



**Universität  
Zürich**<sup>UZH</sup>

**Wissen macht R!**

Dr. Benjamin Fretwurst

2023-01-01

Quellenangabe

---

---

# Inhalt

<b>1</b>	<b>Intro</b>	<b>1</b>
<b>2</b>	<b>Installation</b>	<b>2</b>
2.1	Downloadquellen . . . . .	2
2.1.1	R . . . . .	2
2.1.2	R Studio . . . . .	2
2.1.3	Zotero . . . . .	3
2.2	Reihenfolge der Installation . . . . .	3
2.3	Installationsanleitung . . . . .	3
2.3.1	Installation von tinytex . . . . .	3
2.3.2	Installation von Paketen . . . . .	4
2.4	Die Vorlagen . . . . .	4
<b>3</b>	<b>Fehler erkennen und lösen</b>	<b>6</b>
3.1	Vorgehen bei Fehlern . . . . .	6
3.2	Bei Fragen zu Fehlern mitzugeben . . . . .	6
3.3	Typische Fehler erkennen und beheben . . . . .	7
<b>4</b>	<b>Relitest</b>	<b>8</b>
<b>5</b>	<b>Relitest für Inhaltsanalysen</b>	<b>9</b>
5.1	Relitest . . . . .	9
<b>6</b>	<b>Daten laden und fusionieren</b>	<b>14</b>
6.1	Daten aus SosciSurvey (TF B) . . . . .	14
6.2	Datenfusionen (TF E) . . . . .	14
6.2.1	Die Befragungsdaten an die Inhaltsanalyse matchen . . . . .	14
6.2.2	Die Inhaltsanylisedaten an die Befragung machen . . . . .	15
6.3	Daten speichern und laden (TF E) . . . . .	16
<b>7</b>	<b>Datenaufbereitung</b>	<b>17</b>
7.1	Leere Fälle löschen (TF E) . . . . .	17
7.2	Variablen umbenennen und löschen . . . . .	17
7.3	Label für Variablen und Ausprägungen (teils TF E) . . . . .	18
7.4	Befragungstag aus dem Datum . . . . .	19
7.5	Umkodieren . . . . .	20
7.5.1	Werte wechseln . . . . .	20
7.5.2	Mehrere auf einmal umkodieren . . . . .	20
7.5.3	NA durch o ersetzen . . . . .	21
7.5.4	Ausprägungen zusammenfassen . . . . .	21
7.5.5	Medium umkodieren . . . . .	22
7.5.6	Bedingt umkodieren . . . . .	22
7.5.7	Variablen zu Indizes zusammenfassen . . . . .	23
7.5.8	Umfangreiche Umkodierungstabellen (TF E) . . . . .	23

---

<b>8</b>	<b>Ergebnisse</b>	<b>25</b>
8.1	Wofür welcher Datensatz? . . . . .	25
8.2	Stichprobenbeschreibung . . . . .	26
8.3	Häufigkeitstabelle mit Mehrfachantworten . . . . .	27
8.4	Vorkommen der NF . . . . .	28
8.5	Vergleiche des Vorkommens . . . . .	28
8.6	Bedeutung der Nachrichtenfaktoren für die journalistische Selektion . . . . .	29
8.6.1	Nachrichtenwert und Resonanz mehrerer Medien . . . . .	30
8.7	Thematisierung in Medien und Publikumsresonanz . . . . .	31
8.8	Zusammenspiel der Nachrichtenfaktoren . . . . .	32
8.8.1	Netzwerkplot der Korrelationen . . . . .	33
8.9	Wirkung der NF auf die Beachtungsindikatoren . . . . .	35
8.9.1	Regressionstabellen . . . . .	35
8.9.2	Regressionsdiagramm (Whyskerplot) . . . . .	36
8.10	Vergleich der NF-Gewichte nach Mediengruppen . . . . .	36
8.11	Unterschiede der Gewichte von NF nach Befragtenmerkmalen . . . . .	37
8.11.1	Zwei Gruppen als UV . . . . .	37
8.11.2	Mehrere Gruppen als UV . . . . .	38
<b>9</b>	<b>Forschungsbericht mit Quarto/Word</b>	<b>41</b>
9.1	Quarto ist besser und einfacher . . . . .	41
9.2	Tabellen und Grafiken nach Word schaffen . . . . .	42

---

## **Tabellen- und Abbildungsverzeichnis**

### **Abbildungsverzeichnis**

### **Tabellenverzeichnis**

---



# 1 Intro

R ist ein kostenloses, erweiterbares, modernes und freundliches Statistiksistem (programmierbar mit Lösungen in stapelweise Paketen). Mit R-Studio können Sie Ihre Datenverarbeitung und -analysen in R für sich und andere sehr gut dokumentieren. Mit den Einführungen auf dieser Seite sollen Sie in die Lage versetzt werden, so schnell und einfach wie möglich so viel wie möglich aus dem System herauszuholen. Es wird eine Herausforderung, die Sie schaffen werden. Gehen Sie gut gelaunt an R heran, üben Sie Frustrationstoleranz, machen Sie Ihre Tastatur nicht kaputt und sagen Sie sich immer wieder: Ich muss, ich will, ich kann. :-)

Schauen Sie am Anfang mal Ihre Installation an. Wenn Sie R und R-Studio das erste Mal installieren, dann können Sie das nach der Anleitung tun, die Sie unter [Installation] finden. Dort erfahren Sie auch, wie Sie «tinytex» mit R-Studio installieren und damit die Möglichkeit haben, sehr schöne Berichte direkt in R-Studio setzen zu lassen (zum Beispiel). Damit Sie gleich noch Literaturverweise und Literaturverzeichnis automatisiert mitproduzieren können, finden Sie in der Installationsanleitung auch gleich noch Zotero.

