0.060  -16.056    0.000   -0.610   -0.461

Covariances:
                     Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  Agreeableness ~~
    Conscientisnss     -0.110    0.012   -9.254    0.000   -0.334   -0.334
    Extraversion        0.304    0.025   12.293    0.000    0.683    0.683
```

```
    Neuroticism           0.141    0.018    7.712    0.000    0.223    0.223
    Openness             -0.093    0.011   -8.446    0.000   -0.303   -0.303
  Conscientiousness ~~
    Extraversion         -0.224    0.020  -11.121    0.000   -0.357   -0.357
    Neuroticism          -0.250    0.025  -10.117    0.000   -0.283   -0.283
    Openness              0.130    0.014    9.190    0.000    0.301    0.301
  Extraversion ~~
    Neuroticism           0.292    0.032    9.131    0.000    0.244    0.244
    Openness             -0.265    0.021  -12.347    0.000   -0.453   -0.453
  Neuroticism ~~
    Openness             -0.093    0.022   -4.138    0.000   -0.112   -0.112

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .A1                1.745    0.052   33.725    0.000    1.745    0.882
   .A2                0.807    0.028   28.396    0.000    0.807    0.580
   .A3                0.754    0.032   23.339    0.000    0.754    0.438
   .A4                1.632    0.051   31.796    0.000    1.632    0.740
   .A5                0.852    0.032   26.800    0.000    0.852    0.528
   .C1                1.063    0.035   30.073    0.000    1.063    0.697
   .C2                1.130    0.039   28.890    0.000    1.130    0.650
   .C3                1.170    0.039   30.194    0.000    1.170    0.702
   .C4                0.960    0.040   24.016    0.000    0.960    0.507
   .C5                1.640    0.059   27.907    0.000    1.640    0.615
   .E1                1.814    0.058   31.047    0.000    1.814    0.682
   .E2                1.332    0.049   26.928    0.000    1.332    0.512
   .E3                1.108    0.038   29.522    0.000    1.108    0.607
   .E4                1.088    0.041   26.732    0.000    1.088    0.506
   .E5                1.251    0.040   31.258    0.000    1.251    0.694
   .N1                0.793    0.037   21.575    0.000    0.793    0.320
   .N2                0.836    0.036   23.458    0.000    0.836    0.356
   .N3                1.222    0.043   28.271    0.000    1.222    0.481
   .N4                1.654    0.052   31.977    0.000    1.654    0.672
   .N5                1.969    0.060   32.889    0.000    1.969    0.747
   .O1                0.865    0.032   27.216    0.000    0.865    0.682
   .O2                1.990    0.063   31.618    0.000    1.990    0.826
   .O3                0.691    0.039   17.717    0.000    0.691    0.476
   .O4                1.346    0.040   34.036    0.000    1.346    0.946
   .O5                1.380    0.045   30.662    0.000    1.380    0.788
    Agreeableness     0.234    0.030    7.839    0.000    1.000    1.000
    Conscientisnss    0.463    0.036   12.810    0.000    1.000    1.000
```

```
Extraversion      0.846    0.062   13.693    0.000     1.000     1.000
Neuroticism       1.689    0.073   23.034    0.000     1.000     1.000
Openness          0.404    0.033   12.156    0.000     1.000     1.000
```

Model-Fit:
Der Model-Fit ist nach allen Fit-Indizes (ausser dem SRMR) als ziemlich schlecht zu bewerten:
CFI = 0.782
TLI = 0.754
RMSEA = 0.078 (90% CI [0.076, 0.080])
SRMR = 0.075
Es wäre jetzt natürlich interessant zu wissen, woran das liegt. Dazu könnten wir zum einen die Matrix der standardisierten Residuen ansehen, um zu beurteilen, welche empirischen Kovarianzen durch das Modell besonders schlecht repräsentiert werden. Zum anderen könnten wir uns Modifikationsindizes ausgeben lassen (Welche Parameter könnten/sollten zusätzlich geschätzt werden, um den Fit zu verbessern?). Zu befürchten ist, dass alle Massnahmen, die einen besseren Model-Fit ermöglichen würden (d.h. zusätzliche geschätzte Parameter) nicht mit unserer Idee/Hypothese einer einfachstrukturierten Big-5-Faktorstruktur ohne Residualkovarianzen vereinbar wäre. Unter Umständen könnten aber sehr wenige und ggf. gut interpretierbare Querladungen und/oder Residualkovarianzen für einen akzeptablen Fit sorgen. Schauen Sie es sich an!
Wie sind die (standardisierten) Faktorladungen zu beurteilen?
Bis auf wenige Ausnahmen sind die Faktorladungen als substantiell zu beurteilen (Betrag ca. >= 0.5). Auffällig ist, dass zwei der fünf Faktoren über das Muster von positiven und negativen Faktorladungen in inhaltlich entgegengesetzter Richtung zu interpretieren sind: *Agreeableness* und *Extraversion*. Das spielt eine wichtige Rolle für die Interpretation ihrer Kovarianzen/Korrelationen mit den anderen Faktoren (s.u.). Der Grund dafür liegt in der Festlegung jeweils eines inhaltlich entgegengesetzt formulierten Items als Ankeritem mit der (positiven) Ladung `1`. Die inhaltlich "richtig" formulierten Items erhalten dadurch negative Ladungen.
Welche Korrelationen weisen die Persönlichkeitsfaktoren auf?
Alle Kovarianzen sind signifikant ($p < .001$). Die standardisierten Kovarianzen sind die Korrelationen, diese variieren vom Betrag her zwischen 0.112 und 0.683. Wichtig ist, dass die Korrelationen von *Agreeableness* und *Extraversion* mit den anderen drei Faktoren in die andere Richtung interpretiert bzw. die Faktoren als *Unfriendlyness* (Unfreundlichkeit) und *Introversion* interpretiert werden müssen.

Aufgabe 2

Finden Sie folgende Werte mithilfe der Darstellung und der Ausgabe des Modells aus Aufgabe 1.

a) Wie viele Informationen enthält die Varianz-Kovarianz Matrix der manifesten Variablen (n_{info})?

Lösung

Informationen (wobei p = Anzahl manifester Variablen)

$$\begin{aligned} n_{info} &= \frac{p * (p+1)}{2} \\ &= \frac{25 * (25+1)}{2} \\ &= \frac{650}{2} \\ &= 325 \end{aligned}$$

b) Wie viele Freiheitsgrade hat das Modell?

Lösung

Die Freiheitsgrade lassen sich berechnen als:

$$df = n_{info} - n_{parameter}$$

Also die Anzahl Informationen minus die Anzahl der zu schätzender Parameter. Je nach Darstellung kann man die Anzahl der zu schätzender Parameter auch abzählen, in dem man jeden (nicht gestrichelten) Pfeil zählt.

- Varianzen der latenten Variablen: 5
- Kovarianzen der latenten Variablen. Auch dafür gibt es eine Formel (für den Fall dass alle Kovarianzen frei geschätzt werden), sie ist sogar ziemlich ähnlich wie die von n_{info}, wobei n_{lv} jetzt für die Anzahl latenter Variablen steht:

$$\begin{aligned} n_{cov\ latent\ variables} &= \frac{n_{lv} * (n_{lv}-1)}{2} \\ &= \frac{5 * (5-1)}{2} \\ &= \frac{20}{2} \\ &= 10 \end{aligned}$$

Beachte, dass das $p+1$ aus der Formel oben hier zu einem $p-1$ geworden ist! Wichtig: Diese Formel stimmt nur, wenn *alle* Kovarianzen der latenten Variablen frei geschätzt werden (was normalerweise der Fall ist).

- Varianzen der Residuen der manifesten Variablen: 25

- Zu schätzende Faktorladungen:

$$\begin{aligned} n_{zu\ schtzende\ Ladungen} &= n_{Ladungen} - n_{fixierte\ Ladungen} \\ &= 25 - 5 \\ &= 20 \end{aligned}$$

Wir haben also insgesamt zu schätzende Parameter:
$n_{parameter} = 5 + 10 + 25 + 20 = 60$
Und damit hat das Modell folgende Freiheitsgrade:
$df = n_{info} - n_{parameter} = 325 - 60 = 265$
Das entspricht auch dem Modelloutput:

```
fit_bfi_5F@test$standard$df # degrees of freedom direkt aus dem Modell
```

```
[1] 265
```

Aufgabe 3

a) Schätzen Sie nun dasselbe Modell, diesmal aber ohne Kovarianzen zwischen den latenten Variablen (also so, als ob wir *orthogonale* Faktoren erwarten würden).

Lösung

Modell definieren:

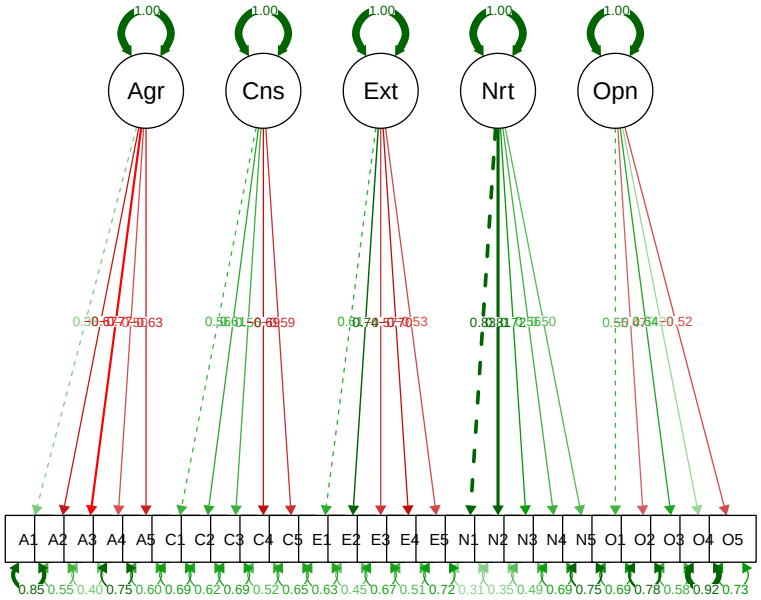
```
bfi_5F_noCV <- "
# Latente Faktoren bleiben:
Agreeableness     =~ A1 + A2 + A3 + A4 + A5
Conscientiousness =~ C1 + C2 + C3 + C4 + C5
Extraversion      =~ E1 + E2 + E3 + E4 + E5
Neuroticism       =~ N1 + N2 + N3 + N4 + N5
Openness          =~ O1 + O2 + O3 + O4 + O5

# Kovarianzen fixieren (wir haben 10 Kovarianzen)
Agreeableness ~~ 0 * Conscientiousness
Agreeableness ~~ 0 * Extraversion
Agreeableness ~~ 0 * Neuroticism
Agreeableness ~~ 0 * Openness
Conscientiousness ~~ 0 * Extraversion
Conscientiousness ~~ 0 * Neuroticism
Conscientiousness ~~ 0 * Openness
Extraversion ~~ 0 * Neuroticism
Extraversion ~~ 0 * Openness
Neuroticism ~~ 0 * Openness
"
```

Modell fitten:

```
fit_bfi_5F_noCV <- cfa(bfi_5F_noCV,
  data = bfi
)
```

Modell darstellen & ausgeben:

```
semPaths(fit_bfi_5F_noCV, "std") # Darstellen
```




```r
summary(fit_bfi_5F_noCV) # Ausgeben
```

```
lavaan 0.6.17 ended normally after 53 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        50

                                                  Used       Total
  Number of observations                          2436        2800

Model Test User Model:

  Test statistic                              5639.709
  Degrees of freedom                               275
  P-value (Chi-square)                           0.000

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
```

```
                     Estimate  Std.Err  z-value  P(>|z|)
  Agreeableness =~
    A1                  1.000
    A2                 -1.468    0.092  -15.885    0.000
    A3                 -1.887    0.116  -16.273    0.000
    A4                 -1.394    0.097  -14.400    0.000
    A5                 -1.502    0.096  -15.658    0.000
  Conscientiousness =~
    C1                  1.000
    C2                  1.172    0.057   20.409    0.000
    C3                  1.047    0.054   19.347    0.000
    C4                 -1.377    0.064  -21.526    0.000
    C5                 -1.403    0.070  -20.029    0.000
  Extraversion =~
    E1                  1.000
    E2                  1.205    0.048   25.012    0.000
    E3                 -0.785    0.036  -21.573    0.000
    E4                 -1.034    0.042  -24.402    0.000
    E5                 -0.713    0.035  -20.195    0.000
  Neuroticism =~
    N1                  1.000
    N2                  0.947    0.024   40.245    0.000
    N3                  0.873    0.024   35.898    0.000
    N4                  0.665    0.025   26.891    0.000
    N5                  0.616    0.026   23.814    0.000
  Openness =~
    O1                  1.000
    O2                 -1.165    0.077  -15.042    0.000
    O3                  1.248    0.074   16.878    0.000
    O4                  0.551    0.052   10.515    0.000
    O5                 -1.106    0.069  -15.970    0.000

Covariances:
                     Estimate  Std.Err  z-value  P(>|z|)
  Agreeableness ~~
    Conscientisnss      0.000
    Extraversion        0.000
    Neuroticism         0.000
    Openness            0.000
  Conscientiousness ~~
    Extraversion        0.000
```

