Workshop LEO 2018: Einführung in die Datennutzung und Sekundäranalysen

02.12.2022

Inhaltsverzeichnis

Liebe Workshop-Teilnehmende,

wir freuen uns sehr über Ihr Interesse an LEO 2018 und über Ihre Anmeldung zum Workshop am 02.12.2022. In Vorbereitung auf den Workshop haben wir dieses Dokument erstellt, das zum einen den Einstieg in die Datenauswertung erleichtern soll, zum anderen gewährleisten soll, dass zu Beginn des Workshops alle technischen Voraussetzungen erfüllt sind. Dies umfasst:

- Download der LEO 2018-Daten über das GESIS-Datenarchiv (https://search.gesis.org/research_data/ZA6266)
- Installation von R und RStudio
- Installation benötigter Pakete in RStudio
- Test der erfolgreichen Installation von R und RStudio.

In Abschnitt 2 und 3 wird erläutert, wie Sie vorgehen müssen, um am 02.12. startklar zu sein. **Wir möchten Sie daher bitten, vor dem Workshop diese Anleitung bis einschließlich des Funktionstests in Abschnitt 3 zu lesen, sich die verlinkten Videos anzusehen und wie beschrieben vorzugehen.** Sollten Sie dabei auf Probleme stoßen, stehen wir gerne per Mail (gregor.dutz@uni-hamburg.de oder karola.cafantaris@uni-hamburg.de) und nach Absprache auch über Zoom zur Verfügung.

Sie können darüber hinaus auch gerne den Rest des Dokuments durcharbeiten. Dies ist nicht unbedingt erforderlich, ist aber insbesondere dann sinnvoll, wenn Sie bisher noch nie mit R gearbeitet haben. Darüber hinaus können Sie auch gerne bereits einen Blick in die Dokumentation des Datensatzes werfen, um sich mit der thematischen Breite des Fragebogens von LEO 2018 vertraut zu machen. Die Dokumentation erhalten Sie ebenfalls über das GESIS-Datenarchiv.

Der Workshop wird online über Zoom stattfinden. Mit dem folgenden Link können Sie dem Zoom-Raum beitreten: https://uni-hamburg.zoom.us/j/66418901647?pwd=ZjNMUTVaVVlNNmFmY2p5bXlDZnlhZz09. Der Passcode lautet 32820659.

Über die Veranstaltungsseite haben Sie Zugriff auf alle Materialien und auch den Ergebnisband von LEO 2018. Sie können dort auch Ihre Anmeldung verwalten. Sollten Sie Ihre Anmeldedaten nicht mehr zur Hand haben, finden Sie einen Link in der Mail, mit denen Ihnen dieses Dokument zugekommen ist.

Wir freuen uns auf einen produktiven und hoffentlich für Sie informativen Workshop am 02.12.2022! Bei Fragen im Vorweg melden Sie sich gerne bei uns.

Mit besten Grüßen das LEO-Team (Dr. Klaus Buddeberg, Dr. Karola Cafantaris, Gregor Dutz, Prof. Dr. Anke Grotlüschen, Kristin Skowranek)

1 Einleitung

Dieses Skript ist eine kurze Einführung in die Durchführung von Analysen mit dem Datensatz von LEO 2018 mit der Statistiksoftware R[1] und RStudio. R und RStudio sind kostenfrei verfügbare Software, die für alle gängigen Betriebssysteme zur Verfügung steht. RStudio ist eine Entwicklungsumgebung, die die Bedienung von R erheblich komfortabler gestaltet.

Um auch mit dieser Software Unerfahrenen die Nutzung der LEO-Daten zu ermöglichen, beginnt dieses Skript mit einer kurzen Einführung in die Bedienung von R/RStudio und einigen Hinweisen zur Datenaufbereitung, bevor auf die Datenanalyse eingegangen wird. Dies ist jedoch keine vollumfängliche Einführung in R/RStudio oder statistische Methoden.

Aus der großen Zahl von Methodeneinführungen soll hier auf zwei Werke verwiesen werden, die bei methodischen und statistischen Fragen, die im Verlauf dieses Skripts entstehen könnten, weiterhelfen können:

- Diekmann, A. (2007). Empirische Sozialforschung: Grundlagen, Methoden, Anwendungen. Reinbek: Rowohlt.
- Wolf, C., & Best, H. (Hrsg.). (2010). Handbuch der sozialwissenschaftlichen Datenanalyse. Wiesbaden: VS Verlag für Sozialwissenschaften.

[1] Unter R versteht man das Programm **und** die Programmiersprache, die für die Durchführung statistischer Berechnungen entwickelt wurden.

Als deutschsprachige Einführung in R sei folgendes Buch empfohlen, dass in der englischsprachigen Ausgabe von den Autor*innen kostenlos zur Verfügung gestellt wird:

- Wickham, H., & Grolemund, G. (2018): R für Data Science: Daten importieren, bereinigen, umformen, modellieren und visualisieren. Heidelberg: O'Reilly
- Wickham, H., & Grolemund, G.: R for Data Science

Außerdem lässt sich in R über die Konsole und ein `?` die Hilfe zu einem Befehl anzeigen. Der Befehl `?mean` zeigt so die Hilfe zu der Funktion `mean()` an, die den Mittelwert berechnet.

An einigen Stellen sind Videos verlinkt, um Sachverhalte zu ergänzen oder ausführlicher zu erklären. Die Videos werden über die Medienplattform Lecture2Go der Universität Hamburg zugänglich gemacht.

Das nachfolgend verlinkte Video ist eine Kurzeinführung in die Datennutzung von LEO 2018 und bietet auch eine knappe Einführung in die Studie selbst:

Video: LEO 2018: Kurzeinführung in die Datennutzung

---

2 Voraussetzungen

2.1 Datenzugang

Die Daten von LEO 2018 sind über das GESIS Datenarchiv beziehbar. Für den Public-Use-File (PUF) ist lediglich eine Anmeldung erforderlich, das Scientific-Use-File (SUF) erfordert eine Beantragung und Genehmigung durch die Projektverantwortlichen. Für den Workshop reich der PUF aus.

Bitte melden Sie sich bei GESIS an und laden Sie sich den Datensatz in der PUF-Version herunter. In dem Datenarchiv finden sich auch der Fragebogen und das Codebook.