## Programmieren heisst kopieren

Sie lernen mit der Zeit wie Sie Aufgaben in R lösen, indem Sie sich Codeschnipsel suchen, diese kopieren und an Ihre Daten und Variablen anpassen. Mit R zu arbeiten verlangt von Ihnen keine aktiven Programmierkenntnisse. Sie müssen nur den richtigen Code (z.B. auf «Stack Overflow») finden und für sich anpassen. So arbeiten übrigens die meisten Programmierer:innen in ihrem Alltag.

## 2 Installation

R ist ein sehr leistungsstarkes System für statistische Analysen. Sie können R alleine herunterladen und den integrierten Editor nutzen, um mit R zu arbeiten. Viel komfortabler ist es aber, die R-Umgebung R-Studio zu nutzen, das Ihnen hilft, Ihre Daten und Variablen leichter zu finden und in den R-Code zu integrieren, Sie finden Ihre Daten leichter und die Dateien, mit denen Sie arbeiten. Zusätzlich hilft Ihnen R-Studio dabei, Ihre Arbeit mit R verständlich zu dokumentieren. Dafür nutzen Sie sehr komfortabel Markdown, eine ganz einfache Kennzeichnungssprache, um Texte zu schreiben und zu formatieren. In R-Studio haben Sie dafür sogar einen Formatierungseeditor, der Ihnen auch dafür Formatierungsschaltflächen zur Verfügung stellt, wie Sie sie von Word und ähnlichen Programmen kennen. Sie können mit dem System sogar so weit gehen, dass Sie Ihre Berichte komplett in R-Studio schreiben und am Ende als schönen PDF-Bericht rauslassen können und sogar einfach als Internetseite rauslassen können. Dabei werden Ihre Auswertungen in R als Tabellen und Grafiken direkt integriert und sie müssen es nicht in ein Word kopieren; UND, wenn Sie etwas an einer Variablen ändern oder etwas am Datensatz korrigieren, würde das automatisch für den ganzen Bericht geändert; UND mit einer für Sie erstellten Vorlage, haben Sie eine schöne Formatierungsvorlage und brauchen sich auch um solche Dinge nicht mehr gross zu kümmern. Dafür können/müssten Sie noch tinytex installieren, müssen sich dann später aber nicht mehr drum kümmern. Eine besondere Herausforderung wissenschaftlicher Arbeiten ist immer wieder die Erstellung von korrekten Literaturnachweisen und eines Quellenverzeichnisses. Auch da arbeitet das System für Sie. Für diese Aufgabe empfehle ich Ihnen Citavi zu nutzen oder Zotero. Ich zeige Ihnen daher noch, wie Sie Zotero installieren und nutzen können.

Hier können Sie sich die Installation des gesamten Systems im Video anschauen.

```
vembedr::embed_youtube("BVBvx99JMU0") |>  
vembedr::use_rounded()
```

### 2.1 Downloadquellen

#### 2.1.1 R

Laden Sie die neuste Version von R herunter:

- für Windows R base und am besten gleich noch R-tools R tools.
- für Macs hier R-[...].pkg für ältere Macs mit Intel-Chips oder R-[...]arm64.pkg für Macs mit M1 oder M2 (bei den Macs braucht es keine R-tools) .
- für Linux R für Ihre Linuxversion .

#### 2.1.2 R Studio

R-Studio finden Sie hier. Ihnen wird da ganz oben die Version angeboten, die für Ihr System die Richtige ist!



### 2.1.3 Zotero

Mit Zotero können Sie Literatur verwalten, Verweise leicht automatisch finden und einfach in RMarkdown-Dokumente integrieren oder auch Verzeichnisse für Word erstellen. Zotero kann kostenlos genutzt werden. Daher gehe ich hier eher darauf ein. Citavi ist auch gut und inzwischen zum Glück auch für den Mac verfügbar. Der Nachteil ist nur, dass Sie Citavi nicht mehr kostenlos nutzen, können, wenn Sie die UZH mal verlassen und an eine andere Uni gehen, wo das Programm vielleicht nicht kostenlos zur Verfügung gestellt wird. Daher gehe ich hier eher auf Zotero ein.

Hier finden Sie Zotero. Installieren Sie dann gleich noch die Erweiterung Dort finden Sie auch gleich die Anleitung für die Installation der Erweiterung.

## 2.2 Reihenfolge der Installation

1. Installieren Sie erst R base
2. dann Rtools (nur auf Windows)
3. dann R-Studio
4. dann tinytex
5. dann Zotero
6. dann Zotero-better-bibtex
7. dann legen Sie den Ordner mit den ganzen Vorlagen bei sich ab.

## 2.3 Installationsanleitung

R und R-Studio sind wie ganz normale Programme. Wie Sie die installieren, zeige ich im Video, aber eigentlich brauchen Sie das nicht unbedingt. Ein bisschen anders ist die Installation von tinytex in R. Und dann zeige ich Ihnen noch, wie Sie Pakete in R-Studio installieren.