```
    Neuroticism           0.000
    Openness              0.000
  Extraversion ~~
    Neuroticism           0.000
    Openness              0.000
  Neuroticism ~~
    Openness              0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .A1                1.691    0.051   33.160    0.000
   .A2                0.770    0.030   26.076    0.000
   .A3                0.694    0.036   19.186    0.000
   .A4                1.645    0.052   31.366    0.000
   .A5                0.965    0.035   27.652    0.000
   .C1                1.048    0.036   29.420    0.000
   .C2                1.084    0.039   27.560    0.000
   .C3                1.143    0.039   29.393    0.000
   .C4                0.990    0.042   23.804    0.000
   .C5                1.726    0.061   28.320    0.000
   .E1                1.682    0.058   29.081    0.000
   .E2                1.182    0.052   22.716    0.000
   .E3                1.224    0.041   30.002    0.000
   .E4                1.106    0.044   25.256    0.000
   .E5                1.306    0.042   31.131    0.000
   .N1                0.766    0.037   20.793    0.000
   .N2                0.812    0.036   22.814    0.000
   .N3                1.234    0.043   28.397    0.000
   .N4                1.703    0.053   32.241    0.000
   .N5                1.982    0.060   32.950    0.000
   .O1                0.881    0.034   26.289    0.000
   .O2                1.884    0.064   29.555    0.000
   .O3                0.849    0.040   21.122    0.000
   .O4                1.305    0.039   33.218    0.000
   .O5                1.278    0.046   27.702    0.000
    Agreeableness     0.288    0.033    8.601    0.000
    Conscientisnss    0.477    0.037   12.866    0.000
    Extraversion      0.978    0.067   14.564    0.000
    Neuroticism       1.716    0.074   23.242    0.000
    Openness          0.388    0.034   11.331    0.000
```

b) Dürfen Sie die beiden Modelle mit einem χ^2-Test vergleichen? Was ist die Voraussetzung

dafür?

Lösung

Ja, die Modelle lassen sich vergleichen, weil das Modell *ohne* Kovarianzen eine restringierte Version des Modells *mit* Kovarianzen ist. Es werden lediglich ein paar Parameter auf Null gesetzt. Das Modell `fit_bfi_5F_noCV` ist somit *genestet* in dem Modell `fit_bfi_5F`. Diese Nestung ist genau die Voraussetzung, um verschiedene Modelle mit einem χ^2-Test zu vergleichen.

c) Welches Modell passt besser auf die Daten?

Lösung

```
anova(fit_bfi_5F_noCV, fit_bfi_5F)
```

```
Chi-Squared Difference Test

                Df    AIC    BIC  Chisq Chisq diff   RMSEA Df diff Pr(>Chisq)
fit_bfi_5F      265 199800 200148 4165.5
fit_bfi_5F_noCV 275 201255 201545 5639.7     1474.2 0.24517      10  < 2.2e-16

fit_bfi_5F
fit_bfi_5F_noCV ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Wir sehen hier, dass das Modell ohne Kovarianzen einen viel grösseren χ^2-Wert hat. Dieser Unterschied ist signifikant. Das bedeutet, dass das Modell mit mehr Parameter signifikant besser auf die Daten passt, als das Modell mit weniger Parameter (ohne Kovarianzen). Das ist nicht verwunderlich, wenn wir berücksichtigen, dass im Modell *mit* Kovarianzen alle Kovarianzen der latenten Variablen signifikant waren (der Modellvergleich ist äquivalent zu einem simultanen Signifikanztest für alle 10 Kovarianzparameter).

7. Strukturgleichungsmodelle (SEM)

7.1. Einführung

In dieser Übung soll vermittelt werden, wie mit `lavaan` Strukturgleichungsmodelle (SEM) definiert und berechnet werden können. SEM stellen eine Kombination aus CFA und Regressionsmodellen dar und ermöglichen es, komplexe Regressionsstrukturen (Pfadmodelle) auf der Ebene latenter Variablen zu modellieren. Dabei können auch indirekte Effekte (Mediationsmodelle) geschätzt werden. Die Schätztheorie unterscheidet sich nicht von derjenigen der CFA.

Mit einem Strukturgleichungsmodell wollen wir die Effekte der durch Jugendliche erlebten emotionalen Unterstützung durch Eltern und Freunde auf deren Lebenszufriedenheit untersuchen. Dabei sollen die beiden Selbstwirksamkeits-Komponenten "Akademische Selbstwirksamkeit" und "Soziale Selbstwirksamkeit" als Mediatoren dienen. Es sollen sowohl direkte als auch indirekte Effekte untersucht werden.

Inhaltliche Annahmen: Sowohl elterliche Unterstützung als auch Unterstützung durch Freunde haben einen positiven Effekt auf die Lebenszufriedenheit. Ausserdem ist aus der Selbstwirksamkeitsforschung bekannt, dass sich erfahrene Unterstützung positiv auf das Selbstwirksamkeitserleben auswirkt, und dass dieses wiederum mit Lebenszufriedenheit im Zusammenhang steht. Eine wichtige Frage ist jedoch, welche differentiellen Effekte sich zeigen, wenn man die Unterstützung aus zwei wichtigen sozialen Kontexten (Eltern und Freunde) und die beiden Selbstwirksamkeitsdomänen "akademisch" und "sozial" gleichzeitig betrachtet.

Zu dem postulierten Strukturmodell (Beziehungen zwischen den latenten Variablen) benötigen wir ein Messmodell (Beziehungen zwischen den latenten Variablen und den manifesten Variablen, also die Faktorenstruktur). In diesem Beispiel wurden die Konstrukte durch folgende manifeste Variablen gemessen:

Emotionale Unterstützung durch Eltern: unt_eltern1, unt_eltern2

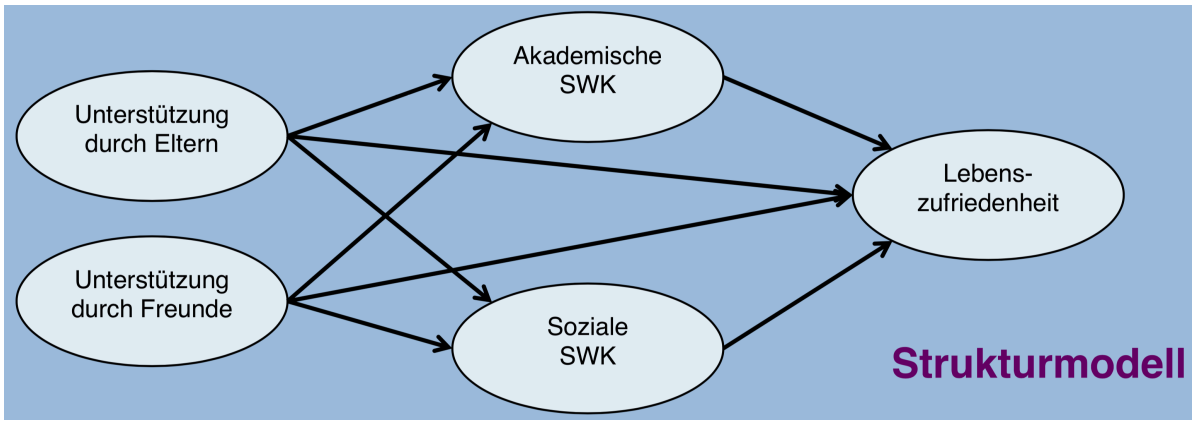
Emotionale Unterstützung durch Freunde: unt_freunde1, unt_freunde2

Akademische Selbstwirksamkeit: swk_akad1, swk_akad2, swk_akad3, swk_akad4, swk_akad5

Soziale Selbstwirksamkeit: swk_soz1, swk_soz2, swk_soz3, swk_soz4

Lebenszufriedenheit: leben1, leben2, leben3

Die angenommenen Effekte lassen sich in folgendem Strukturmodell zusammenfassen:



Wir werden die SEM-Analyse in drei Schritten vornehmen:

Zuerst wollen wir nur das Messmodell betrachten. Es handelt sich dabei um eine Multikonstrukt-CFA mit Interkorrelationen aller latenten Variablen.

Danach betrachten wir das gesamte Strukturgleichungsmodell (Messmodell und Strukturmodell zusammen).

Im letzten Schritt wollen wir zeigen, wie man innerhalb des Strukturgleichungsmodells explizit *indirekte* und *totale* Effekte definieren kann und wie man diese auf Signifikanz testet.

7.2. Packages laden und Datenimport

```
pacman::p_load(tidyverse, ggplot2, ggthemes, haven, lavaan, semPlot, wesanderson)
```

```
data <- read_csv("https://raw.githubusercontent.com/methodenlehre/data/master/statIV_sem/s
  mutate(
    westost = as.factor(westost),
    geschlecht = as.factor(geschlecht)
  )
```

Wie in der Übung zur CFA entfernen wir alle Personen aus dem Datensatz, die auf mindestens einer Variablen einen Wert ausserhalb von +/- 3 SD vom Mittelwert aufweisen.

```
keep <- function(x) {
  mx <- mean(x, na.rm = TRUE) # Wir speichern den Mittelwert der Variable
```

```
  sd3 <- 3 * sd(x, na.rm = TRUE) # Hier speichern wir die Standardabweichung * 3
  between(x, left = mx - sd3, right = mx + sd3)
}

data_clean <- data |>
  filter(rowMeans(sapply(data |> select(-ID, -westost, -geschlecht, -alter), keep), na.rm
```

7.3. Messmodell

Im Messmodell werden die Zusammenhänge der latenten Variablen (Faktoren) mit den manifesten Variablen definiert. Die `lavaan`-Modelldefinition kennen wir bereits von der CFA.

- Fehlervarianzen der beobachteten Variablen und Varianzen der exogenen latenten Variablen müssen nicht eigens spezifiziert werden
- Operator für das Messmodell: `=~` ("wird gemessen durch…")

```
model_measurement <- "
# Messmodell
UNT_Eltern     =~ unt_eltern1 + unt_eltern2
UNT_Freunde    =~ unt_freunde1 + unt_freunde2
SWK_Akademisch =~ swk_akad1 + swk_akad2 + swk_akad3 + swk_akad4 + swk_akad5
SWK_Sozial     =~ swk_soz1 + swk_soz2 + swk_soz3 + swk_soz4
ZUFRIEDEN      =~ leben1 + leben2 + leben3
"
```

Eigenschaften des Messmodells

Wie viele und welche manifesten Variablen hat das Modell?

16 manifeste Variablen (2 für UNT_Eltern, 2 für UNT_Freunde, 5 für SWK_Akademisch, 4 für SWK_Sozial, 3 für ZUFRIEDEN)

Wie viele Informationen enthält die Varianz-Kovarianz-Matrix der manifesten Variablen?

$(16 \cdot 17)/2 = 136$

Wie viele und welche Parameter müssen geschätzt werden?

11 Faktorladungen (eine für jede manifeste Variable minus Anzahl latenter Variablen, da die erste Ladung jeweils auf `1` fixiert wird)

16 Residualvarianzen der manifesten Variablen

5 Varianzen der latenten Variablen

10 Kovarianzen der latenten Variablen

Wie viele Freiheitsgrade besitzt das Modell?

$df = 136 - (11 + 16 + 5 + 10) = 136 - 42 = 94$

7.3.1. Modellschätzung

Die Funktion `sem()` wird auf das oben spezifizierte Modell `model_measurement` angewendet, die Ergebnisse werden in einem Objekt (hier mit dem beliebigen Namen `fit_measurement`) gespeichert. Man könnte hier auch `cfa()` verwenden. Die beiden Funktionen unterscheiden sich nur geringfügig in ihren Default-Einstellungen und sind beide gleichermassen für die Schätzung von Messmodellen geeignet.

```
fit_measurement <- sem(model_measurement,
  data = data_clean
)
summary(fit_measurement,
  fit.measures = TRUE,
  standardized = TRUE
)
```

```
lavaan 0.6.17 ended normally after 62 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        42

                                                  Used       Total
  Number of observations                           262         265

Model Test User Model:

  Test statistic                               288.018
  Degrees of freedom                                94
```

```
  P-value (Chi-square)                           0.000

Model Test Baseline Model:

  Test statistic                              2247.066
  Degrees of freedom                           120
  P-value                                        0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    0.909
  Tucker-Lewis Index (TLI)                       0.884

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -4040.188
  Loglikelihood unrestricted model (H1)      -3896.179

  Akaike (AIC)                                8164.376
  Bayesian (BIC)                              8314.247
  Sample-size adjusted Bayesian (SABIC)       8181.088

Root Mean Square Error of Approximation:

  RMSEA                                          0.089
  90 Percent confidence interval - lower         0.077
  90 Percent confidence interval - upper         0.101
  P-value H_0: RMSEA <= 0.050                    0.000
  P-value H_0: RMSEA >= 0.080                    0.895

Standardized Root Mean Square Residual:

  SRMR                                           0.061

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  UNT_Eltern =~
```

```
    unt_eltern1        1.000                               0.773    0.902
    unt_eltern2        1.052    0.088   11.961    0.000    0.813    0.830
  UNT_Freunde =~
    unt_freunde1       1.000                               0.756    0.759
    unt_freunde2       1.081    0.146    7.405    0.000    0.817    1.004
  SWK_Akademisch =~
    swk_akad1          1.000                               0.687    0.774
    swk_akad2          0.779    0.070   11.118    0.000    0.535    0.689
    swk_akad3          0.940    0.072   13.054    0.000    0.646    0.797
    swk_akad4          0.871    0.072   12.032    0.000    0.599    0.740
    swk_akad5          0.888    0.073   12.247    0.000    0.610    0.752
  SWK_Sozial =~
    swk_soz1           1.000                               0.543    0.786
    swk_soz2           1.146    0.088   13.070    0.000    0.623    0.794
    swk_soz3           0.884    0.071   12.474    0.000    0.480    0.760
    swk_soz4           1.094    0.093   11.766    0.000    0.594    0.722
  ZUFRIEDEN =~
    leben1             1.000                               0.396    0.464
    leben2             1.233    0.178    6.944    0.000    0.488    0.788
    leben3             1.500    0.218    6.891    0.000    0.594    0.766

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  UNT_Eltern ~~
    UNT_Freunde        0.098    0.041    2.364    0.018    0.167    0.167
    SWK_Akademisch     0.229    0.042    5.403    0.000    0.431    0.431
    SWK_Sozial         0.183    0.034    5.438    0.000    0.436    0.436
    ZUFRIEDEN          0.207    0.038    5.501    0.000    0.677    0.677
  UNT_Freunde ~~
    SWK_Akademisch     0.050    0.035    1.418    0.156    0.096    0.096
    SWK_Sozial         0.154    0.036    4.223    0.000    0.376    0.376
    ZUFRIEDEN          0.092    0.028    3.320    0.001    0.307    0.307
  SWK_Akademisch ~~
    SWK_Sozial         0.243    0.035    6.981    0.000    0.651    0.651
    ZUFRIEDEN          0.158    0.031    5.059    0.000    0.582    0.582
  SWK_Sozial ~~
    ZUFRIEDEN          0.147    0.027    5.390    0.000    0.682    0.682

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .unt_eltern1        0.136    0.043    3.170    0.002    0.136    0.186
   .unt_eltern2        0.300    0.053    5.685    0.000    0.300    0.312
   .unt_freunde1       0.420    0.080    5.260    0.000    0.420    0.424
```