2.2 Software

Die notwendige Software ist für alle gängigen Betriebssysteme kostenlos verfügbar:

- R: https://cloud.r-project.org/
- RStudio Desktop: https://posit.co/download/rstudio-desktop/

R muss mindestens in der **Version 4.1** installiert sein, RStudio mindestens in der **Version 1.4**.

Sollte R oder RStudio bereits in einer älteren Version installiert sein, können die neueren Versionen ebenfalls über die Links oben heruntergeladen und einfach installiert werden[2].

Haben Sie R und RStudio installiert, können Sie RStudio nun starten.

Video: LEO 2018: Erste Schritte in RStudio

2.3 R-Pakete

Für die Nutzung des LEO-Datensatze sind einige R-Pakete[3] notwendig, die wie folgt in RStudio installiert werden können:

```
# Längere Befehle können zur besseren Lesbarkeit auf mehrere Zeilen umgebrochen werden
install.packages(pkgs = c("devtools", "tidyverse", "survey", "labelled", "mitools",
                          "questionr", "writexl", "mitml"))
```

[2] Im Fall von R wird die neuere Version **zusätzlich** zu der älteren Version installiert, RStudio sollte jedoch automatisch die neuere Version nutzen.

[3] Pakete erweitern den Funktionsumfang von R. Sie werden von Nutzer*innen erstellt und häufig kostenlos und als Open Source bereitgestellt. Das größte Pakete-Archiv ist das CRAN (The Comprehensive R Archive Network).

Dieser Befehl muss nur ein Mal ausgeführt werden und kann einfach in die Console (standardmäßig im unteren linken Bereich der Benutzeroberfläche von RStudio) kopiert werden. Drücken Sie anschließend die Enter-Taste. RStudio lädt die benötigten Dateien herunter und installiert automatisch die Pakete.

Hinweis: Eventuell erhalten Sie hier eine Warnung, dass RTools nicht installiert sei. Sie können sich RTools unter dem in der Warnung angegebenen Link herunterladen und installieren. Die untenstehenden Beispiele sollten aber auch ohne RTools funktionieren.

Außerdem wird das Paket leo benötigt, welches den Umgang mit den LEO-Daten in R vereinfacht. Die dafür benötigte Datei `leo_0.2.8.zip` ist auf der Veranstaltungsseite unten rechts verfügbar (https://www.conferences.uni-hamburg.de/event/280/). Bitte laden Sie diese Datei herunter. Anschließend klicken Sie in RStudio in der Menuleiste auf "`Tools -> Install Packages ...`". In dem nun geöffneten Fenster wählen Sie unter "`Install from:`" bitte "`Package Archive File`" aus. Wählen Sie nun die eben heruntergeladene Datei mit dem Namen `leo_0.2.8.zip` aus und klicken Sie danach auf "`Install`".

Hinweis: Sollte die Installation des leo-Pakets bei Ihnen nicht funktionieren, gibt es einen alternativen Weg: Laden Sie sich dazu bitte die Datei unter dem folgenden Link herunter: https://www.conferences.uni-hamburg.de/event/280/attachments/395/645/leo_svydesign.R. Speichern Sie diese Datei im Ordner des Projekts, dass Sie zuvor mit RStudio angelegt haben.

In der Sytax zum Funktionstest (siehe nächster Abschnitt) entfernen Sie dann die Zeile `library(leo)` und ersetzen Sie diese mit `source("leo_svydesign.R")`. Sollte es beim Ausführen des neuen Befehls zu einem Fehler kommen, überprüfen Sie bitte, ob die Datei `leo_svydesign.R` im Projektordner vorhanden ist.

Der Funktionstest sollte nun funktionieren. Im weiteren Verlauf und bei weiteren Beispielen ersetzen Sie jeweils `library(leo)` mit `source("leo_svydesign.R")`.

---

3 Funktionstest

Um zu testen, ob die Installation von R, RStudio und den benötigten Paketen erfolgreich war kopieren Sie bitte die untenstehende Syntax in RStudio. Das verlinkte Video zeigt, wie dies funktioniert. Bitte passen Sie in der Syntax noch den Datenpfad zu den LEO-Daten an.

Unter der Syntax finden Sie auch den korrekten Output von R, den Sie mit Ihrem Output vergleichen können, um den korrekten Ablauf des Codes zu überprüfen. Auch im weiteren Verlauf dieses Dokuments finden Sie immer die notwendige Syntax und darunter zur Überprüfung den korrekten Output.

Video: LEO 2018: Funktionstest

```
library(leo)
library(tidyverse)
library(haven)
library(labelled)
library(survey)
library(mitools)
library(questionr)
library(writexl)

# Datensatz liegt hier im selben Ordner wie das Script, ggf. bitte den Pfad anpassen
# Statt eines Backslash (\) bitte immer normale Schrägstriche (/) verwenden.
datenpfad <- "ZA6266_v1-0-0.sav"
```

```r
df <- read_sav(file = datenpfad)

df <- df |> select(pgewges,
                   contains("pv"),
                   f001, altgr5,
                   schulab, zuf001)

describe(x = df$f001)

df$f001 <- to_factor(x = df$f001)

df.sd <- leo_svydesign(data = df)

MIcombine(with(df.sd, svytotal(~f001, na.rm = TRUE)))

MIcombine(with(df.sd, svymean(~pv, na.rm = TRUE)))
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## Lade nötiges Paket: grid
##
## Lade nötiges Paket: Matrix
##
##
## Attache Paket: 'Matrix'
##
##
## Die folgenden Objekte sind maskiert von 'package:tidyr':
##
##     expand, pack, unpack
##
##
## Lade nötiges Paket: survival
##
##
## Attache Paket: 'survey'
##
##
## Das folgende Objekt ist maskiert 'package:graphics':
##
##     dotchart
```

```
## [7192 obs.] Geschlecht
## labelled double: 2 2 2 1 2 1 1 2 2 1 ...
## min: 1 - max: 2 - NAs: 0 (0%) - 2 unique values
## 2 value labels: [1] männlich [2] weiblich
##
##                  n     %
## [1] männlich  3129  43.5
## [2] weiblich  4063  56.5
## Total         7192 100.0
```

```
## Multiple imputation results:
##       with(df.sd, svytotal(~f001, na.rm = TRUE))
##       MIcombine.default(with(df.sd, svytotal(~f001, na.rm = TRUE)))
##                 results       se
## f001männlich 3646.487 78.12358
## f001weiblich 3545.355 65.65154
```

```
## Multiple imputation results:
##       with(df.sd, svymean(~pv, na.rm = TRUE))
##       MIcombine.default(with(df.sd, svymean(~pv, na.rm = TRUE)))
##     results       se
## pv 51.90409 0.199654
```

Sie sind nun startklar für den Workshop! Gerne können Sie aber auch schon ein bisschen "vorarbeiten" und den Rest dieses Dokuments durchsehen. Falls Sie noch nie mit R gearbeitet haben, raten wir Ihnen dazu auch.