### 2.3.1 Installation von tinytex

Gehen Sie in R-Studio oben im Menü auf den Menüpunkt «Tools» und dort ganz oben auf «Install Packages ...». Dann geben Sie im leeren Eingabefeld «Packages» ein: «tinytex». Wenn Sie es richtig geschrieben haben, steht es auch gleich fett zur Auswahl. Starten Sie die Installation, indem Sie auf den Schalter «Install» klicken.

Jetzt ist das Paket tinytex installiert, das Ihnen die Arbeit abnimmt, ein TeX-System zu installieren. Das machen Sie automatisch indem Sie in der R-Studio-Console eingeben: «tinytex::install\_tinytex». Dann machen Sie etwas Schönes. Sie haben selbst bei schnelleren Rechnern Zeit um sich einen Kaffee zu machen. Sie können aber auch in 20 Minuten etwas stöbern, um die 20 Minuten Installation vorbeiziehen zu lassen. Lassen Sie derweil Ihr System in Ruhe die Arbeit machen.

ODER, einfach diesen Code hier kopieren, in die R-(Studio)-Konsole einfügen und mit Enter abschicken:

```
# installiere das R-Paket tinytex
install.packages("tinytex")

# installiere ein winziges LaTeX-System
tinytex::install_tinytex()
```

### 2.3.2 Installation von Paketen

Die meisten Pakete können direkt von CRAN installiert werden. Das geht in RStudio recht einfach über das Menü «Tools» -> «Install Packages» und dann muss nur bei «Packages» eingegeben werden, welche Pakete installiert werden sollen. Manchmal gibt es die neusten Versionen eines Paketes nicht im Repository CRAN. Dann müssen sie direkt von den Entwicklern installiert werden. Die meisten Pakete werden dann über das sogenannte GitHub installiert. Das geht am einfachsten über Befehle nach dem Muster: `remotes::install_github(«Entwicklername/Paket»)`. Häufig wird dann von R noch gefragt, ob die Pakete und alle abhängigen Pakete installiert werden sollen. Oder R fragt «Möchten Sie versuchen, das Paket, welches eine Kompilierung erfordert, aus den Quelltexten zu installieren?». In der Regel ist es gut und richtig, wenn man das bestätigt (Sie geben unten in der «Console» ein: «Yes» und lassen R dann mal machen.). Manchmal gibt es aber einen Fehler weil ein Paket nicht kompiliert werden kann oder so (häufiger auf Macs der Fall). Dann ist es sinnvoll, dass man mal «n» für No eingibt. Dann wählt R die letzte verfügbare binäre, also vorkompilierte Version. Zum Beispiel ist das oft beim Paket «systemfonts» der Fall. Da also besser mal beim zweiten Versuch das Kompilieren ablehnen, also auf «no».

Auch diese Anleitung veraltet in der Regel schon nach einem Tag, weil die Entwickler für R ständig an ihren Paketen basteln. Wenn also eine Installation nicht funktioniert, ist es sinnvoll nach dem Paketnamen zu googlen, also «r Paketname» googlen. In der Regel bekommen Sie dann einen CRAN-Eintrag als einen der ersten Treffer. Wenn Sie da draufgehen, sehen sie recht technische Einträge. Dort suchen Sie dann «Vignette». In den Vignetten steht eigentlich immer als erstes, wie diese Pakete installiert werden sollen. Das kopieren Sie sich nach R und schicken die Zeile einmal ab. Achten Sie darauf, ob vielleicht mal ein Paket umbenannt wurde und es also Nachfolgepakete gibt oder so. Dann nehmen Sie natürlich am besten die neue eventuell umbenannte Version.

```
# installiere tidyverse:
install.packages("tidyverse")

## installiere die neuste Version von tidycomm

# Hier werden Sie gefragt, ob Sie versuchen möchten zu kompilieren und sagen "Yes" oder "No"

# jetzt installiere tidycomm
remotes::install_github("joon-e/tidycomm")
```

## 2.4 Die Vorlagen

Laden Sie diesen Zip-Ordner runter und speichern Sie ihn irgendwo (am besten in einem SWITCH-Drive-Ordner). Entpacken Sie den Ordner (meistens reicht der Doppelklick). In dem Ordner finden sie einige Dateien. Eine Datei heisst «Bericht.Rproj». Öffnen Sie die mal. Dann finden Sie im Panel rechts unten einige Dateien. Öffnen Sie «index.Rmd». Da ist das Setup drin und die Einleitung. Wenn Sie die öffnen und noch nicht alle Pakete installiert haben, die dafür benötigt werden, dann zeigt Ihnen R-Studio oben im Hauptfenster ein schmales gelbes Band an, in dem Sie darauf hingewiesen werden, dass nicht alle Pakete installiert sind. Dort können Sie die Installation direkt und automatisch starten lassen. Das dauert dann eine Weile, aber irgendwann ist es fertig. :-). Wenn die schrittweise Installation glattgelaufen ist, sollten Sie rechts oben einen Reiter sehen, der «Build» heisst. Wenn Sie darauf gehen, sehen Sie einen neuen Schalter mit einem Hammer darauf. Klicken Sie da

den Pfeil nach unten und dann auf «bookdown::pdf\_book» (Wenn hier nur Website steht, dann starten Sie R nach der Installation nochmal neu). Dann warten Sie (beim ersten Mal mit viel Geduld über 15-30 Minuten!). Am Ende sollte Ihnen ein PDF angezeigt werden.