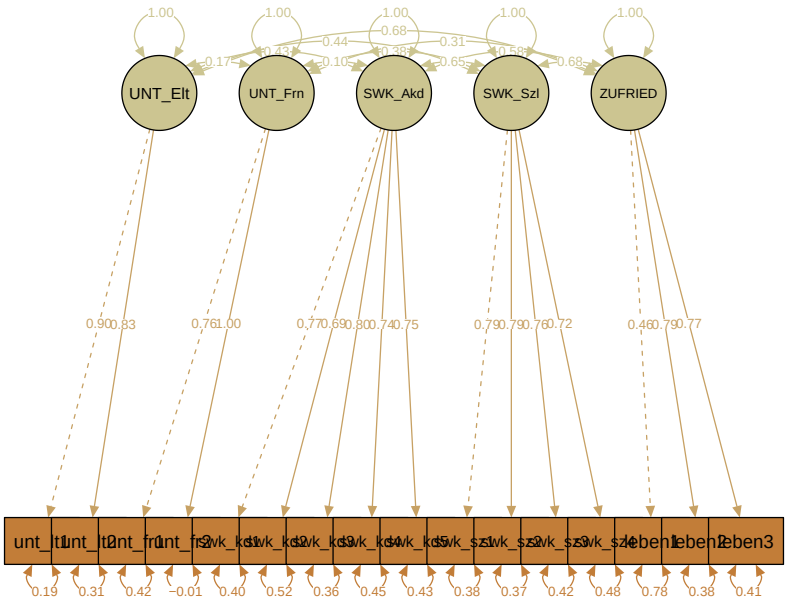
```
.unt_freunde2     -0.005    0.083   -0.059    0.953   -0.005   -0.007
.swk_akad1         0.316    0.035    9.090    0.000    0.316    0.401
.swk_akad2         0.317    0.032   10.032    0.000    0.317    0.525
.swk_akad3         0.239    0.028    8.696    0.000    0.239    0.364
.swk_akad4         0.296    0.031    9.546    0.000    0.296    0.452
.swk_akad5         0.286    0.030    9.402    0.000    0.286    0.435
.swk_soz1          0.183    0.021    8.721    0.000    0.183    0.382
.swk_soz2          0.228    0.027    8.578    0.000    0.228    0.370
.swk_soz3          0.168    0.018    9.145    0.000    0.168    0.422
.swk_soz4          0.325    0.034    9.630    0.000    0.325    0.479
.leben1            0.572    0.053   10.793    0.000    0.572    0.785
.leben2            0.146    0.020    7.281    0.000    0.146    0.379
.leben3            0.249    0.032    7.858    0.000    0.249    0.414
 UNT_Eltern        0.597    0.075    7.930    0.000    1.000    1.000
 UNT_Freunde       0.571    0.106    5.399    0.000    1.000    1.000
 SWK_Akademisch    0.472    0.066    7.100    0.000    1.000    1.000
 SWK_Sozial        0.295    0.041    7.219    0.000    1.000    1.000
 ZUFRIEDEN         0.157    0.043    3.630    0.000    1.000    1.000
```

7.3.2. Darstellung als Pfaddiagramm

Hier soll auch gezeigt werden, wie mit `semPaths()` aus dem Package `semTools` Pfaddiagramme aus den gefitteten `lavaan`-Objekten generiert werden können. Wir lassen uns das Modell mit den standardisierten Parameter Estimates mit `layout = 'spring'` darstellen.

```r
# Wir können schon jetzt die Farben für semPaths definieren, nämlich als Liste
# mit den Farb-Parametern für manifeste (man) und latente (lat) Variablen.
cols <- wes_palette(
  name = "Moonrise2",
  n = 4,
  type = "discrete"
)
colorlist <- list(
  man = cols[2],
  lat = cols[3]
)

semPaths(fit_measurement,
  what = "col", # Gleichmässige Pfade
  whatLabels = "std", # Standardisierte Werte
  style = "mx",
```

```
  color = colorlist,
  rotation = 1,
  layout = "tree", # Layout der Darstellung
  nCharNodes = 7, # Anzahl Buchstaben pro Variable (Abkürzung)
  shapeMan = "rectangle", # Form der manifesten Variable
  sizeMan = 8, # Breite der manifesten Variablen
  sizeMan2 = 5
) # Höhe der manifesten Variablen
```



7.3.3. Modell-Fit

Wie aus dem `summary()`-Output und der graphischen Darstellung ersichtlich, sind die standardisierten Ladungen der Parcels auf den Faktoren alle relativ hoch und auch homogen. Die Faktor-Korrelationen dieses Multikonstrukt-Messmodells liegen zwischen $r = 0.096$ (`UNT_Freunde` $\leftrightarrow$ `SWK_Akademisch`) und $r = 0.682$ (`SWK_Sozial` $\leftrightarrow$ `ZUFRIEDEN`).

Die globale Modellpassung ist mit CFI = 0.909, NNFI/TLI = 0.884, RMSEA = 0.089 (90 %-CI: [0.077; 0.101]) nicht wirklich gut, kann aber nach den gängigen Kriterien als gerade noch akzeptabel bezeichnet werden.

Um festzustellen, wo Probleme in der Modellpassung bestehen, können wir zum einen die standardisierte Residual-Varianz-Kovarianz-Matrix betrachten:

```
resid(fit_measurement,
  type = "standardized"
)$cov
```

```
            unt_l1 unt_l2 unt_f1 unt_f2 swk_k1 swk_k2 swk_k3 swk_k4 swk_k5
unt_eltern1  0.000
unt_eltern2  0.000  0.000
unt_freunde1 -0.474 -0.709  0.000
unt_freunde2  0.057 -0.089  0.000  0.000
swk_akad1    -1.768 -2.728  0.460 -1.627  0.000
swk_akad2     1.293  1.301  1.802  0.446 -0.586  0.000
swk_akad3     1.350 -0.127  0.585 -0.009  1.178 -0.594  0.000
swk_akad4    -0.255 -0.690  2.783  0.838 -0.155 -0.807  1.672  0.000
swk_akad5     0.814  0.468 -0.140  0.576  0.075  2.525 -1.125 -3.025  0.000
swk_soz1      0.527 -0.737  1.612  0.100 -0.188  0.667  0.177  3.558  2.190
swk_soz2      1.531  1.755  0.521  0.056 -0.339 -2.528 -1.886  3.789  1.447
swk_soz3     -1.629 -0.105  2.205  2.661 -1.131 -3.023 -1.318 -0.172 -1.587
swk_soz4     -1.195 -0.462 -2.385 -3.193  1.279 -1.954 -1.731  1.369  0.714
leben1       -0.717 -1.193 -1.280  0.972 -2.441 -1.455 -2.085 -2.858 -1.060
leben2       -5.017 -4.083  3.097  3.955 -0.934  0.926 -0.118 -1.326  1.309
leben3        4.735  4.796 -1.702 -4.460 -0.506  3.922 -0.162  0.350  1.800
            swk_s1 swk_s2 swk_s3 swk_s4 leben1 leben2 leben3
unt_eltern1
unt_eltern2
unt_freunde1
unt_freunde2
swk_akad1
swk_akad2
swk_akad3
swk_akad4
swk_akad5
swk_soz1      0.000
swk_soz2      0.028  0.000
swk_soz3     -1.839 -0.046  0.000
swk_soz4     -1.251  0.017  2.790  0.000
leben1        1.733  0.979  2.644  2.321  0.000
leben2        2.835  1.249  2.497  1.516  0.836  0.000
leben3       -0.539 -2.619 -5.345 -2.875 -0.875 -0.021  0.000
```

Hier zeigt sich, dass die vom Modell implizierte Varianz-Kovarianz-Matrix an vielen Stellen des Modells signifikant von der empirischen Varianz-Kovarianz-Matrix abweicht. Eine Häu-

fung sehr hoher standardisierter Residual-Kovarianzen zeigt sich z.B. bezüglich des Parcels `leben3`.

Eine weitere Möglichkeit zur lokalen Fit-Diagnostik bieten die Modell-Modifikationsindizes. Diese zeigen an, bei welchen zusätzlichen zu schätzenden Modellparametern sich eine deutliche Fit-Verbesserung ergeben würde (im Sinne einer Reduktion von χ^2 um den Wert des entsprechenden MI). Wir lassen uns hier nur die MI ≥ 5 ausgeben, da üblicherweise nur solche als substantiell angesehen werden.

```
modindices(fit_measurement,
  standardized = FALSE,
  minimum.value = 5
)
```

```
               lhs op       rhs     mi    epc
50      UNT_Eltern =~ swk_akad1  6.363 -0.159
60      UNT_Eltern =~    leben2 31.249 -0.454
61      UNT_Eltern =~    leben3 39.880  0.628
71     UNT_Freunde =~  swk_soz3  7.365  0.114
72     UNT_Freunde =~  swk_soz4  9.594 -0.175
74     UNT_Freunde =~    leben2 16.164  0.190
75     UNT_Freunde =~    leben3 21.393 -0.268
80  SWK_Akademisch =~  swk_soz1  5.215  0.166
82  SWK_Akademisch =~  swk_soz3  7.140 -0.180
84  SWK_Akademisch =~    leben1  6.835 -0.268
94      SWK_Sozial =~ swk_akad4 10.514  0.351
97      SWK_Sozial =~    leben2 21.031  0.556
98      SWK_Sozial =~    leben3 36.446 -0.895
145   unt_freunde1 ~~ swk_akad4  8.497  0.069
146   unt_freunde1 ~~ swk_akad5  6.802 -0.061
151   unt_freunde1 ~~    leben1  9.867 -0.097
165   unt_freunde2 ~~    leben3  5.650 -0.046
173      swk_akad1 ~~  swk_soz4  6.213  0.059
179      swk_akad2 ~~ swk_akad5  7.603  0.064
181      swk_akad2 ~~  swk_soz2  6.587 -0.051
186      swk_akad2 ~~    leben3 16.436  0.086
196      swk_akad4 ~~ swk_akad5  7.637 -0.065
198      swk_akad4 ~~  swk_soz2 13.384  0.072
222       swk_soz3 ~~  swk_soz4  9.801  0.060
223       swk_soz3 ~~    leben1  7.223  0.058
224       swk_soz3 ~~    leben2  6.919  0.034
225       swk_soz3 ~~    leben3 23.662 -0.079
```

Es werden insgesamt 24 Modifikationsindizes ausgegeben. Sie beziehen sich entweder auf mögliche Querladungen (z.B. `UNT_Eltern =~ leben3`: MI = 39.880) oder auf Residualkovarianzen zwischen manifesten Variablen (z.B. `unt_freunde1 ~~ swk_akad4`: MI = 8.497). Im Sinne unseres theoretischen Modells (bestimmte inhaltlich definierte Items messen bestimmte latente Konstrukte) ist die Hinzufügung von Querladungen und Residualkovarianzen nicht sinnvoll.

Solche Model-Fit-Probleme ergeben sich nicht selten bei Self-Report-Studien. Beim Antworten auf Likert-Skalen bestehen oft individuelle Unterschiede in der Skalennutzung (sog. Antwortstile oder Response Styles). Diese können zu generell erhöhten Korrelationen der Variablen untereinander führen, die ggf. in einem Multikonstrukt-Messmodell keine angemessene Berücksichtigung finden können. Es gibt hierzu Lösungsansätze (Definition von Methoden- und Response-Style-Faktoren), die wir hier aber nicht weiter vertiefen.

Ein weiterer Grund für die Model-Fit Probleme in dieser Studie könnte die allgemeine Ähnlichkeit der untersuchten Konstrukte in Bezug auf die dort erfragten Lebensbereiche sein. Beispielsweise enthalten die Lebenszufriedenheits-Items solche zum Bereich "Freunde", der Freunde-Kontext spielt aber auch bei den Items zur sozialen Selbstwirksamkeit und natürlich beim denen zur Unterstützung durch Freunde eine grosse Rolle.

Da der Model-Fit noch akzeptabel ist, arbeiten wir mit diesem Messmodell weiter und wollen im nächsten Schritt die unseren inhaltlichen Hypothesen entsprechenden Effekte zwischen den latenten Variablen modellieren (Strukturmodell).

7.4. Gesamtmodell

7.4.1. Modelldefinition

Zusätzlich zum Messmodell wird im Gesamtmodell auch das Strukturmodell definiert. Das Strukturmodell repräsentiert die Zusammenhänge/Effekte zwischen den latenten Variablen.

```
model <- "
# Messmodell
UNT_Eltern     =~ unt_eltern1 + unt_eltern2
UNT_Freunde    =~ unt_freunde1 + unt_freunde2
SWK_Akademisch =~ swk_akad1 + swk_akad2 + swk_akad3 + swk_akad4 + swk_akad5
SWK_Sozial     =~ swk_soz1 + swk_soz2 + swk_soz3 + swk_soz4
ZUFRIEDEN      =~ leben1 + leben2 + leben3

# Strukturmodell
# Regressionsgleichungen
```

```
SWK_Akademisch ~ UNT_Eltern + UNT_Freunde
SWK_Sozial     ~ UNT_Eltern + UNT_Freunde
ZUFRIEDEN      ~ SWK_Akademisch + SWK_Sozial + UNT_Eltern + UNT_Freunde

# Residual-Kovarianzen
SWK_Akademisch ~~ SWK_Sozial
"
```

- Operator für das Strukturmodell (Regressionen latenter Variablen): `~` ("wird vorhergesagt durch…")
- Operator für (Residual-)Varianzen und Kovarianzen: `~~` (bei Varianzen steht links und rechts dieselbe Variable, bei Kovarianzen unterschiedliche). Wie schon im Messmodell (CFA) müssen Varianzen und Kovarianzen exogener latenter Variablen nicht angegeben werden (werden automatisch geschätzt). Auch die Residualvarianzen endogener latenter Variablen werden automatisch geschätzt.