4 Vorgehen bei der Datenanalyse

4.1 Pakete laden

Die installierten Pakete stellen verschiedene Funktionen bereit, die den Funktionsumfang von R erweitern und für die Datenauswertung notwendig sind. Das Laden der Pakete ist nur zu Beginn einer Sitzung notwendig; es schadet jedoch auch nicht, wenn die Pakete bei jedem Ausführen der Syntax neu geladen werden. Erst nach dem Laden können Funktionen aus den Paketen genutzt werden.

```
library(leo)
library(tidyverse)
library(haven)
library(labelled)
library(survey)
library(mitools)
library(questionr)
library(writexl)
```

4.2 Daten einlesen

Die Daten im SPSS-Format können mit dem Paket `haven` eingelesen werden. Unter Windows muss im Dateipfad jeder Backslash mit einem normalen Schrägstrich ersetzt werden.

Hinweis: Ein vorangestelltes `#` kennzeichnet einen Kommentar. Dahinter folgender Text oder Befehle werden nicht ausgeführt.

```
# Datensatz liegt hier im selben Ordner wie das Script
datenpfad <- "ZA6266_v1-0-0.sav"
# Einlesen des Datensatzes und speichern im Objekt `df`
df <- read_sav(file = datenpfad)
```

Bei Auswertungen der so importierten Daten sollte die Gewichtungsvariable `pgewges` genutzt werden. Bei Berechnungen mit den Ergebnissen des Literalitätstests muss außerdem bedacht werden, das die Personenfähigkeiten als zehn sogenannte plausible Werte (PVs) vorliegen.

4.3 Variablenauswahl und Datenaufbereitung

Zunächst ist es jedoch sinnvoll, nur mit den für die Auswertungen benötigten Variablen weiterzuarbeiten, diese zu überprüfen und gegebenenfalls zu bearbeiten.

4.3.1 Variablen auswählen

Variablen können über den Variablennamen (siehe Codebook) mit der Funktion `select` ausgewählt werden. Dabei müssen mindestens eine Gewichtungsvariable und die Variablen mit den Personenfähigkeiten

(plausible Werte, PVs) enthalten sein. Die folgende Syntax enthält außerdem beispielhaft die Variablen `f001` (Gender), `altgr5` (Alter in fünf Gruppen), `schulab` (Schulabschluss) und `zuf001` (Allgemeine Lebenszufriedenheit auf einer Skala von 0-10).

```
df <- df |> select(pgewges,          # Gewichtungsvariable
                   contains("pv"),   # wählt alle PV-Variablen aus
                   f001, altgr5,     # Gender, Alter
                   schulab, zuf001) # Schulabschluss, Zufriedenheit
```

Die Funktion `describe` aus dem Paket `questionr` eignet sich gut um die Ausprägungen der vorher importierten Variablen zu betrachten. Dies bietet sich insbesondere bei kategorialen Variablen an, die in LEO 2018 die häufigste Art von Variablen sind. Die angezeigten Häufigkeiten beziehen sich hier immer auf die **ungewichteten** Fallzahlen. Bei Variablen mit mehr als zehn Ausprägungen muss mit dem Argument `freq.n.max` die Zahl der angezeigten Ausprägungen erhöht werden.

```
describe(x = df$f001)
```

```
## [7192 obs.] Geschlecht
## labelled double: 2 2 2 1 2 1 1 2 2 1 ...
## min: 1 - max: 2 - NAs: 0 (0%) - 2 unique values
## 2 value labels: [1] männlich [2] weiblich
##
##                  n     %
## [1] männlich  3129  43.5
## [2] weiblich  4063  56.5
## Total         7192 100.0
```

```
describe(x = df$altgr5)
```

```
## [7192 obs.] Altersgruppen (5 Gruppen)
## labelled double: 5 4 5 4 4 4 3 5 3 3 ...
## min: 1 - max: 5 - NAs: 0 (0%) - 5 unique values
## 5 value labels: [1] 18-24 Jahre [2] 25-34 Jahre [3] 35-44 Jahre [4] 45-54 Jahre [5] 55-64 Jahre
##
##                     n     %
## [1] 18-24 Jahre  1091  15.2
## [2] 25-34 Jahre  1320  18.4
## [3] 35-44 Jahre  1217  16.9
## [4] 45-54 Jahre  1566  21.8
## [5] 55-64 Jahre  1998  27.8
## Total            7192 100.0
```

```
describe(x = df$schulab)
```

```
## [7192 obs.] höchster Schulabschluss
## labelled double: 2 2 2 3 2 2 3 3 1 1 ...
## min: 0 - max: 9 - NAs: 0 (0%) - 6 unique values
## 6 value labels: [0] kein Abschluss [1] niedrig [2] mittel [3] hoch [4] derzeit Schulbesuch [9] ke
##
##                              n     %
## [0] kein Abschluss         366   5.1
## [1] niedrig               1738  24.2
## [2] mittel                2176  30.3
## [3] hoch                  2756  38.3
## [4] derzeit Schulbesuch    143   2.0
## [9] keine Angabe            13   0.2
## Total                     7192 100.0
```

```
describe(x = df$zuf001, freq.n.max = 100)
```

```
## [7192 obs.] Allgemeine Lebenszufriedenheit
## labelled double: 8 9 7 3 5 4 9 10 8 4 ...
```

```
## min: 0 - max: 99 - NAs: 0 (0%) - 12 unique values
## 12 value labels: [0] 0 - bin völlig unzufrieden [1] 1 [2] 2 [3] 3 [4] 4 [5] 5 [6] 6 [7] 7 [8] 8 [
##
##                                                        n     %
## [0] 0 - bin völlig unzufrieden                        39   0.5
## [1] 1                                                 16   0.2
## [2] 2                                                 41   0.6
## [3] 3                                                 79   1.1
## [4] 4                                                123   1.7
## [5] 5                                                443   6.2
## [6] 6                                                457   6.4
## [7] 7                                                886  12.3
## [8] 8                                               2014  28.0
## [9] 9                                               1447  20.1
## [10] 10 - bin voll und ganz mit meinem Leben zufrieden 1634  22.7
## [99] Keine Angabe                                     13   0.2
## Total                                               7192 100.0
```

Video: LEO 2018: Blick in den Datensatz und die plausiblen Werte

---

4.3.2 Datenaufbereitung

Häufig müssen die Variablen noch aufbereitet werden, um anschließend sinnvolle Analysen durchführen zu können. Dazu zählen das Recodieren bestimmter Werte und das Setzen einer anderen Referenzkategorie. Da die importierten Daten standardmäßig als kontinuierliche Daten vorliegen, müssen kategoriale Variablen auch in Faktorvariablen umgewandelt werden.