---

# 3 Fehler erkennen und lösen

## 3.1 Vorgehen bei Fehlern

*Problemlösungsworkflow (je Schritt 15 Minuten)*

1. Fragen Sie erst sich, was wohl die Fehlermeldung heissen könnte und was anders ist zu Vorlagen und Beispielen (und google bzw. auf Stack Overflow suchen),
2. dann Ihre AG-Kollegen,
3. bei Task-Force-Aufgaben Ihre Task-Force-Kolleg:innen (zB TF D auf MS-Teams)
4. dann Ihre Task-Force-E,
5. dann im Forum auf OLAT,
6. dann direkt an [b.fretwurst@ikmz.uzh.ch](mailto:b.fretwurst@ikmz.uzh.ch) oder auf Teams «Fretwurst», «katharina» oder «Nadia»,
7. Stack Overflow mit «reprex» bzw. «mwe» (wenn Sie nicht wissen, was «reprex oder mwe sind, stellen Sie dort besser keine Fragen!»),
8. Fretwurst bei Paketautor:innen und externen Expert:innen.

## 3.2 Bei Fragen zu Fehlern mitzugeben

Wenn Sie Fehlermeldungen bekommen, senden Sie nicht einfach nur Screenshots! Damit kann niemand etwas anfangen und Sie wollen ja sicher auch die Lösungen nicht als Screenshot zurück haben ☐. Wenn Ihnen geholfen werden soll, müssen Sie die R-Zeilen (bzw. die R-Script-Datei) sowie den zugrundeliegenden Datensatz mitliefern und mitteilen, wo der Fehler auftritt.


Zu Ihren Fragen liefern Sie also bitte immer:

1. die Datendatei (als .RDS, .RData oder Excel-Datei),
2. Betroffener Befehl muss mitgeliefert werden als Text. Im Idealfall die .Rmd mit Zeilenhinweis (lokalisieren Sie den Fehler innerhalb einer Pipe).
3. Den Fehleroutput als Text und zusätzlich gerne als Screenshot.
4. Was schon probiert wurde.

**Gezippter Ordner:** Alles (1.-4.) zusammen – und mit der bei Ihnen vorliegenden Ordnerstruktur – können Sie uns schicken, indem Sie Ihren Projektordner komprimieren (z.B. als zip oder tar) und uns den per Teams oder Email schicken.

## 3.3 Typische Fehler erkennen und beheben

Die häufigsten Fehler und Lösungen sind:

1. Pakete sind nicht installiert: Tools  «Install Packages» Paket installieren
2. Pakete sind nicht geladen: Paket laden mit `library(«tidyverse»)` oder besser noch, die Paketnamen vor die Befehle schreiben (mit `::` dazwischen): `sjmisc::frq()`
3. Datensätze können nicht gefunden werden: Den richtigen Pfad einstellen mit Unterordner zB `DATEN_BF <- readRDS(«Daten/DATEN_BF.RDS»)`.
4. Datenobjekt wird nicht gefunden, weil es noch nicht geladen wurde: Daten laden zB `DATEN_BF <- readRDS(«Daten/DATEN_BF.RDS»)`
5. Datenobjekt wird nicht gefunden, weil es falsch geschrieben ist: Kontrollieren Sie auch Gross-Klein-Schreibung. Wenn Sie zB einen Objektnamen `DATEN_BF` markieren, dann `highlighted` R-Studio alle gleich geschriebenen Passagen. So sehen, Sie schnell, ob Sie sich irgendwo vertippt haben.
6. Variable wird nicht gefunden, weil sie falsch geschrieben ist: Schreiben Sie den Variablennamen neu und schauen Sie ob sie ihn in der Hilfsliste sehen.
7. Kommata oder Klammern fehlen oder sind zu viel: Es wird links in R ein roter Kreis mit weissem x angezeigt. Wenn Sie da drauf gehen, sagt Ihnen R-Studio in der Regel was fehlt oder zu viel ist.
8. Es werden mehrere Zeilen grün ausgeführt, aber es passiert nichts weiter: Schauen Sie ob am Ende ein Pipe-Operator (`|>`) zu viel ist.
9. Es kann «eine Funktion nicht gefunden» werden, weil ein Paket fehlt oder die Funktion ist falsch geschrieben ist: Googlen Sie die Funktion und schauen, aus welchem Paket die ist. Sie können dann probieren, ob es mit `Paketname::Funktion` geht zB `janitor::percent`.
10. Variablen sind vom falschen Typ: Wandeln Sie den Typ um. ZB: `DATEN_BF |> mutate(Nachricht = as.numeric(Nachricht))`.

Googlen Sie Ihre Fehlermeldungen oder Funktionen, die Fehler erzeugen und schauen sich den Syntax an, wie die Funktionen geschrieben sein sollten. Üben Sie sich in Frustrationstoleranz. Haben Sie Geduld und suchen Sie den Fehler im Detail.