Wir müssen daher nur *einen* Parameter mit `~~` spezifizieren: In diesem Modell haben wir zwei parallele latente Mediatorvariablen (`SWK_Akademisch` und `SWK_Sozial`), die keinerlei Effekte aufeinander haben. Es ist aber anzunehmen, dass diese beiden Variablen kovariieren (vgl. auch die Korrelation von $r = 0.651$ im Messmodell oben), da neben den bereichsspezifischen Selbstwirksamkeiten auch eine übergeordnete allgemeine Selbstwirksamkeit angenommen werden kann (vgl. auch die CFA zur Lebenszufriedenheit). Da es sich bei `SWK_Akademisch` und `SWK_Sozial` um endogene latente Variablen handelt (beide werden sowohl von `UNT_Eltern` als auch von `UNT_Freunde` vorhergesagt), muss hier eine Residualkovarianz $\psi =$ `SWK_Akademisch ~~ SWK_Sozial` spezifiziert werden.

Eigenschaften des Gesamtmodells

Wie viele und welche manifesten Variablen hat das Modell?

16 manifeste Variablen (2 für UNT_Eltern, 2 für UNT_Freunde, 5 für SWK_Akademisch, 4 für SWK_Sozial, 3 für ZUFRIEDEN)

Wie viele Informationen enthält die Varianz-Kovarianz-Matrix der manifesten Variablen?

$(16 \cdot 17)/2 = 136$

Wie viele und welche Parameter müssen geschätzt werden?

11 Faktorladungen (eine für jede manifeste Variable minus Anzahl latenter Variablen, da die erste Ladung jeweils auf `1` fixiert wird)

16 Residualvarianzen der manifesten Variablen

8 Strukturpfade

2 Varianzen der *exogenen* latenten Variablen

1 Kovarianz der beiden *exogenen* latenten Variablen

3 Residualvarianzen der *endogenen* latenten Variablen

1 Residualkovarianz der beiden *endogenen* latenten Variablen `SWK_Akademisch` und `SWK_Sozial`

Wie viele Freiheitsgrade besitzt das Modell?

$df = 136 - (11 + 16 + 8 + 2 + 1 + 3 + 1) = 136 - 42 = 94$

7.4.2. Modellschätzung

```r
fit <- sem(model,
  data = data_clean
)

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

```
lavaan 0.6.17 ended normally after 53 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        42

                                                  Used       Total
  Number of observations                           262         265

Model Test User Model:

  Test statistic                               288.018
  Degrees of freedom                                94
  P-value (Chi-square)                           0.000

Model Test Baseline Model:
```

```
  Test statistic                                2247.066
  Degrees of freedom                                120
  P-value                                         0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                     0.909
  Tucker-Lewis Index (TLI)                        0.884

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)               -4040.188
  Loglikelihood unrestricted model (H1)       -3896.179

  Akaike (AIC)                                 8164.376
  Bayesian (BIC)                               8314.247
  Sample-size adjusted Bayesian (SABIC)        8181.088

Root Mean Square Error of Approximation:

  RMSEA                                           0.089
  90 Percent confidence interval - lower          0.077
  90 Percent confidence interval - upper          0.101
  P-value H_0: RMSEA <= 0.050                     0.000
  P-value H_0: RMSEA >= 0.080                     0.895

Standardized Root Mean Square Residual:

  SRMR                                            0.061

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  UNT_Eltern =~
    unt_eltern1       1.000                               0.773    0.902
    unt_eltern2       1.052    0.088   11.961    0.000    0.813    0.830
  UNT_Freunde =~
```

```
    unt_freunde1       1.000                               0.756    0.759
    unt_freunde2       1.081    0.146    7.405    0.000    0.817    1.004
  SWK_Akademisch =~
    swk_akad1          1.000                               0.687    0.774
    swk_akad2          0.779    0.070   11.118    0.000    0.535    0.689
    swk_akad3          0.940    0.072   13.054    0.000    0.646    0.797
    swk_akad4          0.871    0.072   12.032    0.000    0.599    0.740
    swk_akad5          0.888    0.073   12.247    0.000    0.610    0.752
  SWK_Sozial =~
    swk_soz1           1.000                               0.543    0.786
    swk_soz2           1.146    0.088   13.070    0.000    0.623    0.794
    swk_soz3           0.884    0.071   12.474    0.000    0.480    0.760
    swk_soz4           1.094    0.093   11.766    0.000    0.594    0.722
  ZUFRIEDEN =~
    leben1             1.000                               0.396    0.464
    leben2             1.233    0.178    6.944    0.000    0.488    0.788
    leben3             1.500    0.218    6.891    0.000    0.594    0.766

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  SWK_Akademisch ~
    UNT_Eltern        0.379    0.065    5.797    0.000    0.427    0.427
    UNT_Freunde       0.022    0.057    0.395    0.693    0.025    0.025
  SWK_Sozial ~
    UNT_Eltern        0.270    0.049    5.502    0.000    0.384    0.384
    UNT_Freunde       0.224    0.046    4.892    0.000    0.311    0.311
  ZUFRIEDEN ~
    SWK_Akademisch    0.085    0.051    1.671    0.095    0.147    0.147
    SWK_Sozial        0.265    0.077    3.439    0.001    0.363    0.363
    UNT_Eltern        0.226    0.046    4.934    0.000    0.441    0.441
    UNT_Freunde       0.043    0.032    1.373    0.170    0.083    0.083

Covariances:
                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
 .SWK_Akademisch ~~
   .SWK_Sozial         0.170    0.027    6.265    0.000    0.598    0.598
  UNT_Eltern ~~
    UNT_Freunde        0.098    0.041    2.364    0.018    0.167    0.167

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .unt_eltern1       0.136    0.043    3.170    0.002    0.136    0.186
   .unt_eltern2       0.300    0.053    5.685    0.000    0.300    0.312
```

```
.unt_freunde1      0.420    0.080    5.260    0.000    0.420    0.424
.unt_freunde2     -0.005    0.083   -0.059    0.953   -0.005   -0.007
.swk_akad1         0.316    0.035    9.090    0.000    0.316    0.401
.swk_akad2         0.317    0.032   10.032    0.000    0.317    0.525
.swk_akad3         0.239    0.028    8.696    0.000    0.239    0.364
.swk_akad4         0.296    0.031    9.546    0.000    0.296    0.452
.swk_akad5         0.286    0.030    9.402    0.000    0.286    0.435
.swk_soz1          0.183    0.021    8.721    0.000    0.183    0.382
.swk_soz2          0.228    0.027    8.578    0.000    0.228    0.370
.swk_soz3          0.168    0.018    9.145    0.000    0.168    0.422
.swk_soz4          0.325    0.034    9.630    0.000    0.325    0.479
.leben1            0.572    0.053   10.793    0.000    0.572    0.785
.leben2            0.146    0.020    7.281    0.000    0.146    0.379
.leben3            0.249    0.032    7.858    0.000    0.249    0.414
 UNT_Eltern        0.597    0.075    7.930    0.000    1.000    1.000
 UNT_Freunde       0.571    0.106    5.399    0.000    1.000    1.000
.SWK_Akademisch    0.384    0.056    6.879    0.000    0.814    0.814
.SWK_Sozial        0.211    0.031    6.827    0.000    0.715    0.715
.ZUFRIEDEN         0.054    0.017    3.220    0.001    0.343    0.343
```

7.4.3. Modell-Fit

Der Model Fit ist genau gleich geblieben! Wir haben schon oben gesehen, dass wir im Gesamtmodell genauso viele Parameter und damit Freiheitsgrade haben wie im Messmodell.

Das bedeutet, dass das Strukturmodell saturiert ist!

Im Messmodell hatten wir 15 Varianz- und Kovarianzparameter der latenten Variablen, somit war die Varianz-Kovarianzmatrix der latenten Variablen (mit 5*6/2 = 15 Elementen) vollständig und unrestringiert. Im Strukturmodell haben wir nun auch 15 Parameter. Die Parameter des Strukturmodells sind zwar schätzbar und wir können somit unsere postulierten Effekte überprüfen, aber dieser Teil des Gesamtmodells ist gerade so identifiziert (weil in Bezug auf die latenten Variablen gilt: $n_{Info} = n_{Par}$) und trägt damit nichts zur Überprüfung des Model Fits des Gesamtmodells bei! Anders ausgedrückt: Wir können nicht überprüfen, ob das Strukturmodell gut auf unsere Daten passt, nur der Fit des Messmodells ist überprüfbar.

Interpretation der geschätzten Strukturkoeffizienten:

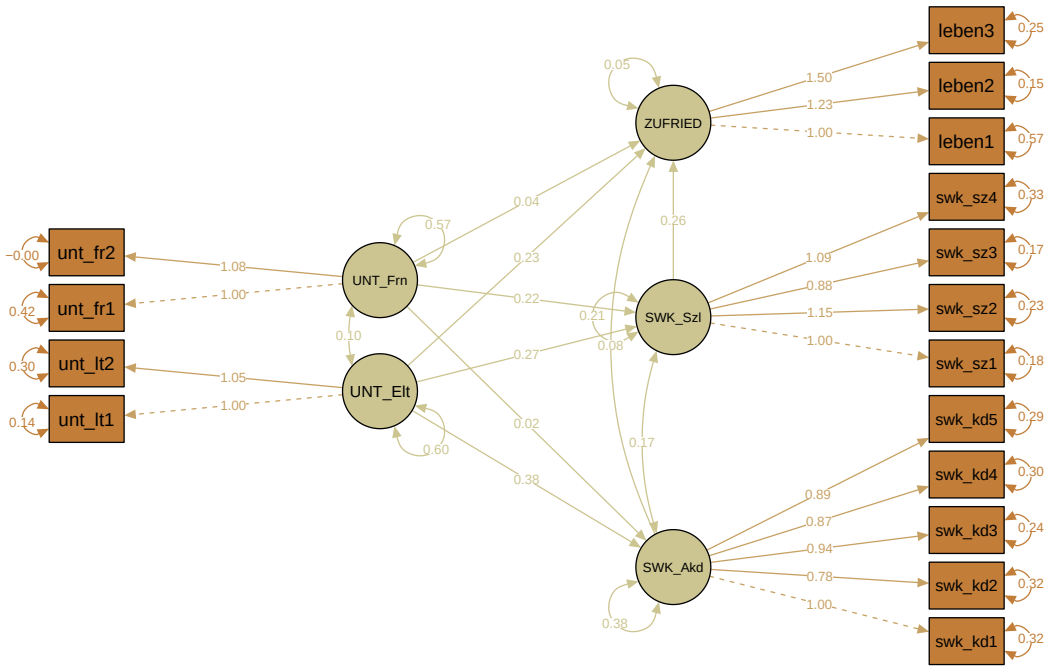
Von den 8 postulierten Strukturpfaden sind 5 signifikant und gehen in die erwartete Richtung. Nicht signifikant sind die Effekte von `UNT_Freunde` auf `SWK_Akademisch` (Std.all = 0.025, $p = 0.693$), von `UNT_Freunde` auf `ZUFRIEDEN` (Std.all = 0.083, $p = 0.170$) und von `SWK_Akademisch` auf `ZUFRIEDEN` (Std.all = 0.147, $p = 0.095$).

Besonders auffällig ist, dass die wahrgenommene Unterstützung durch die Eltern einen substantiellen direkten Effekt auf die Zufriedenheit hat, während der direkte Effekt der wahrgenommenen Unterstützung durch die Freunde nicht signifikant war. Zudem erstaunt, dass die soziale Selbstwirksamkeit einen direkten Einfluss auf die Zufriedenheit hat, aber dass die akademische Selbstwirksamkeit keinen signifikanten direkten Einfluss auf die Zufriedenheit hat.

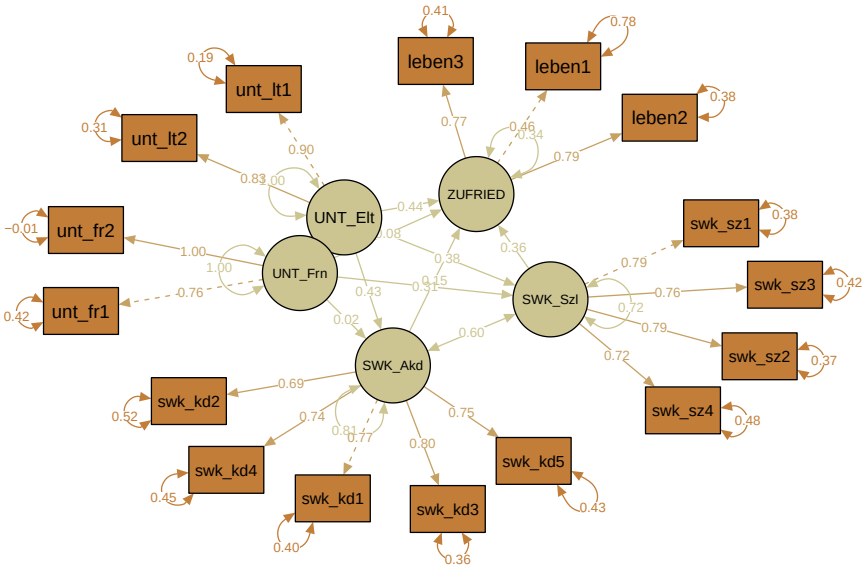
7.4.4. Darstellung als Pfaddiagramm

Wir lassen uns das Modell jetzt auf zwei verschiedene Arten darstellen, einmal mit den unstandardisierten Parameter Estimates (`whatLabels = "par"`) mit `layout = 'tree2'` und einmal mit den standardisierten Parameter Estimates (`whatLabels = "std"`) mit `layout = 'spring'`.