4.3.3 Umwandeln in Faktorvariablen

R behandelt kategoriale Variablen im importierten Datensatz standardmäßig wie kontinuierliche Variablen. Im einfachsten Fall der Datenaufbereitung müssen kategoriale Variablen also zumindest in sog. Faktorvariablen umgewandelt werden. Dies ist z.B. bei der Gender-Variablen der Fall:

```
# Umwandeln in Faktorvariable
df$f001 <- to_factor(x = df$f001)
```

Bei der Umwandlung in eine Faktorvariable gehen in R die Informationen über die ursprüngliche Codierung im Datensatz verloren. Als Ausweg aus dieser Problematik bietet sich an eine neue Variable zu erstellen, statt die bereits im Datensatz vorhandene Variable zu überschreiben, also etwa:

```
# Erstellung einer neuen Faktorvariablen
df$f001_faktor <- to_factor(x = df$f001)
```

Meist wird die Information über die ursprünglichen Ausprägungen aber nicht weiter benötigt. Auf dieses Vorgehen wird hier daher verzichtet.

4.3.4 Recodieren und fehlende Werte setzen

Ein Blick in die Variable Schulabschluss (`schulab`) zeigt, dass die Variable eine Kategorie `keine Angabe` enthält, die hier in den weiteren Auswertungen nicht interessiert. Um solche Fälle zu entfernen, sollen diese Werte auf fehlend (`NA`) gesetzt werden. Mit Hilfe der `describe`-Funktion (s.o.) kann man herausfinden, welchem Wert die Ausprägung `keine Angabe` entspricht und diese Ausprägung auf `NA` setzen. Hier werden also alle Fälle mit der Ausprägung 9 auf fehlend gesetzt. Anschließend wird die Variable in eine Faktorvariable umgewandelt und unbesetzte Kategorien - hier also die Kategorie `keine Angabe` - entfernt.

```
# `keine Angabe` auf fehlend setzen
df$schulab[df$schulab == 9] <- NA
```

```
# Umwandeln in Faktor und leere Kategorien entfernen
df$schulab <- to_factor(x = df$schulab, drop_unused_labels = TRUE)
```

Manchmal sind auch etwas komplexere Recodierungen notwendig. Als Beispiel soll die Variable Lebenszufriedenheit (`zuf001`) in eine neue Variable recodiert werden, die angibt, ob eine Person über- oder unterdurchschnittlich zufrieden ist. Um den Durchschnitt zu berechnen, müssen auch hier die Personen mit der Ausprägung `keine Angabe` zunächst auf `NA` gesetzt werden. `Keine Angabe` entspricht bei dieser Variable dem Wert 99. Anschließend kann der gewichtete Durchschnitt berechnet werden (der hier etwa 8,2[4] beträgt) und eine neue Variable entsprechend gebildet werden.

```
# `keine Angabe` auf fehlend setzen
df$zuf001[df$zuf001 == 99] <- NA
# gewichteten Durchschnitt berechnen (8,169571 ist ungefähr 8,2)
wtd.mean(x = df$zuf001, weights = df$pgewges)
```

```
## [1] 8.169571
```

```
# Neue Variable erstellen
df$zuf001_durchschnitt <- NA
# unterdurchschnittlich zufriedene Personen auf 0 setzen
df$zuf001_durchschnitt[df$zuf001 < 8.2] <- 0
# überdurchschnittlich zufriedene Personen auf 1 setzen
df$zuf001_durchschnitt[df$zuf001 > 8.2] <- 1
```

Diese Variable liegt nun als kontinuierliche Variable mit den Ausprägungen 0 und 1 vor. Auch diese Variable kann in eine Faktorvariable mit den entsprechenden Labeln umgewandelt werden:

```
df$zuf001_durchschnitt <- factor(x = df$zuf001_durchschnitt,
                                 labels = c("unterd. Zufr.", "überd. Zufr."))

describe(x = df$zuf001_durchschnitt)
```

```
## [7192 obs.]
## nominal factor: "unterd. Zufr." "überd. Zufr." "unterd. Zufr." "unterd. Zufr." "unterd. Zufr." "u
## 2 levels: unterd. Zufr. | überd. Zufr.
## NAs: 13 (0.2%)
##
##                   n     %  val%
## unterd. Zufr. 4098  57.0  57.1
## überd. Zufr.  3081  42.8  42.9
## NA              13   0.2    NA
## Total         7192 100.0 100.0
```

Die Variable `zuf001` wird nicht in einen Faktor umgewandelt, da sie weiterhin wie eine kontinuierliche Variable behandelt werden soll.

4.3.5 Neue Referenzkategorie setzen

Für Regressionen soll manchmal das Referenzlevel nicht das erste Level sein, sondern etwa der mittlere Wert von drei Ausprägungen. Dazu kann ein Faktor mit der Funktion `relevel` angepasst werden. Als Beispiel soll bei der Variable `altgr5`, welche das Alter der Befragten in fünf Gruppen darstellt, die älteste Altersgruppe als Referenzkategorie gesetzt werden.

```
# Zuerst in Faktor umwandeln
df$altgr5 <- to_factor(x = df$altgr5)
# Dann die Referenzkategorie setzen
df$altgr5 <- fct_relevel(df$altgr5, "55-64 Jahre")
```

Die Benennung der Referenzkategorie muss exakt mit dem Werte-Label übereinstimmen und kann mit Hilfe der `describe`-Funktion herausgefunden und kopiert werden.

[4] R verwendet durchgängig den Punkt `.` als Dezimaltrennzeichen.

Die Reihenfolge der Kategorien wirkt sich auch auf Häufigkeitstabellen etc. aus. Auch hier kann es daher manchmal sinnvoll sein eine neue Variable zu erstellen. Alternativ kann auch bei den Analysen selbst eine andere Referenzkategorie gesetzt werden (s.u.).