Ein Kollege gibt hier ein paar Hinweise wie Fehler in R lesbar sind in einer PDF-Datei. Hier habe ich noch ein Video mit typischen Fehlern und Lösungen:

## 4 Relitest

## 5 Relitest für Inhaltsanalysen

Dazu gibt es auch ein Video:

Hier finden Sie den im Video erwähnten gezippten Ordner mit der Relitest.Rmd. Den Ordner müssen Sie entpacken und an eine gute Stelle in ihrem System tun. In dem Ordner wo sich diese Datei «Relitest.Rmd» befindet, muss es einen Unterordner mit der Bezeichnung «Daten» geben, wo die Excel-Dateien «RelidatenCoder1.xlsx» usw. liegen. s ## Relidaten Zusammenbinden

Sie werden die Inhaltsanalyse in Excel kodieren. Die Exceldateien müssen dann in R importiert werden. Das geht zB mit **read\_xlsx** des Pakets **read\_xl**. Wie Sie sehen werden die Exceldateien der Coder (im Beispiel 1 bis 3) jeweils in R-Dateien eingelesen (RelitestCoder1 <- ...) und dann mit dem Befehl rbind (steht für Rows aneinander binden) zu einer Datei zusammengefügt, die dann angeschaut oder für Relitests weiterverarbeitet werden kann.

```
# Lade alle Relitestdateien aus dem Ordner "Daten" (den es geben muss mit allen Relidatei

RelitestCoder1 <- readxl::read_excel("data/RelidatenCoder1.xlsx")
RelitestCoder2 <- readxl::read_excel("data/RelidatenCoder2.xlsx")
RelitestCoder3 <- readxl::read_excel("data/Klausis erster Relitest.xlsx")

# Binde die Relidatensätze der Coder:innen zu einem Datenobjekt "RelitestGesamt" zusammen:
Relitest_Gesamt <- rbind(RelitestCoder1, RelitestCoder2, RelitestCoder3)

Relitest_Gesamt |>
  dplyr::select(CODER, CU) |>
  dplyr::mutate(Fehler = ifelse(is.na(CODER) | is.na(CU), "FEHLER! Muss in der RelitestGe
  dplyr::count(Fehler)
## # A tibble: 2 x 2
##   Fehler
##   <chr>
## 1 FEHLER! Muss in der RelitestGesamt.xlsx repariert und als Relitest_bearbeitet.xls~
## 2 alles gut

# hier schreibe ich die Daten mal aus R raus in Excel, damit ich sie in Excel besser angu

writexl::write_xlsx(Relitest_Gesamt, "data/Relitest_Gesamt.xlsx")
```

### 5.1 Relitest

Den Relitest selbst kann man mit test\_icr des Pakets tidycomm machen. Dazu werden zunächst die Daten geladen (im Beispiel «RelitestDaten») und dann mit der Pipe (neu statt %>% der Pipoperator |>) der Befehl test\_icr ausgeführt. An erster Stelle steht die Variable für die Coding Unit (CU), die im Datensatz RelitestBeispiele einfach «CU» heisst. Als nächstes folgt die Variable für den Coder

(im Beispiel einfach «CODER»). In den Variablen CU und CODER dürfen nur Zahlen stehen! Also muss jeder Kodierer eine Nummer eingeben und auch die Artikel im Relimaterial nummeriert sein. Dann können Sie einzelne Variablennamen anfügen, für die die Reliwerte ausgewertet werden sollen oooooer Sie lassen das weg und es werden für alle Variablen Reliwerte berechnet. Wenn Sie fehlende Werte im Datensatz haben, dann sollten Sie am Ende «na.omit = TRUE» stehen haben, sonst meckert R, dass Fehlende Werte gefunden wurden.

```
# hier lade ich die Relidaten aus Excel wieder rein in R
RelitestDaten <- readxl::read_xlsx("data/Relitest_bearbeitet.xlsx")

# Sie können mit dem folgenden Befehl Variablen umkodieren und gleich in eckigen Klammern
RelitestDaten <- RelitestDaten |>
  mutate(Worte_gr = sjmisc::rec(Worte, rec =
    "min:100 = 1 [Kurzmeldung];
     101:150 = 2 [kurzer Artikel];
     151:300 = 3 [länger];
     301:500 = 4 [lang];
     501:max = 5 [Langtext]")) |>
  relocate(Worte_gr, .after = Worte) # hier sortiere ich die neue Variable direkt hinter

RelitestDaten |>
  sjmisc::frq(Worte_gr) # machen wir mal ne Häufigkeitsauszählung für "Worte_gr"
## Worte_gr <numeric>
## # total N=190 valid N=190 mean=3.61 sd=1.20
##
## Value |          Label | N | Raw % | Valid % | Cum. %
## -----
## 1 | Kurzmeldung | 19 | 10.00 | 10.00 | 10.00
## 2 | kurzer Artikel | 19 | 10.00 | 10.00 | 20.00
## 3 | länger | 18 | 9.47 | 9.47 | 29.47
## 4 | lang | 96 | 50.53 | 50.53 | 80.00
## 5 | Langtext | 38 | 20.00 | 20.00 | 100.00
## <NA> | <NA> | 0 | 0.00 | <NA> | <NA>

RelitestDaten <- RelitestDaten |>
  dplyr::mutate(Voyer_r = sjmisc::rec(Voyeur, rec = "0 = 0 [kommt nicht vor]; 1:3 = 1 [ko
  dplyr::relocate(Voyer_r, .after = Voyeur)

RelitestDaten |>
  sjmisc::frq(Voyer_r)
## Voyer_r <numeric>
## # total N=190 valid N=190 mean=0.44 sd=0.50
##
## Value |          Label | N | Raw % | Valid % | Cum. %
## -----
## 0 | kommt nicht vor | 106 | 55.79 | 55.79 | 55.79
## 1 | kommt vor | 84 | 44.21 | 44.21 | 100.00
```



```
## <NA> | <NA> | 0 | 0.00 | <NA> | <NA>
```