```
semPaths(fit,
  what = "col",
  whatLabels = "par",
  style = "mx",
  color = colorlist,
  rotation = 2,
  layout = "tree2",
  mar = c(1, 2, 1, 2),
  nCharNodes = 7,
  shapeMan = "rectangle",
  sizeMan = 8,
  sizeMan2 = 5
)
```




```r
semPaths(fit,
  what = "col",
  whatLabels = "std",
  style = "mx",
  color = colorlist,
  rotation = 1,
  layout = "spring",
  nCharNodes = 7,
  shapeMan = "rectangle",
  sizeMan = 8,
  sizeMan2 = 5
)
```



Typische Aufgabe: Berechnung indirekter und totaler Effekte

Alle Berechnungen sollen mit den standardisierten Estimates (`Std.all`) durchgeführt werden!

Spezifische indirekte Effekte

1. `UNT_Eltern` → `SWK_Akademisch` → `ZUFRIEDEN`:

$0.427 \cdot 0.147 = 0.063$

2. `UNT_Eltern` → `SWK_Sozial` → `ZUFRIEDEN`:

$0.384 \cdot 0.363 = 0.139$

3. `UNT_Freunde` → `SWK_Akademisch` → `ZUFRIEDEN`:

$0.025 \cdot 0.147 = 0.004$

4. `UNT_Freunde` → `SWK_Sozial` → `ZUFRIEDEN`:

$0.311 \cdot 0.363 = 0.113$

Totale indirekte Effekte

1. `UNT_Eltern` → `ZUFRIEDEN` über `SWK_Akademisch` und `SWK_Sozial`:

$0.427 \cdot 0.147 + 0.384 \cdot 0.363 = 0.202$

2. `UNT_Freunde` $\rightarrow$ `ZUFRIEDEN` über `SWK_Akademisch` und `SWK_Sozial`:

$0.025 \cdot 0.147 + 0.311 \cdot 0.363 = 0.117$

Totale Effekte

1. `UNT_Eltern` $\rightarrow$ `ZUFRIEDEN`:

$0.427 \cdot 0.147 + 0.384 \cdot 0.363 + 0.441 = 0.643$

2. `UNT_Freunde` $\rightarrow$ `ZUFRIEDEN`:

$0.025 \cdot 0.147 + 0.311 \cdot 0.363 + 0.083 = 0.200$

Wir können indirekte und totale Effekte auch in `lavaan` schätzen lassen und dort dann auch Signifikanztests für diese erhalten. Das ist unser letzter Schritt in der Analyse des vorliegenden Strukturgleichungsmodells.

7.5. Testung indirekter und totaler Effekte

Sobald im Strukturmodell eine oder mehrere latente Mediatorvariablen (d.h. solche, die sowohl Prädiktor als auch Prädikand anderer latenter Variablen sind) vorhanden sind, handelt es sich um eine Mediationsanalyse. Um indirekte und totale Effekte zu schätzen, müssen die Parameter des Strukturmodells zuerst in der Modelldefinition benannt werden (Vormultiplikation mit b1, b2, b3 usw.)

- Spezifische (und ggf. totale) indirekte Effekte können dann als Produkte (bzw. Summe der Produkte) der Pfadparameter definiert werden (Operator: `:=`)
- Totale Effekte können gleichermassen als Summe von indirekten und direkten Effekten definiert werden.

Die Schätzung (und Testung) indirekter und totaler Effekte sollte mit der Option `se = "bootstrap"` durchgeführt werden, da ein Produkt zweier (oder mehrerer) Pfadkoeffizienten nicht wie die Pfadkoeffizienten selber approximativ normalverteilt ist.

```
model_mediation <- "
# Messmodell
UNT_Eltern     =~ unt_eltern1 + unt_eltern2
UNT_Freunde    =~ unt_freunde1 + unt_freunde2
SWK_Akademisch =~ swk_akad1 + swk_akad2 + swk_akad3 + swk_akad4 + swk_akad5
SWK_Sozial     =~ swk_soz1 + swk_soz2 + swk_soz3 + swk_soz4
```

```
ZUFRIEDEN      =~ leben1 + leben2 + leben3

# Strukturmodell
# regressions
SWK_Akademisch ~ b1 * UNT_Eltern + b3 * UNT_Freunde
SWK_Sozial     ~ b2 * UNT_Eltern + b4 * UNT_Freunde
ZUFRIEDEN      ~ b5 * UNT_Eltern + b6 * UNT_Freunde + b7 * SWK_Akademisch + b8 * SWK_Sozia

# residual covariances
SWK_Akademisch ~~ SWK_Sozial

# indirect effects
b1b7 := b1 * b7
b2b8 := b2 * b8
totalind_eltern  := b1b7 + b2b8
b3b7 := b3 * b7
b4b8 := b4 * b8
totalind_freunde := b3b7 + b4b8

# total effects
total_eltern  := totalind_eltern + b5
total_freunde := totalind_freunde + b6
"
```

```
# Jetzt mit Bootstrap:
# Der iseed-Befehl ist nur für die Replizierbarkeit (vergleichbar mit set.seed())
fit_mediation <- sem(model_mediation,
  data = data_clean,
  se = "bootstrap",
  iseed = 123
)

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

```
lavaan 0.6.17 ended normally after 53 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        42
```

```
                                                  Used       Total
  Number of observations                           262         265

Model Test User Model:

  Test statistic                                288.018
  Degrees of freedom                             94
  P-value (Chi-square)                            0.000

Model Test Baseline Model:

  Test statistic                               2247.066
  Degrees of freedom                            120
  P-value                                         0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                     0.909
  Tucker-Lewis Index (TLI)                        0.884

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)               -4040.188
  Loglikelihood unrestricted model (H1)       -3896.179

  Akaike (AIC)                                 8164.376
  Bayesian (BIC)                               8314.247
  Sample-size adjusted Bayesian (SABIC)        8181.088

Root Mean Square Error of Approximation:

  RMSEA                                           0.089
  90 Percent confidence interval - lower          0.077
  90 Percent confidence interval - upper          0.101
  P-value H_0: RMSEA <= 0.050                     0.000
  P-value H_0: RMSEA >= 0.080                     0.895

Standardized Root Mean Square Residual:

  SRMR                                            0.061

Parameter Estimates:
```

```
  Standard errors                             Bootstrap
  Number of requested bootstrap draws             1000
  Number of successful bootstrap draws            1000

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  UNT_Eltern =~
    unt_eltern1       1.000                              0.773    0.902
    unt_eltern2       1.052    0.103   10.234    0.000    0.813    0.830
  UNT_Freunde =~
    unt_freunde1      1.000                              0.756    0.759
    unt_freunde2      1.081    0.193    5.595    0.000    0.817    1.004
  SWK_Akademisch =~
    swk_akad1         1.000                              0.687    0.774
    swk_akad2         0.779    0.068   11.403    0.000    0.535    0.689
    swk_akad3         0.940    0.064   14.618    0.000    0.646    0.797
    swk_akad4         0.871    0.066   13.169    0.000    0.599    0.740
    swk_akad5         0.888    0.070   12.749    0.000    0.610    0.752
  SWK_Sozial =~
    swk_soz1          1.000                              0.543    0.786
    swk_soz2          1.146    0.086   13.360    0.000    0.623    0.794
    swk_soz3          0.884    0.076   11.574    0.000    0.480    0.760
    swk_soz4          1.094    0.106   10.319    0.000    0.594    0.722
  ZUFRIEDEN =~
    leben1            1.000                              0.396    0.464
    leben2            1.233    0.262    4.709    0.000    0.488    0.788
    leben3            1.500    0.351    4.274    0.000    0.594    0.766

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  SWK_Akademisch ~
    UNT_Eltrn (b1)    0.379    0.073    5.208    0.000    0.427    0.427
    UNT_Frend (b3)    0.022    0.067    0.335    0.738    0.025    0.025
  SWK_Sozial ~
    UNT_Eltrn (b2)    0.270    0.054    4.979    0.000    0.384    0.384
    UNT_Frend (b4)    0.224    0.046    4.823    0.000    0.311    0.311
  ZUFRIEDEN ~
    UNT_Eltrn (b5)    0.226    0.056    4.051    0.000    0.441    0.441
    UNT_Frend (b6)    0.043    0.037    1.163    0.245    0.083    0.083
    SWK_Akdms (b7)    0.085    0.055    1.530    0.126    0.147    0.147
    SWK_Sozil (b8)    0.265    0.107    2.464    0.014    0.363    0.363
```

```
Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
 .SWK_Akademisch ~~
   .SWK_Sozial         0.170    0.028    6.015    0.000    0.598    0.598
  UNT_Eltern ~~
    UNT_Freunde        0.098    0.039    2.505    0.012    0.167    0.167

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .unt_eltern1       0.136    0.052    2.610    0.009    0.136    0.186
   .unt_eltern2       0.300    0.062    4.844    0.000    0.300    0.312
   .unt_freunde1      0.420    0.103    4.069    0.000    0.420    0.424
   .unt_freunde2     -0.005    0.110   -0.044    0.965   -0.005   -0.007
   .swk_akad1         0.316    0.042    7.484    0.000    0.316    0.401
   .swk_akad2         0.317    0.037    8.504    0.000    0.317    0.525
   .swk_akad3         0.239    0.028    8.434    0.000    0.239    0.364
   .swk_akad4         0.296    0.033    8.918    0.000    0.296    0.452
   .swk_akad5         0.286    0.033    8.789    0.000    0.286    0.435
   .swk_soz1          0.183    0.030    6.028    0.000    0.183    0.382
   .swk_soz2          0.228    0.035    6.539    0.000    0.228    0.370
   .swk_soz3          0.168    0.018    9.210    0.000    0.168    0.422
   .swk_soz4          0.325    0.039    8.364    0.000    0.325    0.479
   .leben1            0.572    0.082    6.989    0.000    0.572    0.785
   .leben2            0.146    0.032    4.551    0.000    0.146    0.379
   .leben3            0.249    0.044    5.644    0.000    0.249    0.414
    UNT_Eltern        0.597    0.080    7.467    0.000    1.000    1.000
    UNT_Freunde       0.571    0.118    4.826    0.000    1.000    1.000
   .SWK_Akademisch    0.384    0.052    7.437    0.000    0.814    0.814
   .SWK_Sozial        0.211    0.029    7.242    0.000    0.715    0.715
   .ZUFRIEDEN         0.054    0.019    2.843    0.004    0.343    0.343

Defined Parameters:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
    b1b7              0.032    0.021    1.501    0.133    0.063    0.063
    b2b8              0.072    0.032    2.266    0.023    0.140    0.140
    totalind_eltrn    0.104    0.029    3.591    0.000    0.202    0.202
    b3b7              0.002    0.007    0.285    0.776    0.004    0.004
    b4b8              0.059    0.028    2.089    0.037    0.113    0.113
    totalind_frend    0.061    0.030    2.051    0.040    0.117    0.117
    total_eltern      0.330    0.066    4.961    0.000    0.643    0.643
    total_freunde     0.104    0.047    2.239    0.025    0.199    0.199
```

Anmerkung: Auch die Standardfehler der anderen Parameter wurden jetzt per Bootstrap ge-

schätzt. Daher haben sich die p-Werte leicht verändert. An den Signifikanzentscheidungen ändert das aber nichts!

Welche der von Hand berechneten indirekten und totalen Effekte sind nun signifikant?

Spezifische indirekte Effekte

1. `UNT_Eltern` → `SWK_Akademisch` → `ZUFRIEDEN`:

Dieser indirekte Effekt ist nicht signifikant, `b1b7` $= 0.063, p = 0.133$.

2. `UNT_Eltern` → `SWK_Sozial` → `ZUFRIEDEN`:

Dieser indirekte Effekt ist signifikant, `b2b8` $= 0.14, p = 0.023$.

3. `UNT_Freunde` → `SWK_Akademisch` → `ZUFRIEDEN`:

Dieser indirekte Effekt ist nicht signifikant, `b3b7` $= 0.004, p = 0.776$.

4. `UNT_Freunde` → `SWK_Sozial` → `ZUFRIEDEN`:

Dieser indirekte Effekt ist signifikant, `b4b8` $= 0.113, p = 0.037$.

Totale indirekte Effekte

1. `UNT_Eltern` → `ZUFRIEDEN` über `SWK_Akademisch` und `SWK_Sozial`:

Dieser totale indirekte Effekt ist signifikant, `totalind_eltern` $= 0.202, p < 0.001$.

2. `UNT_Freunde` → `ZUFRIEDEN` über `SWK_Akademisch` und `SWK_Sozial`:

Dieser totale indirekte Effekt ist signifikant, `totalind_freunde` $= 0.117, p = 0.04$.

Totale Effekte

1. `UNT_Eltern` → `ZUFRIEDEN`:

Dieser totale Effekt ist signifikant, `total_eltern` $= 0.643, p < 0.001$.

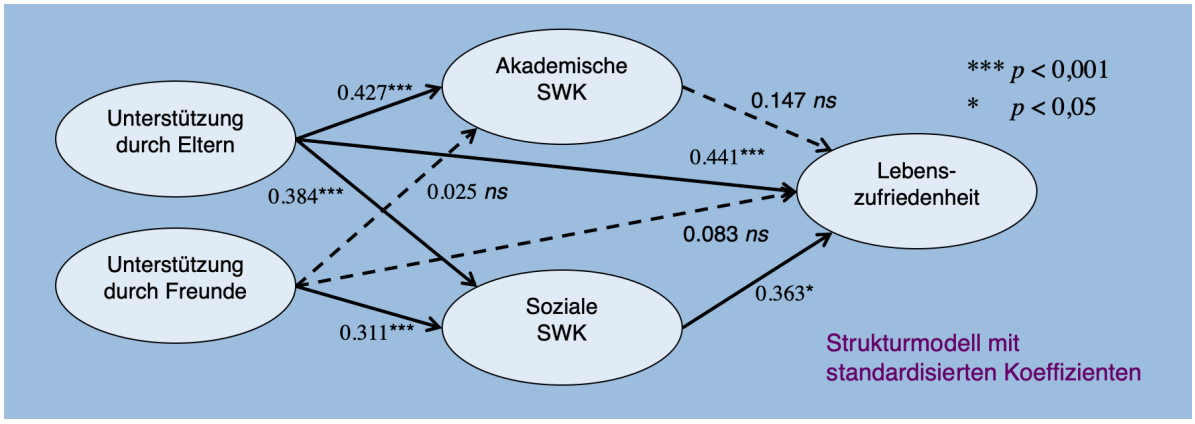
2. `UNT_Freunde` → `ZUFRIEDEN`:

Dieser totale Effekt ist signifikant, `total_freunde` $= 0.199, p = 0.025$.