4.4 Durchführen von Analysen

Bei der Durchführung von Analysen mit dem Datensatz von LEO 2018 müssen zwei Dinge beachtet werden. Zum einen muss die Gewichtung berücksichtigt werden, um korrekte Ergebnisse von Schätzern und Standardfehlern zu erhalten. Hierfür ist das Paket `survey` geeignet, welches Funktionen für die Durchführung von Analysen (Durchschnitte, Häufigkeiten, Regressionsanalysen, etc.) mit Survey-Daten bereitstellt.

Zum anderen liegen die getesteten literalen Fähigkeiten im Datensatz als zehn sogenannte plausible Werte vor. Bei Berechnungen müssen diese wie multiple Imputationen behandelt werden. Dies bedeutet, dass alle Analysen entsprechend der Anzahl der plausiblen Werte (hier: 10) mehrfach durchgeführt werden müssen. Anschließend werden diese zehn Berechnungen mit speziellen Rechenregeln gepoolt um korrekte Ergebnisse zu erhalten. Hierfür ist das Paket `mitools` zuständig.

Dies bedeutet, dass Berechnungen mit nur einem PV oder einem durchschnittlichen PV keine korrekten Ergebnisse liefern! Es ist auch zu beachten, dass PVs nicht für die individuelle Diagnostik genutzt werden können. Zweck von plausiblen Werten ist die Bestimmung der Eigenschaften von (Sub-)Populationen.

4.4.1 Erstellen des Datenobjekts

Da der Umgang mit `survey` und `mitools` etwas umständlich ist, stellt das Paket `leo` eine Funktion bereit, die automatisch ein Datenobjekt erzeugt, welches für methodisch korrekte Auswertungen geeignet ist. Zuvor müssen, wie etwa oben beschrieben, die Arbeiten zur Datensatzaufarbeitung abgeschlossen sein. Insbesondere sollten Variablen ausgewählt und in Faktoren umgewandelt worden sein. Wichtig ist außerdem, dass im vorbereiteten Datensatz mindestens die Gewichtungsvariable `pgewges` und die PV-Variablen enthalten sind.

Für Auswertungen mit dem `survey`-Paket muss ein sog. Designobjekt erstellt werden, das unter anderem die Daten und Informationen zur Gewichtung enthält. Im Fall von LEO 2018 sind auch die Informationen zu den plausiblen Werten enthalten. Der oben vorbereitete Datensatz kann einfach in ein solches Designobjekt umgewandelt werden:

```
df.sd <- leo_svydesign(data = df)
```

Mit dem so erzeugten Designobjekt können nun Analysen am Datensatz durchgeführt werden. Dabei wird die jeweilige Analyse zunächst zehnfach durchgeführt, anschließend werden die Ergebnisse miteinander kombiniert (Pooling).

4.4.2 Durchschnitt berechnen

Bei der Berechnung des durchschnittlichen Schreib- und Lesekompetenz in der Bevölkerung auf der kontinuierlichen LEO-Skala ist das Vorgehen wie folgt:

```
# Berechnungen werden mit diesm Befehl automatisch 10x durchgeführt
pv_durchschnitt_temp <- with(df.sd, svymean(~pv, na.rm = TRUE))
# Ergebnisse werden gepoolt
pv_durchschnitt <- MIcombine(pv_durchschnitt_temp)
# Durchschnitt anzeigen
summary(pv_durchschnitt)
```

```
## Multiple imputation results:
##       with(df.sd, svymean(~pv, na.rm = TRUE))
##       MIcombine.default(pv_durchschnitt_temp)
##     results       se   (lower   upper) missInfo
## pv 51.90409 0.199654 51.51022 52.29797     23 %
```

```r
# Die Befehle können auch miteinander kombiniert werden
pv_durchschnitt <- MIcombine(with(df.sd, svymean(~pv, na.rm = TRUE)))
# Die kompaktere Schreibweise führt zum selben Ergebnis:
summary(pv_durchschnitt)
```

```
## Multiple imputation results:
##       with(df.sd, svymean(~pv, na.rm = TRUE))
##       MIcombine.default(with(df.sd, svymean(~pv, na.rm = TRUE)))
##     results       se   (lower   upper) missInfo
## pv 51.90409 0.199654 51.51022 52.29797     23 %
```

Video: LEO 2018: Pooling der Ergebnisse

Für die Lebenszufriedenheit kann das Ergebnis von oben repliziert werden.

```r
zuf001_durchschnitt <- MIcombine(with(df.sd, svymean(~zuf001, na.rm = TRUE)))
summary(zuf001_durchschnitt)
```

```
## Multiple imputation results:
##       with(df.sd, svymean(~zuf001, na.rm = TRUE))
##       MIcombine.default(with(df.sd, svymean(~zuf001, na.rm = TRUE)))
##         results         se   (lower   upper) missInfo
## zuf001 8.169571 0.02628425 8.118055 8.221087      0 %
```

Streng genommen müsste die Berechnung der durchschnittlichen Lebenszufriedenheit nicht zehn Mal erfolgen und daraus der Mittelwert gepoolt werden, da in diese Berechnungen keine Daten aus dem Kompetenztest und somit keine plausiblen Werte einfließen. Es schadet jedoch auch nicht hier trotzdem so vorzugehen: Der hier berechnete Durchschnitt ist identisch mit dem auf "normalem" Weg berechneten Durchschnitt weiter oben.

4.4.3 Häufigkeitstabellen

Häufigkeitstabellen können sowohl in absoluten als auch relativen (prozentualen) Häufigkeiten berechnet werden. Es ist auch möglich Kreuztabellen zu berechnen.

4.4.3.1 Absolute Häufigkeitstabellen Als Beispiel für eine absolute Häufigkeitstabelle dient hier eine andere Variable, die in Form plausibler Werte vorliegt. Statt der Literalität auf der kontinuierlichen LEO-Skala wird hier die Einteilung in die sogenannten Alpha-Levels genutzt. Die Alpha-Levels 1-3 entsprechen dabei geringer Literalität.

```r
level_abs <- MIcombine(with(df.sd, svytotal(~alpha_pv, na.rm = TRUE)))
level_abs
```

```
## Multiple imputation results:
##       with(df.sd, svytotal(~alpha_pv, na.rm = TRUE))
##       MIcombine.default(with(df.sd, svytotal(~alpha_pv, na.rm = TRUE)))
##               results       se
## alpha_pva1   45.8599 12.34056
## alpha_pva2  241.4815 29.44115
## alpha_pva3  579.5603 41.51163
## alpha_pva4 1472.6011 62.87005
## alpha_pva5 4852.3392 79.31979
```

Die Nachkommastellen kommen hier zu Stande, da für die Berechnungen die Gewichtungsvariable genutzt wird.