# hier kommt dann der eigentliche Relitest. in test\_icr muss ganz vorne der Variablenname

```
RelitestOutput <- RelitestDaten |>
  tidycomm::test_icr(unit_var = CU, coder_var = CODER, Medium:Kurios, kripp_alpha = TRUE,
```

RelitestOutput # hier den Output des Relitest mal schnell in R angucken

```
## # A tibble: 39 x 10
##   Variable   n_Units n_Coders n_Categories Level Agreement Holstis_CR Krippendorffs_A
## * <chr>     <int>   <int>     <int> <chr>     <dbl>     <dbl>
## 1 Medium          10     19         5 nomin~     1         1
## 2 ThemSchwer     10     19         6 nomin~     0.5       0.860
## 3 Titel           10     19        36 nomin~     0.1       0.665
## 4 Worte           10     19         42 nomin~     0         0.378
## 5 Worte_gr       10     19         5 nomin~     0.2       0.850
## 6 ArtLang        10     19         2 nomin~     0.7       0.959
## # i 33 more rows
## # i 2 more variables: Lotus <dbl>, S_Lotus <dbl>
```

# Das ist im Prinzip das Gleiche wie oben, aber für Worte wird das Skalenniveau "interval"

```
RelitestOutput <- RelitestDaten |>
  tidycomm::test_icr(unit_var = CU, coder_var = CODER, Medium:Kurios, #
    levels = c(Worte = "interval", ArtLang = "ordinal"), # hier die Skalenniveaus
    kripp_alpha = TRUE, lotus = TRUE, s_lotus = TRUE, agreement = FALSE, holsti =
  dplyr::mutate_if(is.numeric, round, digits = 2) |> # hiermit noch auf zwei Stellen r
  dplyr::mutate(S_Lotus = ifelse(is.na(S_Lotus), 1, S_Lotus)) # Wenn alle Coder:innen n
```

```
RelitestOutput |>
  kableExtra::kbl() |> # und hier noch hübsch und dann bunt mit kbl
  kableExtra::kable_material() |>
  kableExtra::column_spec(6, color = ifelse(RelitestOutput$Krippendorffs_Alpha > 0.3 &
    background = kableExtra::spec_color(RelitestOutput$Krippendorffs_Alpha[0:100],
  kableExtra::column_spec(7, color = ifelse(RelitestOutput$Lotus > 0.6 & RelitestOutput
    background = kableExtra::spec_color(RelitestOutput$Lotus[0:100], begin = 0,
  kableExtra::column_spec(8, color = ifelse(RelitestOutput$S_Lotus > 0.6 & RelitestOutp
    background = kableExtra::spec_color(RelitestOutput$S_Lotus[0:100], begin =
## Warning in ensure_len_latex(background, nrows, off, include_thead, "white", : The numb
## of provided values in background does not equal to the number of rows.

## Warning in ensure_len_latex(background, nrows, off, include_thead, "white", : The numb
## of provided values in background does not equal to the number of rows.

## Warning in ensure_len_latex(background, nrows, off, include_thead, "white", : The numb
## of provided values in background does not equal to the number of rows.
```

Variable	n_Units	n_Coders	n_Categories	Level	Krippendorffs_Alpha	Lotus	S_Lotus
Medium	10	19	5	nominal	1.00	1.00	1.00
ThemSchwer	10	19	6	nominal	0.79	0.91	0.89
Titel	10	19	36	nominal	0.64	0.79	0.79
Worte	10	19	42	interval	0.72	0.59	0.58
Worte_gr	10	19	5	nominal	0.78	0.92	0.90
ArtLang	10	19	2	ordinal	0.92	0.98	0.96
Datum	10	19	4	nominal	0.67	0.92	0.89
Format	10	19	4	nominal	0.85	0.96	0.95
Bilder	10	19	13	nominal	0.75	0.85	0.84
Nutzen_ig	10	19	4	nominal	0.37	0.83	0.78
Alltagswelt	10	19	3	nominal	0.05	0.79	0.69
Nah_Welt	10	19	2	nominal	0.40	0.88	0.77
Nah_EU	10	19	2	nominal	0.65	0.93	0.86
Nah_CH	10	19	2	nominal	0.42	0.84	0.68
Nah_Kant	10	19	2	nominal	0.38	0.88	0.76
Nah_Stadt	10	19	2	nominal	0.44	0.86	0.73
Nah_Gem	10	19	2	nominal	0.32	0.93	0.85
Nah_nix	10	19	2	nominal	0.07	0.97	0.95
Naehe	10	19	16	nominal	0.40	0.66	0.64
Voyeur	10	19	4	nominal	0.30	0.69	0.59
Voyer_r	10	19	2	nominal	0.31	0.78	0.56
Sex	10	19	1	nominal	1.00	1.00	1.00
Schicksal	10	19	2	nominal	0.36	0.83	0.66
Risiko	10	19	4	nominal	0.20	0.59	0.46
Serviceinfos	9	19	2	nominal	0.15	0.83	0.66
Person	10	19	5	nominal	0.28	0.63	0.53
Promis	10	19	4	nominal	0.43	0.73	0.64
Emo	10	19	4	nominal	0.15	0.68	0.57
Einfluss	10	19	4	nominal	0.54	0.81	0.75
Kontroverse	10	19	4	nominal	0.40	0.79	0.72
Konflikt	10	19	2	nominal	0.10	0.91	0.82
Normbruch	10	19	4	nominal	0.68	0.87	0.82
Katastrophe	10	19	4	nominal	0.39	0.81	0.74
Aktual	10	19	4	nominal	0.07	0.56	0.42
Kontinuität	10	19	2	nominal	0.49	0.82	0.64
Ueberrasch	10	19	3	nominal	0.01	0.74	0.61
Sensation	9	19	2	nominal	0.24	0.74	0.47
Superlative	10	19	2	nominal	0.01	0.95	0.91
Kurios	10	19	2	nominal	0.21	0.79	0.59