7.6. Zusammenfassung

- **Gerade noch akzeptabler Modell-Fit des Messmodells**
- **Strukturmodell saturiert, Modellpassung nicht überprüfbar**
- **Unterstützung durch *Eltern* am wichtigsten für die Lebenszufriedenheit Jugendlicher**
  - Totaler Effekt mehr als doppelt so gross im Vergleich zu Unterstützung durch Freunde
  - Direkter Effekt am stärksten, aber auch indirekter Effekt über soziale Selbstwirksamkeit substantiell
- **Unterstützung durch Freunde hat nur indirekt über die soziale Selbstwirksamkeit einen Effekt auf die Lebenszufriedenheit**
- **Akademische Selbstwirksamkeit unter Konstanthaltung der sozialen Selbstwirksamkeit irrelevant für Lebenszufriedenheit**

Zusammenfassung der Effekte im Strukturmodell:



7.7. Übung

Diese Übung basiert auf Daten einer Befragung von $n = 300$ Jugendlichen.

Wir laden zuerst die benötigten Packages:

```
pacman::p_load(lavaan, tidyverse, semPlot)
```

Jetzt können wir die Daten herunterladen.

```
# Daten einlesen
data <- read_csv("https://raw.githubusercontent.com/methodenlehre/data/master/statIV_sem/s

# Info zu den Variablen einlesen
variableInfo <- read_csv("https://raw.githubusercontent.com/methodenlehre/data/master/stat
```

Die Jugendlichen haben folgende 14 Items jeweils auf einer Skala von 1-5 gerated:

```
# Variablennamen und Itemwortlaut
print(variableInfo[1:2], n = 14)
```

```
# A tibble: 14 x 2
   variable       label
   <chr>          <chr>
 1 optimistic_1   In uncertain times, I usually expect the best
 2 optimistic_2   I believe in the idea that 'every cloud has a silver lining'
 3 optimistic_3   I always look on the bright side of things
 4 optimistic_4   I'm always optimistic about my future
 5 peer_1         People my age spend their free time with me
 6 peer_2         People my age often share secrets, stories etc. with me
 7 peer_3         People my age are usually friendly to me
 8 peer_4         People my age usually stick up for me
 9 peer_5         People my age like to ask me to hang out with them
10 satisfaction_1 Satisfaction with friendships
11 satisfaction_2 Satisfaction with health
12 satisfaction_3 Satisfaction with school
13 satisfaction_4 Satisfaction with family
14 satisfaction_5 Satisfaction with life as a whole
```

```
# Die Kodierung (Labels der Skalenwerte 1-5) der Items finden Sie in der 3. Spalte
# der Variableninfo (bei Bedarf anschauen mit variableInfo[3]).
# Bei Optimismus und Peer-Akzeptanz handelt es sich um Ablehnung - Zustimmung, bei
# Lebenszufriedenheit um Unzufrieden - Zufrieden.
```

Aufgabe 1

In der Aufgabe geht es erst um ein Messmodell mit drei Faktoren, das alle Items beinhaltet. Nennen Sie die Faktoren `peer` (Peer-Akzeptanz), `optimism` (Optimismus) & `lifesat` (Lebenszufriedenheit). Zunächst ist nur das Messmodell von Interesse, d.h. es spielt noch keine Rolle, welche Effekte wir später (Aufgabe 2) zwischen den latenten Variablen erwarten.

a) Fitten und visualisieren Sie das Modell.

Lösung

Messmodell Definition

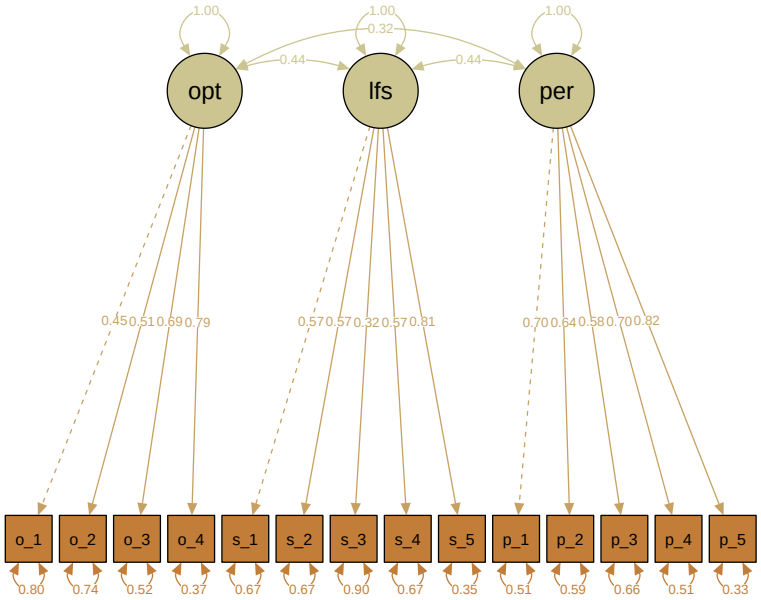
```
mess_modell <- "
optimism =~ optimistic_1 + optimistic_2 + optimistic_3 + optimistic_4
lifesat  =~ satisfaction_1 + satisfaction_2 + satisfaction_3 +
            satisfaction_4 + satisfaction_5
peer     =~ peer_1 + peer_2 + peer_3 + peer_4 + peer_5
"
```

Messmodell schätzen

```
mess_fit <- sem(mess_modell,
  data = data
)
```

Visualisierung des Messmodells

```
semPaths(mess_fit,
  "col",
  "std",
  color = colorlist
)
```



b) Berechnen Sie von Hand die Freiheitsgrade des Modells.

Lösung

Freiheitsgrade des Messmodells
Die Freiheitsgrade lassen sich berechnen als:

$$df = n_{info} - n_{parameter}$$

n_{info} wird wie gewohnt mit folgender Formel berechnet:

$$\begin{aligned} n_{info} &= \frac{p \cdot (p+1)}{2} \\ &= \frac{14 \cdot (14+1)}{2} \\ &= \frac{210}{2} \\ &= 105 \end{aligned}$$

Wir schätzen folgende Parameter:

- Varianzen der latenten Variablen: 3
- Kovarianzen der latenten Variablen:

Einerseits in diesem Fall leicht abzuzählen, dass es zwischen 3 Variablen insgesamt 3 Kovarianzen gibt. Aber wir können es auch mit einer Formel berechnen:

$$
\begin{aligned}
n_{cov\ latent\ variables} &= \frac{n_{lv} \cdot (n_{lv} - 1)}{2} \\
&= \frac{3 \cdot (3 - 1)}{2} \\
&= \frac{6}{2} \\
&= 3
\end{aligned}
$$

> ! Diese Formel stimmt nur, wenn *alle* Kovarianzen der latenten Variablen frei geschätzt werden (was normalerweise der Fall ist).

- Varianzen der Residuen der manifesten Variablen: 14
- Zu schätzende Faktorladungen:

$$
\begin{aligned}
n_{zu\ schtzende\ Ladungen} &= n_{Ladungen} - n_{fixierte\ Ladungen} \\
&= 14 - 3 \\
&= 11
\end{aligned}
$$

Wir haben also insgesamt zu schätzende Parameter:
$n_{parameter} = 3 + 3 + 14 + 11 = 31$
Und damit hat das Modell folgende Freiheitsgrade:
$df = n_{info} - n_{parameter} = 105 - 31 = 74$
Das erhalten wir auch im Modelloutput:

```
summary(mess_fit,
  fit.measures = TRUE,
  std = TRUE
)
```

```
lavaan 0.6.17 ended normally after 39 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        31

                                                  Used       Total
  Number of observations                           289         300

Model Test User Model:

  Test statistic                               148.526
```

```
  Degrees of freedom                                74
  P-value (Chi-square)                           0.000

Model Test Baseline Model:

  Test statistic                              1145.966
  Degrees of freedom                                91
  P-value                                        0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    0.929
  Tucker-Lewis Index (TLI)                       0.913

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -4388.515
  Loglikelihood unrestricted model (H1)      -4314.252

  Akaike (AIC)                                8839.031
  Bayesian (BIC)                              8952.690
  Sample-size adjusted Bayesian (SABIC)       8854.384

Root Mean Square Error of Approximation:

  RMSEA                                          0.059
  90 Percent confidence interval - lower         0.045
  90 Percent confidence interval - upper         0.073
  P-value H_0: RMSEA <= 0.050                    0.136
  P-value H_0: RMSEA >= 0.080                    0.005

Standardized Root Mean Square Residual:

  SRMR                                           0.059

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured
```

```
Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  optimism =~
    optimistic_1      1.000                               0.384    0.449
    optimistic_2      1.045    0.186    5.607    0.000    0.402    0.510
    optimistic_3      1.664    0.260    6.397    0.000    0.640    0.693
    optimistic_4      1.951    0.301    6.478    0.000    0.750    0.794
  lifesat =~
    satisfaction_1    1.000                               0.458    0.574
    satisfaction_2    1.013    0.141    7.205    0.000    0.464    0.571
    satisfaction_3    0.647    0.143    4.538    0.000    0.296    0.319
    satisfaction_4    0.967    0.134    7.205    0.000    0.443    0.571
    satisfaction_5    1.389    0.166    8.371    0.000    0.636    0.808
  peer =~
    peer_1            1.000                               0.585    0.699
    peer_2            0.849    0.088    9.604    0.000    0.497    0.643
    peer_3            0.613    0.070    8.777    0.000    0.359    0.582
    peer_4            0.927    0.090   10.356    0.000    0.543    0.701
    peer_5            1.089    0.094   11.551    0.000    0.637    0.816

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  optimism ~~
    lifesat           0.077    0.019    4.077    0.000    0.440    0.440
    peer              0.072    0.020    3.584    0.000    0.320    0.320
  lifesat ~~
    peer              0.119    0.025    4.811    0.000    0.445    0.445

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .optimistic_1      0.585    0.053   11.106    0.000    0.585    0.798
   .optimistic_2      0.460    0.043   10.730    0.000    0.460    0.740
   .optimistic_3      0.442    0.054    8.211    0.000    0.442    0.519
   .optimistic_4      0.331    0.059    5.626    0.000    0.331    0.370
   .satisfaction_1    0.427    0.041   10.348    0.000    0.427    0.671
   .satisfaction_2    0.445    0.043   10.377    0.000    0.445    0.674
   .satisfaction_3    0.773    0.066   11.654    0.000    0.773    0.898
   .satisfaction_4    0.406    0.039   10.376    0.000    0.406    0.674
   .satisfaction_5    0.215    0.037    5.827    0.000    0.215    0.347
   .peer_1            0.359    0.037    9.786    0.000    0.359    0.512
   .peer_2            0.351    0.034   10.388    0.000    0.351    0.587
```

```
.peer_3          0.250    0.023   10.842    0.000    0.250    0.661
.peer_4          0.305    0.031    9.759    0.000    0.305    0.509
.peer_5          0.204    0.028    7.375    0.000    0.204    0.334
 optimism        0.148    0.042    3.529    0.000    1.000    1.000
 lifesat         0.210    0.045    4.707    0.000    1.000    1.000
 peer            0.343    0.055    6.285    0.000    1.000    1.000
```

c) Wie ist der Model Fit des Modells aufgrund der uns bekannten globalen Fit-Indizes zu beurteilen?

Lösung

Model Fit
Der Model Fit ist nach den inkrementellen Fit-Indizes CFI = 0.929 und TLI = 0.913 als akzeptabel zu beurteilen. Nach dem RMSEA = 0.059 (90% CI [0.045, 0.073]) ist der Fit als gut (nicht signifikant abweichend von 0.05) zu beurteilen.

d) Berechnen Sie den RMR des Modells per Hand.

Lösung

Berechnung des RMR von Hand
Im Output oben wird nur das Standardized Root Mean Square Residual (SRMR) ausgegeben. Dieses zeigt mit 0.059 einen guten Model-Fit an. Den unstandardisierten RMR erhalten wir mit `fitmeasures()`:

```
fitmeasures(mess_fit)
```

```
                npar                 fmin                chisq 
              31.000                0.257              148.526 
                  df               pvalue       baseline.chisq 
              74.000                0.000             1145.966 
         baseline.df      baseline.pvalue                  cfi 
              91.000                0.000                0.929 
                 tli                  nnfi                  rfi 
               0.913                0.913                0.841 
                 nfi                  pnfi                  ifi 
               0.870                0.708                0.930 
                 rni                  logl     unrestricted.logl 
               0.929            -4388.515            -4314.252 
                 aic                   bic                ntotal 
            8839.031             8952.690              289.000 
```

```
              bic2                rmsea        rmsea.ci.lower
          8854.384                0.059                0.045
      rmsea.ci.upper       rmsea.ci.level         rmsea.pvalue
               0.073                0.900                0.136
      rmsea.close.h0 rmsea.notclose.pvalue     rmsea.notclose.h0
               0.050                0.005                0.080
                 rmr           rmr_nomean                 srmr
               0.039                0.039                0.059
        srmr_bentler   srmr_bentler_nomean                 crmr
               0.059                0.059                0.063
         crmr_nomean           srmr_mplus     srmr_mplus_nomean
               0.063                0.059                0.059
               cn_05                cn_01                  gfi
             186.008              205.700                0.933
                agfi                 pgfi                  mfi
               0.905                0.657                0.879
                ecvi
               0.728
```

RMR = 0.039
Für die Berechnung per Hand benötigen wir zuerst die (unstandardisierte) Residual-Varianz-Kovarianz-Matrix:

```
lavInspect(mess_fit, what = "resid")
```

```
$cov
               optm_1 optm_2 optm_3 optm_4 stsf_1 stsf_2 stsf_3 stsf_4 stsf_5
optimistic_1    0.000
optimistic_2    0.031  0.000
optimistic_3   -0.026 -0.015  0.000
optimistic_4   -0.004 -0.005  0.013  0.000
satisfaction_1  0.032 -0.034 -0.032 -0.051  0.000
satisfaction_2  0.090 -0.016 -0.012 -0.016  0.013  0.000
satisfaction_3  0.108  0.037  0.068  0.082 -0.028  0.051  0.000
satisfaction_4 -0.001 -0.012  0.028  0.021 -0.030 -0.006  0.019  0.000
satisfaction_5  0.034  0.044 -0.010 -0.027 -0.002 -0.003 -0.008  0.011  0.000
peer_1          0.042  0.013  0.003  0.002  0.107 -0.011 -0.033 -0.011  0.031
peer_2         -0.034  0.055  0.029 -0.048  0.013 -0.055 -0.121 -0.034 -0.072
peer_3         -0.002  0.033  0.011  0.014  0.051 -0.030 -0.042 -0.028  0.039
peer_4         -0.020  0.015  0.081  0.031  0.084  0.003 -0.079  0.016  0.005
peer_5         -0.054 -0.025 -0.032 -0.019  0.083 -0.022 -0.116 -0.014 -0.014
               peer_1 peer_2 peer_3 peer_4 peer_5
```

```
optimistic_1
optimistic_2
optimistic_3
optimistic_4
satisfaction_1
satisfaction_2
satisfaction_3
satisfaction_4
satisfaction_5
peer_1          0.000
peer_2         -0.009  0.000
peer_3         -0.027  0.012  0.000
peer_4         -0.044  0.002  0.058  0.000
peer_5          0.035  0.010 -0.028 -0.011  0.000
```

Jetzt müssen alle Elemente quadriert und diese dann gemittelt werden, und anschliessend muss die Wurzel gezogen werden. Die Diagonalelemente (alle = 0) müssen in der Summe im Zähler natürlich nicht explizit berücksichtigt werden:

$$
\begin{aligned}
RMR &= \sqrt{\frac{\sum_{i=1}^{p}\sum_{j=1}^{i}\left(s_{ij}-\hat{\sigma}_{ij}\right)^2}{p\cdot(p+1)/2}} \\
&= \sqrt{\frac{0.031^2+(-0.026)^2+(-0.015)^2+...+0.010^2+(-0.028)^2+(-0.011^2)}{105}} \\
&= 0.039
\end{aligned}
$$

Mit so vielen Elementen in der Varianz-Kovarianz-Matrix ist das natürlich äusserst mühsam zu berechnen!

e) Betrachten/interpretieren Sie das Muster der Faktorladungen sowie die geschätzten Kovarianzen/Korrelationen der latenten Variablen.