Für andere Variablen ist eine solche Tabelle genau so einfach zu erstellen:

```
schulab_abs <- MIcombine(with(df.sd, svytotal(~schulab, na.rm = TRUE)))
schulab_abs
```

```
## Multiple imputation results:
##       with(df.sd, svytotal(~schulab, na.rm = TRUE))
##       MIcombine.default(with(df.sd, svytotal(~schulab, na.rm = TRUE)))
##                               results       se
## schulabkein Abschluss       354.730 29.07345
## schulabniedrig             1636.904 53.34838
## schulabmittel              2410.198 63.58500
## schulabhoch                2675.297 62.56083
## schulabderzeit Schulbesuch  103.502 12.47517
```

Auch Kreuztabellen lassen sich erstellen. Dabei ist auf die Reihenfolge der Variablen zu achten, um die Kreuztabelle in die gewünschte Richtung zu erhalten. Im Beispiel wird betrachtet, wie sich Männer und Frauen jeweils auf die Alpha-Level aufteilen.

```
f001_level_abs <- MIcombine(with(df.sd,
                                 svyby(~f001, ~alpha_pv, svytotal, na.rm = TRUE)))
f001_level_abs
```

```
## Multiple imputation results:
##       with(df.sd, svyby(~f001, ~alpha_pv, svytotal, na.rm = TRUE))
##       MIcombine.default(with(df.sd, svyby(~f001, ~alpha_pv, svytotal,
##     na.rm = TRUE)))
##                     results        se
## a1:f001männlich   38.4227 11.903388
## a2:f001männlich  153.2228 21.832283
## a3:f001männlich  314.1987 32.150238
## a4:f001männlich  805.9216 51.723166
## a5:f001männlich 2334.7212 67.570828
## a1:f001weiblich    7.4372  3.982559
## a2:f001weiblich   88.2587 16.983136
## a3:f001weiblich  265.3616 28.369590
## a4:f001weiblich  666.6795 37.212682
## a5:f001weiblich 2517.6180 59.153422
```

4.4.3.2 Relative Häufigkeitstabellen

Statt der absoluten sind häufig relative, also prozentuale, Häufigkeitstabellen interessanter, insbesondere um verschiedene Gruppen zu vergleichen. Die Verteilung in Prozent auf die einzelnen Alpha-Levels erhält man wie folgt:

```
level_rel <- MIcombine(with(df.sd, svymean(~alpha_pv, na.rm = TRUE)))
level_rel
```

```
## Multiple imputation results:
##       with(df.sd, svymean(~alpha_pv, na.rm = TRUE))
##       MIcombine.default(with(df.sd, svymean(~alpha_pv, na.rm = TRUE)))
##                 results          se
## alpha_pva1 0.006376656 0.001711015
## alpha_pva2 0.033577142 0.004057121
## alpha_pva3 0.080585794 0.005629126
## alpha_pva4 0.204759935 0.008335856
## alpha_pva5 0.674700473 0.008765907
```

Um prozentuale Werte zu erhalten müssen die Werte mit 100 multipliziert werden.

Die Anteile der einzelnen Alpha-Levels zwischen Frauen und Männern unterscheiden sich. Um diese Unterschiede zu betrachten, kann eine relative Kreuztabelle berechnet werden, wobei auch hier auf die richtige "Richtung" zu achten ist.

```
level_f001_rel <- MIcombine(with(df.sd, svyby(~alpha_pv, ~f001, svymean, na.rm = TRUE)))
level_f001_rel
```

```
## Multiple imputation results:
##       with(df.sd, svyby(~alpha_pv, ~f001, svymean, na.rm = TRUE))
##       MIcombine.default(with(df.sd, svyby(~alpha_pv, ~f001, svymean,
##     na.rm = TRUE)))
##                           results          se
## männlich:alpha_pva1 0.010536909 0.003246495
## weiblich:alpha_pva1 0.002097731 0.001122735
## männlich:alpha_pva2 0.042019291 0.005878750
## weiblich:alpha_pva2 0.024894178 0.004757049
## männlich:alpha_pva3 0.086164766 0.008572246
## weiblich:alpha_pva3 0.074847681 0.007747214
## männlich:alpha_pva4 0.221013156 0.013322701
## weiblich:alpha_pva4 0.188043087 0.009730671
## männlich:alpha_pva5 0.640265878 0.013260382
## weiblich:alpha_pva5 0.710117323 0.011848377
```

So erhält man eine Kreuztabelle, die zeigt, wie sich ein Geschlecht auf die Alpha-Levels aufteilt. Es lässt sich erkennen, dass etwa 71 Prozent der Frauen dem Alpha-Level 5 zuzurechnen sind, aber nur etwa 64 Prozent der Männer.

Dreht man die Variablen im Funktionsaufruf um, erhält man eine andere Aussage.

```
f001_level_rel <- MIcombine(with(df.sd, svyby(~f001, ~alpha_pv, svymean, na.rm = TRUE)))
f001_level_rel
```

```
## Multiple imputation results:
##       with(df.sd, svyby(~f001, ~alpha_pv, svymean, na.rm = TRUE))
##       MIcombine.default(with(df.sd, svyby(~f001, ~alpha_pv, svymean,
##     na.rm = TRUE)))
##                     results         se
## a1:f001männlich 0.8378257 0.08504799
## a2:f001männlich 0.6352832 0.05091381
## a3:f001männlich 0.5420240 0.03789877
## a4:f001männlich 0.5471509 0.02134593
## a5:f001männlich 0.4811523 0.01026848
## a1:f001weiblich 0.1621743 0.08504799
## a2:f001weiblich 0.3647168 0.05091381
## a3:f001weiblich 0.4579760 0.03789877
## a4:f001weiblich 0.4528491 0.02134593
## a5:f001weiblich 0.5188477 0.01026848
```

Nun zeigt die Kreuztabelle, wie sich die Alpha-Levels auf die Geschlechter aufteilt. Es ist zum Beispiel zu erkennen, dass etwa 48 Prozent derjenigen auf Alpha-Level 5 männlich und etwa 52 Prozent weiblich sind.