---

```
writexl::write_xlsx(RelitestOutput, "RelitestOutput.xlsx")
```

# 6 Daten laden und fusionieren

## 6.1 Daten aus SosciSurvey (TF B)

Gehen Sie in SosciSurvey unter «Steuerung» auf «Erhobene Daten» und dann auf «Auswahlkriterien für gültige Fälle» und haken Sie «Interview (Aufruf der Fragenbogen-URL)» an und den Rest wieder ab, falls das noch an ist. Dann gehen Sie unter «Erhobene Daten» auf «Daten herunterladen». Dort finden Sie mehrere Reiter für verschiedene Datentypen. Sie können Excel nehmen und sich so die Daten bequemer ansehen. Nehmen Sie dann für den R-Datensatz besser den Reiter «GNU R» und benennen Sie dort unter «Name d. Daten-Frame in R:» ihren Datensatz so wie Sie ihn gerne hätten (meine Empfehlung DATEN\_BF). Bei «Variablen-Typen» nehmen Sie «Numerische Codes für Skalen ...» und bei «Residualoptionen:» nehmen Sie «Werte behalten, ...». Etwas weiter unten sehen Sie dann noch den Reiter «Variablen». Dort nehmen Sie «Variablen, die im heruntergeladenen Datensatz min. zwei unterschiedliche Ausprägungen haben» (alles andere sind langweilige Konstanten). Darunter können Sie noch die «Verweildauer ...» anhakeln und «Kennwerte zur Datenqualität ...». In der R-Onlineanleitung zu Methoden-Aufbau haben Sie das auch im Video erklärt.

Einige Variablen kommen aus SosciSurvey als logische Variablen (True/Fals), die wollen wir alle gleich mal in Dummies (0/1-Variablen) umwandeln, weil wir wissen, dass 0 immer «FALSE» bedeutet und 1 «TRUE» und wir mit 0/1-Variablen (den Dummies) viel besser arbeiten und rechnen können.

```
# mutiere die Variablen für die gilt: is.logical als numerisch
DATEN_BF <- DATEN_BF |>
  mutate(across(where(is.logical), as.numeric))
```

## 6.2 Datenfusionen (TF E)

Da wir Kommunikationswissenschaftler sind und Medieneigenschaften mit Befragungsergebnissen verbinden wollen, fusionieren wir hier die Datensätze der Inhaltsanalyse mit denen der Befragung.

### 6.2.1 Die Befragungsdaten an die Inhaltsanalyse matchen

```
# Erstmal die beiden Datensätze laden:

IA <- readxl::read_excel("Daten/IA_Gesamt.xlsx")
BEF <- readxl::read_excel("Daten/Befragung_ÜX.xlsx")

# Hier mal eine einfache Häufigkeitsauszählung
BEF |> sjmisc::frq(Nenn_Code1, Nenn_Code2, NennCode3)

# Und hier wird gezählt, wie oft in BEF die verschiedenen Themen in der ersten Nennung vor
```

```

Nenn1_DT <- BEF |> # Erstelle eine neue Datentabelle (DT) "Nenn1" die aus BEF verarbeitet
  count(Nenn_Code1, name = "Nenn1") |> # gruppierere nach der Variablen "Nenn_Code1" im Dat
  rename("Thema" = "Nenn_Code1") # Nenne jetzt noch die Bezeichnung "Nenn_Code1" in "Them

# ... und hier für die zweite
Nenn2_DT <- BEF |>
  count(Nenn_Code2, name = "Nenn2") |>
  rename("Thema" = "Nenn_Code2")

# ... Sie ahnen es:
Nenn3_DT <- BEF |>
  count(NennCode3, name = "Nenn3") |>
  rename("Thema" = "NennCode3")

# Jetzt bauen wir einen Datensatz für die Themen, die Zählungen hat, wenn sie in "Nenn1_D
Nennungen <- full_join(Nenn1_DT, Nenn2_DT, by = "Thema") |> # erstmal die Nenn2-Daten an
  full_join(., Nenn3_DT, by = "Thema") |> # dann noch die Nenn3_DT an die beiden von oben
  rowwise() |> # mache einen Summenindex, der Zeilenweise (rowwise) ...
  mutate(Nenn_Gesamt = sum(c(Nenn1, Nenn2, Nenn3))) # ... die Anzahl der Nennungen aus Ne

# Diese Anzahl der Nennungen kleben wir jetzt hinten an den IA-Datensatz und nennen den f
IAuBEF <- left_join(IA, Nennungen, by = c("Thema" = "Thema"))