Lösung

Interpretation der geschätzten Parameter (Fokus auf Faktorladungen und Kovarianzen/Korrelationen der latenten Variablen)

- Die standardisierten Faktorladungen zeigen für die meisten Items relativ hohe Ladungen auf dem zugehörigen Faktor, einige Items laden aber eher niedrig: z.B. die Ladung von 0.449 des Items `optimistic_1` ("In uncertain times, I usually expect the best") auf dem Optimismus-Faktor und besonders die Ladung von 0.319 des Items `satisfaction_3` ("Satisfaction with school") auf dem Lebenszufriedenheits-Faktor.
- Die Korrelationen der latenten Variablen sind substantiell und alle statistisch signi-

fikant. Besonders interessant ist die Korrelation von 0.445 für `lifesat ~~ peer`. Wir werden später sehen, dass diese genau dem (standardisierten) totalen Effekt der UV `peer` auf die AV `lifesat` entspricht.

Aufgabe 2

Jetzt kommt das Strukturmodell dazu: Wir wollen wissen, ob es einen Effekt von Peer-Akzeptanz (Aussmass erfahrener Akzeptanz durch Gleichaltrige) auf Lebenszufriedenheit gibt, und ob dieser Effekt durch Optimismus (generelle positive Einstellung in Bezug auf die Zukunft) mediiert wird.

a) Spezifizieren Sie ein Mediationsmodell mit folgenden Strukturpfaden: Der Faktor `peer` soll einen Effekt auf `optimism` und dieser wiederum einen Effekt auf `lifesat` haben, zudem soll es einen direkten Effekt von `peer` auf `lifesat` geben.

Lösung

Mediationsmodell definieren

```
mediations_modell <- "
# Messmodell
optimism =~ optimistic_1 + optimistic_2 + optimistic_3 + optimistic_4
lifesat  =~ satisfaction_1 + satisfaction_2 + satisfaction_3 +
            satisfaction_4 + satisfaction_5
peer     =~ peer_1 + peer_2 + peer_3 + peer_4 + peer_5

# Strukturmodell
optimism ~ peer
lifesat  ~ peer + optimism
"
```

b) Fitten Sie das Modell und betrachten Sie den Model Fit.

Lösung

Mediationsmodell fitten und ausgeben

```r
med_fit <- sem(mediations_modell,
  data = data
)
summary(med_fit,
  fit.measures = TRUE,
  std = TRUE
)
```

```
lavaan 0.6.17 ended normally after 40 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        31

                                                  Used       Total
  Number of observations                           289         300

Model Test User Model:

  Test statistic                               148.526
  Degrees of freedom                                74
  P-value (Chi-square)                           0.000

Model Test Baseline Model:

  Test statistic                              1145.966
  Degrees of freedom                                91
  P-value                                        0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    0.929
  Tucker-Lewis Index (TLI)                       0.913

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -4388.515
  Loglikelihood unrestricted model (H1)      -4314.252

  Akaike (AIC)                                8839.031
  Bayesian (BIC)                              8952.690
```

```
  Sample-size adjusted Bayesian (SABIC)       8854.384

Root Mean Square Error of Approximation:

  RMSEA                                          0.059
  90 Percent confidence interval - lower         0.045
  90 Percent confidence interval - upper         0.073
  P-value H_0: RMSEA <= 0.050                    0.136
  P-value H_0: RMSEA >= 0.080                    0.005

Standardized Root Mean Square Residual:

  SRMR                                           0.059

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  optimism =~
    optimistic_1      1.000                               0.384    0.449
    optimistic_2      1.045    0.186    5.607    0.000    0.402    0.510
    optimistic_3      1.664    0.260    6.397    0.000    0.640    0.693
    optimistic_4      1.951    0.301    6.478    0.000    0.750    0.794
  lifesat =~
    satisfaction_1    1.000                               0.458    0.574
    satisfaction_2    1.013    0.141    7.205    0.000    0.464    0.571
    satisfaction_3    0.647    0.143    4.538    0.000    0.296    0.319
    satisfaction_4    0.967    0.134    7.205    0.000    0.443    0.571
    satisfaction_5    1.389    0.166    8.371    0.000    0.636    0.808
  peer =~
    peer_1            1.000                               0.585    0.699
    peer_2            0.849    0.088    9.604    0.000    0.497    0.643
    peer_3            0.613    0.070    8.777    0.000    0.359    0.582
    peer_4            0.927    0.090   10.356    0.000    0.543    0.701
    peer_5            1.089    0.094   11.551    0.000    0.637    0.816

Regressions:
```

```
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  optimism ~
    peer              0.210    0.056    3.749    0.000    0.320    0.320
  lifesat ~
    peer              0.265    0.063    4.184    0.000    0.338    0.338
    optimism          0.395    0.109    3.615    0.000    0.331    0.331

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .optimistic_1      0.585    0.053   11.106    0.000    0.585    0.798
   .optimistic_2      0.460    0.043   10.730    0.000    0.460    0.740
   .optimistic_3      0.442    0.054    8.210    0.000    0.442    0.519
   .optimistic_4      0.331    0.059    5.626    0.000    0.331    0.370
   .satisfaction_1    0.427    0.041   10.348    0.000    0.427    0.671
   .satisfaction_2    0.445    0.043   10.377    0.000    0.445    0.674
   .satisfaction_3    0.773    0.066   11.654    0.000    0.773    0.898
   .satisfaction_4    0.406    0.039   10.376    0.000    0.406    0.674
   .satisfaction_5    0.215    0.037    5.827    0.000    0.215    0.347
   .peer_1            0.359    0.037    9.786    0.000    0.359    0.512
   .peer_2            0.351    0.034   10.388    0.000    0.351    0.587
   .peer_3            0.250    0.023   10.842    0.000    0.250    0.661
   .peer_4            0.305    0.031    9.759    0.000    0.305    0.509
   .peer_5            0.204    0.028    7.375    0.000    0.204    0.334
   .optimism          0.133    0.038    3.508    0.000    0.898    0.898
   .lifesat           0.148    0.033    4.532    0.000    0.704    0.704
    peer              0.343    0.055    6.285    0.000    1.000    1.000
```

Model Fit
Genau wie im Vorlesungsbeispiel ist der Model Fit exakt gleich wie im Messmodell! Statt der 3 Varianzen und 3 Kovarianzen der latenten Variablen im Messmodell schätzen wir jetzt 3 Strukturpfade, 1 Varianz der exogenen latenten Variablen `peer` und 2 Residualvarianzen der endogenen latenten Variablen `optimism` und `lifesat`! In beiden Modellen werden also 6 Parameter auf Ebene der latenten Variablen geschätzt. Somit handelt es sich bei unserem Mediationsmodell um ein *saturiertes Strukturmodell*. Das bedeutet, dass sich der Model Fit nur auf den Messmodell-Teil des Modells bezieht (vgl. Vorlesungsbeispiel oben).

c) Welche Pfade im `Strukturmodell` sind signifikant? Welcher Pfad ist vom Betrag her am stärksten?

Lösung

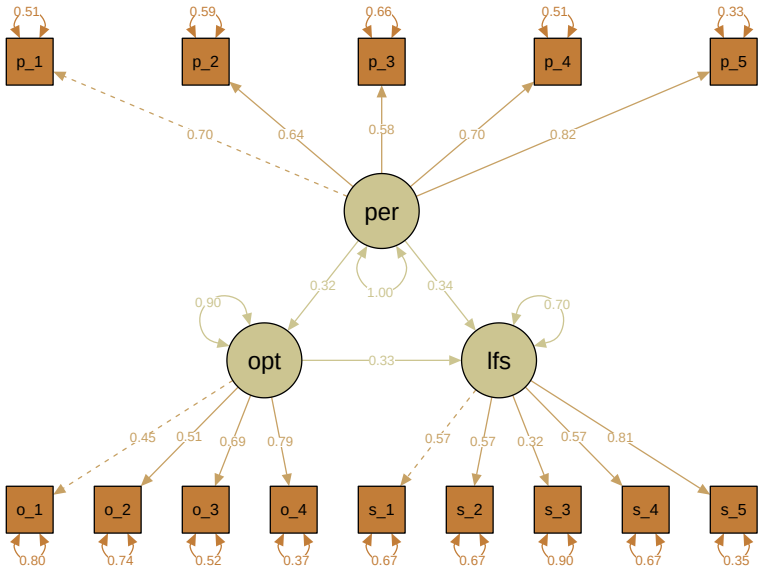
Strukturpfade
Alle Regressionsparameter wurden signifikant mit jeweils $p < 0.001$. Der stärkste Pfad ist der (direkte) Effekt von `peer` auf `lifesat`, $\beta_{peer->lifesat} = 0.3385$, die beiden anderen Effekte sind aber fast genauso gross.

d) Berechnen Sie den geschätzten *indirekten* sowie den geschätzten *totalen* Effekt von `peer` auf `lifesat` zuerst von Hand. Überprüfen Sie anschliessend den Effekt, indem Sie ein neues Modell definieren, das diesen Effekt ausgibt.

Lösung

Indirekter und Totaler Effekt
Zuerst von Hand, am besten mit Hilfe einer Visualisierung:

```
semPaths(med_fit,
  "col",
  "std",
  color = colorlist
)
```



Wir berechnen die Effekte anhand der standardisierten Parameterschätzer. Für eine höhere Genauigkeit lassen wir uns die Effekte nochmal auf 4 Kommastellen gerundet ausgeben

(sonst kommen wir nicht auf das gleiche Ergebnis wie unten im Modell mit definierten Parametern):

```
lavInspect(med_fit, "std")$beta
```

```
          optmsm lifest  peer
optimism  0.000       0 0.320
lifesat   0.331       0 0.338
peer      0.000       0 0.000
```

```
# Runden auf mehr als 3 Kommastellen funktioniert nur so:
as.vector(lavInspect(med_fit, "std")$beta) |> round(4)
```

```
[1] 0.0000 0.3313 0.0000 0.0000 0.0000 0.0000 0.3200 0.3385 0.0000
```

Der *indirekte Effekt* ist definiert als das Produkt der beiden Strukturkoeffizienten von `peer` über `optimism` auf `lifesat`:

$$\begin{aligned}\beta_{indirekt} &= (0.32 * 0.3313) \\ &= 0.106016 \approx 0.106\end{aligned}$$

Der *totale Effekt* ist definiert als der *direkte Effekt* plus der *indirekte Effekt*.

$$\begin{aligned}\beta_{total} &= \beta_{direkt} + \beta_{indirekt} \\ &= 0.3385 + (0.32 * 0.3313) \\ &= 0.3385 + 0.106016 \\ &= 0.444516 \approx 0.445\end{aligned}$$

Und jetzt mit `lavaan`:
Zuerst müssen wir das Modell neu definieren:

```
effekt_modell <- "
# Messmodell
optimism =~ optimistic_1 + optimistic_2 + optimistic_3 + optimistic_4
lifesat  =~ satisfaction_1 + satisfaction_2 + satisfaction_3 +
            satisfaction_4 + satisfaction_5
peer     =~ peer_1 + peer_2 + peer_3 + peer_4 + peer_5

# Strukturmodell
optimism ~ a * peer
lifesat  ~ b * peer + c * optimism

# Effekte
indEff := a * c
totEff := indEff + b
"
```