Für die Auswertungen im Workshop empfehlen wir Ihnen statt der Variablen `alpha_pv` die Variable `lowlit` zu benutzen. Diese Variable dichotomisiert die Alpha-Levels in gering literalisierte (Alpha-Levels 1-3) und nicht gering literalisierte (Alpha-Levels 4 und darüber) Erwachsene. Dies macht die Interpretation der Kreuztabellen einfacher und übersichtlicher, wenn die andere Variable viele Ausprägungen hat.

```
f001_level_rel <- MIcombine(with(df.sd, svyby(~f001, ~lowlit, svymean, na.rm = TRUE)))
f001_level_rel
```

```
## Multiple imputation results:
##       with(df.sd, svyby(~f001, ~lowlit, svymean, na.rm = TRUE))
##       MIcombine.default(with(df.sd, svyby(~f001, ~lowlit, svymean,
##     na.rm = TRUE)))
```

```
##                       results         se
## a4-5:f001männlich 0.4965479 0.00873403
## a1-3:f001männlich 0.5834936 0.02730420
## a4-5:f001weiblich 0.5034521 0.00873403
## a1-3:f001weiblich 0.4165064 0.02730420
```

4.4.4 Regressionen

4.4.4.1 Lineare Regressionen Lineare Regressionen werden in erster Linie genutzt, wenn ein linearer Zusammenhang zwischen der kontinuierlichen abhängigen/zu erklärenden Variablen und der oder den unabhängigen/erklärenden Variablen vermutet wird. Die Regressionsgleichung wird als Formel in der Form `y ~ v1 + v2 + ...` deklariert.

Im Beispiel ist die abhängige Variable die Lese- und Schreibfähigkeit auf der kontinuierlichen LEO-Skala. Diese soll mit Hilfe der Variablen Geschlecht, Schulabschluss, Altersgruppe und allgemeine Lebenszufriedenheit erklärt werden.

```
# Spezifizieren des Regressionsmodells
m1 <- pv ~ f001 + schulab + altgr5 + zuf001
# Durchführen der linearen Regression
m1_reg <- with(df.sd, svyglm(formula = m1))
summary(MIcombine(m1_reg))
```

```
## Multiple imputation results:
##       with(df.sd, svyglm(formula = m1))
##       MIcombine.default(m1_reg)
##                                 results         se     (lower     upper) missInfo
## (Intercept)                  35.26543197 1.2436676 32.8225095 37.7083545     13 %
## f001weiblich                  2.01466331 0.3202481  1.3867522  2.6425744      5 %
## schulabniedrig                7.46473062 1.0042994  5.4948416  9.4346197      8 %
## schulabmittel                13.40108092 0.9489500 11.5410695 15.2610923      2 %
## schulabhoch                  16.91929415 0.9751197 15.0074663 18.8311220      5 %
## schulabderzeit Schulbesuch 13.50308778 1.4493938 10.6565243 16.3496512     13 %
## altgr518-24 Jahre            -0.39188690 0.5754886 -1.5216288  0.7378550     11 %
## altgr525-34 Jahre            -0.10848716 0.5024644 -1.0951366  0.8781623     12 %
## altgr535-44 Jahre            -0.33560079 0.5394018 -1.3984349  0.7272333     21 %
## altgr545-54 Jahre            -0.09315226 0.4175526 -0.9121552  0.7258507      8 %
## zuf001                        0.38253483 0.1066961  0.1724105  0.5926592     19 %
```

Da während der Datenaufbereitung die Variablen Geschlecht, Schulabschluss und Alter in Faktorvariablen umgewandelt werden, werden die in der Regression auch entsprechend behandelt. Die Zufriedenheit wurde als kontinuierliche Variable beibehalten.

Beim Schulabschluss wurde die Referenzkategorie bei der Datenaufbereitung nicht geändert, obwohl der mittlere Schulabschluss als Referenz hier Sinn machen könnte. Dies kann noch ad hoc bei der Definition der Regressionsgleichung geschehen.

```
# Spezifizieren des Regressionsmodells
m2 <- pv ~ f001 + relevel(schulab, ref = "mittel") + altgr5 + zuf001
# Durchführen der linearen Regression
m2_reg <- with(df.sd, svyglm(formula = m2))
summary(MIcombine(m2_reg))
```

```
## Multiple imputation results:
##       with(df.sd, svyglm(formula = m2))
##       MIcombine.default(m2_reg)
##                                                        results         se
## (Intercept)                                         48.66651289 0.9318206
## f001weiblich                                         2.01466331 0.3202481
## relevel(schulab, ref = "mittel")kein Abschluss     -13.40108092 0.9489500
## relevel(schulab, ref = "mittel")niedrig             -5.93635030 0.4595699
```

```
## relevel(schulab, ref = "mittel")hoch                  3.51821323 0.4051132
## relevel(schulab, ref = "mittel")derzeit Schulbesuch   0.10200686 1.1610112
## altgr518-24 Jahre                                    -0.39188690 0.5754886
## altgr525-34 Jahre                                    -0.10848716 0.5024644
## altgr535-44 Jahre                                    -0.33560079 0.5394018
## altgr545-54 Jahre                                    -0.09315226 0.4175526
## zuf001                                                0.38253483 0.1066961
##                                                          (lower      upper)
## (Intercept)                                          46.8351413  50.4978844
## f001weiblich                                          1.3867522   2.6425744
## relevel(schulab, ref = "mittel")kein Abschluss      -15.2610923 -11.5410695
## relevel(schulab, ref = "mittel")niedrig              -6.8407664  -5.0319342
## relevel(schulab, ref = "mittel")hoch                  2.7201191   4.3163074
## relevel(schulab, ref = "mittel")derzeit Schulbesuch  -2.1838490   2.3878627
## altgr518-24 Jahre                                    -1.5216288   0.7378550
## altgr525-34 Jahre                                    -1.0951366   0.8781623
## altgr535-44 Jahre                                    -1.3984349   0.7272333
## altgr545-54 Jahre                                    -0.9121552   0.7258507
## zuf001                                                0.1724105   0.5926592
##                                                     missInfo
## (Intercept)                                            15 %
## f001weiblich                                            5 %
## relevel(schulab, ref = "mittel")kein Abschluss          2 %
## relevel(schulab, ref = "mittel")niedrig                18 %
## relevel(schulab, ref = "mittel")hoch                   20 %
## relevel(schulab, ref = "mittel")derzeit Schulbesuch    19 %
## altgr518-24 Jahre                                      11 %
## altgr525-34 Jahre                                      12 %
## altgr535-44 Jahre                                      21 %
## altgr545-54 Jahre                                       8 %
## zuf001                                                 19 %
```

Es ist aber vorzuziehen solche Änderungen bereits während der Datenaufbereitung vorzunehmen, auch weil - wie hier zu sehen ist - die Ausgabe unübersichtlicher wird.

4.4.4.2 Logistische Regressionen

Standardmäßig werden Regressionen als lineare Regressionen durchgeführt. Möchte man eine logistische Regression mit einer binären erklärenden Variablen durchführen, muss dies beim Funktionsaufruf spezifiziert werden.

Bei der Erzeugung des Datenobjekts wurde automatisch eine binäre Variable `lowlit` erzeugt welche angibt, ob die getesteten literalen Fähigkeiten einer Person in den Bereich geringer Literalität fallen oder nicht. Auch diese Variable liegt - wie alle Variablen, die Auskunft über die getesteten literalen Fähigkeiten einer Person geben - in Form von plausiblen Werten vor.

In der Regression soll daher untersucht werden, welchen Effekt die Variablen Geschlecht, Schulabschluss, Altersgruppe und allgemeine Lebenszufriedenheit auf die Chance haben, geringe literale Fähigkeiten zu besitzen.

```
# Spezifizieren des Regressionsmodells
m3 <- lowlit ~ f001 + schulab + altgr5 + zuf001
# Durchführen der logistischen Regression
m3_reg <- with(df.sd,
               svyglm(formula = m3, family = stats::quasibinomial(link = "logit")))
summary(MIcombine(m3_reg))
```

```
## Multiple imputation results:
##       with(df.sd, svyglm(formula = m3, family = stats::quasibinomial(link = "logit")))
##       MIcombine.default(m3_reg)
##                               results        se      (lower      upper)
## (Intercept)                1.0015990 0.3492291  0.31301098  1.69018703
```

```
## f001weiblich               -0.2758810 0.1287827 -0.52895691 -0.02280518
## schulabniedrig             -1.4559916 0.2328069 -1.91871674 -0.99326655
## schulabmittel              -2.7684854 0.2239061 -3.20878154 -2.32818923
## schulabhoch                -2.9548897 0.2516432 -3.45126790 -2.45851146
## schulabderzeit Schulbesuch -2.6163895 0.5730045 -3.74818996 -1.48458913
## altgr518-24 Jahre           0.1456964 0.2955304 -0.44522658  0.73661933
## altgr525-34 Jahre           0.1317179 0.2039716 -0.27050852  0.53394425
## altgr535-44 Jahre           0.4219752 0.2018482  0.02223638  0.82171399
## altgr545-54 Jahre           0.1160524 0.1691968 -0.21657646  0.44868134
## zuf001                     -0.1124432 0.0361288 -0.18361403 -0.04127241
##                            missInfo
## (Intercept)                    22 %
## f001weiblich                   14 %
## schulabniedrig                 34 %
## schulabmittel                  16 %
## schulabhoch                    23 %
## schulabderzeit Schulbesuch     25 %
## altgr518-24 Jahre              40 %
## altgr525-34 Jahre              22 %
## altgr535-44 Jahre              29 %
## altgr545-54 Jahre              15 %
## zuf001                         20 %
```

4.4.4.3 Regressionsergebnisse exportieren

Die Ergebnisse der eben durchgeführten Regressionen können in eine Excel-Tabelle exportiert werden. Dafür müssen die zwischengespeicherten Ergebnisse zunächst mit Hilfe der Funktion `leo_regresult` in eine Tabelle umgewandelt werden. Die Tabelle enthält neben den Regressionskoeffizienten auch die Standardfehler, p-Werte und auf Wunsch den Konfidenzintervall.

```
m1_tab <- leo_regresult(model = m1_reg, conf.int = TRUE)
m1_tab
```

```
## # A tibble: 11 x 7
##    term                        estimate std.er~1 stati~2  p.value conf.~3 conf.~4
##    <chr>                          <dbl>    <dbl>   <dbl>    <dbl>   <dbl>   <dbl>
##  1 (Intercept)                   35.3       1.24   28.4   0         32.8    37.7
##  2 f001weiblich                   2.01      0.320   6.29  3.58e-10   1.39    2.64
##  3 schulabniedrig                 7.46      1.00    7.43  1.73e-13   5.49    9.43
##  4 schulabmittel                 13.4       0.949  14.1   0         11.5    15.3
##  5 schulabhoch                   16.9       0.975  17.4   0         15.0    18.8
##  6 schulabderzeit Schulbesuch    13.5       1.45    9.32  0         10.7    16.3
##  7 altgr518-24 Jahre             -0.392     0.575  -0.681 4.96e- 1  -1.52    0.738
##  8 altgr525-34 Jahre             -0.108     0.502  -0.216 8.29e- 1  -1.10    0.878
##  9 altgr535-44 Jahre             -0.336     0.539  -0.622 5.34e- 1  -1.40    0.727
## 10 altgr545-54 Jahre             -0.0932    0.418  -0.223 8.23e- 1  -0.912   0.726
## 11 zuf001                         0.383     0.107   3.59  4.04e- 4   0.172   0.593
## # ... with abbreviated variable names 1: std.error, 2: statistic, 3: conf.low,
## #   4: conf.high
```

Anschließend kann die Tabelle in eine Excel-Datei exportiert werden.

```
write_xlsx(x = m1_tab, path = "ErgebnisseReg1.xlsx")
```

Es können auch mehrere Ergebnistabellen in eine Excel-Tabelle exportiert werden, die dann jeweils auf eigenen Arbeitsblättern erscheinen. Dazu müssen die Tabellen erst in einer Liste zwischengespeichert werden, die exportiert werden kann.

```
m2_tab <- leo_regresult(model = m2_reg, conf.int = TRUE)
m3_tab <- leo_regresult(model = m3_reg, conf.int = TRUE)
```

```r
ergebnisse <- list(Modell1 = m1_tab, # Der Name links vom = wird der Arbeitsblattname
                   Modell2 = m2_tab,
                   Modell3 = m3_tab)

write_xlsx(x = ergebnisse, path = "ErgebnisseReg1-3.xlsx")
```