Jetzt können wir einfach das Modell schätzen und uns die Effekte ausgeben lassen. Beachten Sie, dass das Modell jetzt mit Bootstrap-Standardfehlern geschätzt werden muss:

```
eff_fit <- sem(effekt_modell,
  data = data,
  se = "bootstrap",
  iseed = 123
)
summary(eff_fit,
  fit.measures = TRUE,
  std = TRUE
)
```

```
lavaan 0.6.17 ended normally after 40 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        31

                                                  Used       Total
  Number of observations                           289         300

Model Test User Model:

  Test statistic                               148.526
```

```
  Degrees of freedom                                74
  P-value (Chi-square)                           0.000

Model Test Baseline Model:

  Test statistic                              1145.966
  Degrees of freedom                                91
  P-value                                        0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    0.929
  Tucker-Lewis Index (TLI)                       0.913

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -4388.515
  Loglikelihood unrestricted model (H1)      -4314.252

  Akaike (AIC)                                8839.031
  Bayesian (BIC)                              8952.690
  Sample-size adjusted Bayesian (SABIC)       8854.384

Root Mean Square Error of Approximation:

  RMSEA                                          0.059
  90 Percent confidence interval - lower         0.045
  90 Percent confidence interval - upper         0.073
  P-value H_0: RMSEA <= 0.050                    0.136
  P-value H_0: RMSEA >= 0.080                    0.005

Standardized Root Mean Square Residual:

  SRMR                                           0.059

Parameter Estimates:

  Standard errors                            Bootstrap
  Number of requested bootstrap draws             1000
  Number of successful bootstrap draws            1000
```

```
Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  optimism =~
    optimistic_1      1.000                               0.384    0.449
    optimistic_2      1.045    0.240    4.359    0.000    0.402    0.510
    optimistic_3      1.664    0.357    4.663    0.000    0.640    0.693
    optimistic_4      1.951    0.408    4.779    0.000    0.750    0.794
  lifesat =~
    satisfaction_1    1.000                               0.458    0.574
    satisfaction_2    1.013    0.163    6.217    0.000    0.464    0.571
    satisfaction_3    0.647    0.188    3.435    0.001    0.296    0.319
    satisfaction_4    0.967    0.165    5.853    0.000    0.443    0.571
    satisfaction_5    1.389    0.242    5.738    0.000    0.636    0.808
  peer =~
    peer_1            1.000                               0.585    0.699
    peer_2            0.849    0.110    7.713    0.000    0.497    0.643
    peer_3            0.613    0.111    5.518    0.000    0.359    0.582
    peer_4            0.927    0.133    6.994    0.000    0.543    0.701
    peer_5            1.089    0.090   12.159    0.000    0.637    0.816

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  optimism ~
    peer       (a)    0.210    0.065    3.225    0.001    0.320    0.320
  lifesat ~
    peer       (b)    0.265    0.078    3.412    0.001    0.338    0.338
    optimism   (c)    0.395    0.115    3.437    0.001    0.331    0.331

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .optimistic_1      0.585    0.068    8.589    0.000    0.585    0.798
   .optimistic_2      0.460    0.045   10.142    0.000    0.460    0.740
   .optimistic_3      0.442    0.053    8.305    0.000    0.442    0.519
   .optimistic_4      0.331    0.070    4.756    0.000    0.331    0.370
   .satisfaction_1    0.427    0.057    7.460    0.000    0.427    0.671
   .satisfaction_2    0.445    0.046    9.782    0.000    0.445    0.674
   .satisfaction_3    0.773    0.076   10.223    0.000    0.773    0.898
   .satisfaction_4    0.406    0.047    8.723    0.000    0.406    0.674
   .satisfaction_5    0.215    0.040    5.416    0.000    0.215    0.347
   .peer_1            0.359    0.070    5.146    0.000    0.359    0.512
   .peer_2            0.351    0.042    8.366    0.000    0.351    0.587
```

```
   .peer_3           0.250    0.029    8.608    0.000    0.250    0.661
   .peer_4           0.305    0.044    6.905    0.000    0.305    0.509
   .peer_5           0.204    0.035    5.862    0.000    0.204    0.334
   .optimism         0.133    0.044    3.002    0.003    0.898    0.898
   .lifesat          0.148    0.050    2.927    0.003    0.704    0.704
    peer             0.343    0.076    4.532    0.000    1.000    1.000

Defined Parameters:
                  Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
    indEff           0.083    0.030    2.752    0.006    0.106    0.106
    totEff           0.348    0.082    4.259    0.000    0.445    0.445
```

Beide Effekte entsprechen den von Hand gerechneten Effekten. Sowohl der **indirekte Effekt** als auch der **totale Effekt** wurde signifikant, `totEff` $= 0.445, p < 0.001$; `indEff` $= 0.106, p = 0.006$. Der signifikante indirekte Effekt bedeutet, dass eine Mediation gegeben ist. Da aber auch der direkte Effekt von `peer` auf `lifesat` signifikant ist, spricht man hier von einer *partiellen Mediation*, d.h. ein Teil des (totalen) Effekts von Peer-Akzeptanz auf Lebenszufriedenheit wird über Optimismus vermittelt. In normalem Deutsch könnte man das so formulieren:
"Jugendliche, die sich von ihren Peers akzeptiert fühlen, sind mit ihrem Leben zufriedener. Ein Teil dieser höheren Zufriedenheit kommt dadurch zustande (kann so erklärt werden), dass sich-akzeptiert-fühlende Jugendliche mit grösserem Optimismus in die Zukunft blicken und dieser Optimismus wiederum mit höherer Lebenszufriedenheit einhergeht."

Aufgabe 3

Berechnen Sie den BIC des Mediationsmodells.

Lösung

BIC

$$BIC = \chi^2 + ln(n) \cdot t$$

Um den BIC zu berechnen benötigen wir also drei Werte:

- Den χ^2-Wert des Modells ($\chi^2 = 148.526$)
- Die Anzahl geschätzter Parameter ($t = n_{par} = 31$)
- Die Anzahl Teilnehmer ($n = 289$)

Diese Werte finden wir alle im Modelloutput oder direkt bei den `fitmeasures` des Modells. Der Grund für die kleinere Anzahl Teilnehmer (`ntotal = 289`) als die oben angegebenen

$n = 300$ ist übrigens die automatische Entfernung von Fällen mit fehlenden Werten in `lavaan`.

```
fitmeasures(med_fit)
```

```
                npar                 fmin                chisq 
              31.000                0.257              148.526 
                  df               pvalue       baseline.chisq 
              74.000                0.000             1145.966 
         baseline.df      baseline.pvalue                  cfi 
              91.000                0.000                0.929 
                 tli                 nnfi                  rfi 
               0.913                0.913                0.841 
                 nfi                 pnfi                  ifi 
               0.870                0.708                0.930 
                 rni                 logl    unrestricted.logl 
               0.929            -4388.515            -4314.252 
                 aic                  bic               ntotal 
            8839.031             8952.690              289.000 
                bic2                rmsea       rmsea.ci.lower 
            8854.384                0.059                0.045 
      rmsea.ci.upper       rmsea.ci.level         rmsea.pvalue 
               0.073                0.900                0.136 
      rmsea.close.h0 rmsea.notclose.pvalue     rmsea.notclose.h0 
               0.050                0.005                0.080 
                 rmr            rmr_nomean                 srmr 
               0.039                0.039                0.059 
        srmr_bentler   srmr_bentler_nomean                 crmr 
               0.059                0.059                0.063 
         crmr_nomean            srmr_mplus     srmr_mplus_nomean 
               0.063                0.059                0.059 
              cn_05                cn_01                  gfi 
            186.008              205.700                0.933 
                agfi                 pgfi                  mfi 
               0.905                0.657                0.879 
                ecvi                                            
               0.728                                            
```

Das führt zu der Lösung:

$$
\begin{aligned}
BIC &= \chi^2 + ln(n) \cdot t \\
&= 148.526 + ln(289) \cdot 31 \\
&= 324.185
\end{aligned}
$$

Wie wir schon aus dem Kapitel zur CFA wissen, unterscheidet sich der so berechnete BIC von dem von `lavaan` ausgegebenen (8952.690). Letzterer wird auf Basis der $-2 \cdot ln(L)$ und nicht auf Basis von χ^2 berechnet. Das spielt aber keine Rolle, da der BIC zum Vergleich *verschiedener Modelle* dient (was wir hier nicht tun), und die Differenz der BICs verschiedener Modelle unabhängig von der Berechnungsmethode dieselbe ist.

Aufgabe 4

Lassen Sie sich die *Modifikations-Indizes* ausgeben (nur solche >= 5). Interpretieren Sie den Output.

Lösung

```
modindices(med_fit, minimum.value = 5)
```

```
               lhs op            rhs     mi    epc sepc.lv sepc.all sepc.nox
50         lifesat =~         peer_2  8.536 -0.319  -0.146   -0.189   -0.189
58            peer =~ satisfaction_1 14.737  0.356   0.209    0.261    0.261
60            peer =~ satisfaction_3  9.338 -0.353  -0.206   -0.223   -0.223
67    optimistic_1 ~~ satisfaction_2  5.910  0.079   0.079    0.155    0.155
82    optimistic_2 ~~ satisfaction_5  6.081  0.062   0.062    0.196    0.196
84    optimistic_2 ~~         peer_2  6.246  0.065   0.065    0.163    0.163
97    optimistic_3 ~~         peer_4  6.081  0.065   0.065    0.178    0.178
141 satisfaction_5 ~~         peer_3  7.304  0.050   0.050    0.214    0.214
146         peer_1 ~~         peer_4  9.474 -0.082  -0.082   -0.248   -0.248
147         peer_1 ~~         peer_5 15.499  0.112   0.112    0.414    0.414
151         peer_3 ~~         peer_4 18.874  0.087   0.087    0.313    0.313
152         peer_3 ~~         peer_5 10.126 -0.063  -0.063   -0.281   -0.281
```

Alle Modifikationsindizes beziehen sich auf das Messmodell. Das Strukturmodell ist ja wie wir gesehen haben bereits saturiert, daher können dort keine zusätzlichen Parameter geschätzt werden. Grundsätzlich sollten im Messmodell keine zusätzlichen Parameter wie Querladungen und Residualkovarianzen manifester Variablen zugelassen werden, da diese gegen die Einfachstruktur-Hypothese der Messmodell-CFA sprechen. Allerdings können wir natürlich schauen, wo im Messmodell Fit-Probleme existieren, um das Modell besser zu verstehen.

Der höchste Modifikationsindex ist hier der für eine mögliche Residualkovarianz zwischen den Items `peer_3` und `peer_4` (mi = 18.874). Diese wäre positiv (epc = 0.087), d.h. diese beiden Items korrelieren stärker miteinander als es durch den gemeinsamen Faktor `peer` abgebildet werden kann.
Besonders gut interpretierbar ist zudem ein weiterer Modifikationsindex: eine mögliche Querladung des Items `satisfaction_1` ("Satisfaction with friendships") auf dem `peer`-Faktor. Wenn wir diese Querladung zulassen würden, würden wir eine (unstandardisierte) Ladung von 0.356 erhalten (epc = expected parameter change) und das χ^2 des Modells wäre um 14.737 Punkte niedriger. Diese Werte sind allerdings approximative Schätzungen, da für die Berechnung der Modifikationsindizes die verschiedenen Alternativmodelle (mit je einem zusätzlichen Parameter) nicht explizit geschätzt werden.
Zu Demonstrationszwecken können wir das Modell mit der Querladung schätzen:

```
mediations_modell_crossload <- "
# Messmodell
optimism =~ optimistic_1 + optimistic_2 + optimistic_3 + optimistic_4
lifesat  =~ satisfaction_1 + satisfaction_2 + satisfaction_3 +
            satisfaction_4 + satisfaction_5
peer     =~ peer_1 + peer_2 + peer_3 + peer_4 + peer_5 + satisfaction_1

# Strukturmodell
optimism ~ peer
lifesat  ~ peer + optimism
"
med_fit_crossload <- sem(mediations_modell_crossload, data = data)
summary(med_fit_crossload)
```

```
lavaan 0.6.17 ended normally after 45 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        32

                                                  Used       Total
  Number of observations                           289         300

Model Test User Model:

  Test statistic                               134.306
  Degrees of freedom                                73
  P-value (Chi-square)                           0.000
```

```
Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  optimism =~
    optimistic_1      1.000
    optimistic_2      1.046    0.186    5.611    0.000
    optimistic_3      1.662    0.260    6.400    0.000
    optimistic_4      1.949    0.301    6.481    0.000
  lifesat =~
    satisfaction_1    1.000
    satisfaction_2    1.298    0.223    5.818    0.000
    satisfaction_3    0.854    0.203    4.213    0.000
    satisfaction_4    1.247    0.214    5.835    0.000
    satisfaction_5    1.805    0.290    6.220    0.000
  peer =~
    peer_1            1.000
    peer_2            0.844    0.088    9.620    0.000
    peer_3            0.609    0.069    8.789    0.000
    peer_4            0.924    0.089   10.407    0.000
    peer_5            1.088    0.093   11.656    0.000
    satisfaction_1    0.327    0.087    3.743    0.000

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  optimism ~
    peer              0.205    0.056    3.698    0.000
  lifesat ~
    peer              0.174    0.052    3.330    0.001
    optimism          0.321    0.094    3.420    0.001

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .optimistic_1      0.585    0.053   11.105    0.000
   .optimistic_2      0.460    0.043   10.727    0.000
   .optimistic_3      0.442    0.054    8.215    0.000
   .optimistic_4      0.331    0.059    5.635    0.000
```

```
   .satisfaction_1    0.417    0.039   10.762    0.000
   .satisfaction_2    0.445    0.043   10.306    0.000
   .satisfaction_3    0.767    0.066   11.623    0.000
   .satisfaction_4    0.403    0.039   10.267    0.000
   .satisfaction_5    0.202    0.039    5.150    0.000
   .peer_1            0.357    0.036    9.800    0.000
   .peer_2            0.352    0.034   10.435    0.000
   .peer_3            0.251    0.023   10.872    0.000
   .peer_4            0.306    0.031    9.806    0.000
   .peer_5            0.202    0.027    7.403    0.000
   .optimism          0.133    0.038    3.512    0.000
   .lifesat           0.094    0.028    3.364    0.001
    peer              0.345    0.055    6.321    0.000
```

Die Querladung wird mit 0.327 geschätzt und $\chi^2 = 134.306$. Der Vergleich mit dem χ^2 des Ausgangsmodells (148.526) zeigt eine Differenz von 14.220 (vgl. mi = 14.737).

Bates, Douglas M. 2010. *Lme4: Mixed-Effects Modeling with R*. Unpublished.

Belenky, Gregory, Nancy J. Wesensten, David R. Thorne, Maria L. Thomas, Helen C. Sing, Daniel P. Redmond, Michael B. Russo, und Thomas J. Balkin. 2003. „Patterns of Performance Degradation and Restoration during Sleep Restriction and Subsequent Recovery: A Sleep Dose-Response Study“. *Journal of Sleep Research* 12 (1): 1–12. https://doi.org/10.1046/j.1365-2869.2003.00337.x.

Eid, Michael, Mario Gollwitzer, und Manfred Schmitt. 2017. *Statistik und Forschungsmethoden: mit Online-Materialien*. 5., korrigierte Auflage. Weinheim Basel: Beltz.

Gelman, Andrew, und Jennifer Hill. 2007. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press.