```

### 6.2.2 Die Inhaltsanylседaten an die Befragung machten

```

# Hier mache ich eine Tabelle "IA-Thema",
IA_Thema <- DATEN_IA |>
  filter(!is.na(TC)) |> # die keine NAs enthält und ...
  group_by(TC) |> #... und nach "Thema" gruppiert ist, also Infos pro Thema enthält und z
  summarise(MedienVorkomm = n(), # Die Anzahl der Artikel, wo das Thema vorkam
            KURIO = mean(KURIO), # Die durchschnittliche Reichweite, die für die Artike
            KNTR = mean(KNTR), # das Gleiche für das Risiko
            SCHICK = mean(SCHICK),
            ) # und Personalisierung

# Hier werden jetzt, ähnlich wie oben, die Datensätze der IA an die BEF geklebt und zu BE
BEFuIA <- DATEN_BF |>
  select(-any_of(c("MedienVorkommen1", "KURIO1", "KNTR1", "SCHICK1",
                  "MedienVorkommen2", "KURIO2", "KNTR2", "SCHICK2",
                  "MedienVorkommen3", "KURIO3", "KNTR3", "SCHICK3"))) |>
  left_join(., IA_Thema, by = c("TCW1" = "TC")) |> # zu jeder Nennung (TCW) wir das "Them
  rename(MedienVorkommen1 = "MedienVorkomm", KURIO1 = "KURIO", KNTR1 = "KNTR", SCHICK1 =
  left_join(., IA_Thema, by = c("TCW2" = "TC")) |> # jetzt die Themenvariablen nach Nenn2
  rename(MedienVorkommen2 = "MedienVorkomm", KURIO2 = "KURIO", KNTR2 = "KNTR", SCHICK2 =
  left_join(., IA_Thema, by = c("TCW3" = "TC")) |> # und, is klar, gell

```

```

rename(MedienVorkommen3 = "MedienVorkomm", KURIO3 = "KURIO", KNTR3 = "KNTR", SCHICK3 =
DATEN_BF <- BEFuIA |> # Jetzt machte ich aus den Variablen Themen noch jeweils einen Mitt
rowwise() |>
mutate(KURIO_m = mean(c(KURIO1, KURIO2, KURIO3), na.rm = T)) |>
mutate(KNTR_m = mean(c(KNTR1, KNTR2, KNTR3), na.rm = T)) |>
mutate(SCHICK_m = mean(c(SCHICK1, SCHICK2, SCHICK3), na.rm = T)) |>
mutate(KURIO_m = replace(KURIO_m, is.na(KURIO_m), 0)) |> # für eine Variable die NA dur
mutate(across(c(KNTR_m, KURIO_m, SCHICK_m), ~replace(.x, is.na(.x), 0) )) # across mehr

saveRDS(DATEN_BF, file = "Daten/DATEN_BF.RDS")

```

### 6.3 Daten speichern und laden (TF E)

Wenn Sie durch den Import der Daten aus SosciSurvey ein Objekt mit dem Namen DATEN\_BF haben, dann können Sie dieses Objekt als R-Daten speichern. Später können Sie diese Daten immer wieder mit readRDS aufrufen. Sie können so auch zwischendurch mal ihre Daten als Datensatz speichern. Ich selbst packe mir diese Zeilen immer ganz an den Anfang in den r-Chung «Setup» wo auch die Pakete geladen werden. Die Daten liegen und werden abgespeichert im Unterordner «Daten».

```

DATEN_BF <- readRDS("Daten/DATEN_BF.RDS")
saveRDS(DATEN_BF, file = "Daten/DATEN_BF.RDS")

# Das erste Mal oder wenn Sie nochmal was in der Excel editiert haben:
#DATEN_IA <- read_xlsx("Daten/DATEN_IA.xlsx")

DATEN_IA <- readRDS("Daten/DATEN_IA.RDS")
saveRDS(DATEN_IA, file = "Daten/DATEN_IA.RDS")

DATEN_BF <- readRDS("Daten/BEFuIA.RDS")

DATEN_IA <- readRDS("Daten/IAuBEF.RDS")

```

## 7 Datenaufbereitung

Wer sich in die Datenaufbereitung (Data transformation) etwas tiefer einarbeiten will (oder weil er:sie muss), empfehle ich das gut lesbare und klaren Beispielen und schönem Code versehene <https://r4ds.hadley.nz/data-transformation>.

Laden Sie sich als erstes die Datei Auswertung.Rmd herunter. Darin finden Sie Codebeispiele, die Sie kopieren können und so ändern, dass Sie auf Ihre Datensätze und zu Ihren Variablen passen.

Erstmal ein paar Daten laden:

### 7.1 Leere Fälle löschen (TF E)

Hier gehe ich davon aus, dass ein Interview erfolgreich beendet wurde, wenn der:die Interviewer:in am Ende etwas zum Verständnis des Interviews eingegeben hat.

```
DATEN_BF <- DATEN_BF |>
  filter(!is.na(PLZ))

## Das geht nur einmal, weil R dann die Variablen nicht mehr findet, die wir gerade gelö
DATEN_BF <- DATEN_BF |>
  select(-any_of(SERIAL:MODE))
```

### 7.2 Variablen umbenennen und löschen

Wenn Sie den einfachen Befehl «select» nehmen, dann werden die Variablen behalten, die Sie auflisten. Wenn Sie ein Minus davor schreiben, werden die Variablen gelöscht, die Sie auflisten. Häufig führt das später zu Fehlermeldungen, weil bei mehrfacher Ausführung gelöschte Variablen natürlich nicht mehr im Datensatz gefunden werden. Um diesen Fehlermeldungen aus dem Weg zu gehen, können Sie «any\_of» verwenden. Dann prüft R, ob eine der aufgeführten Variablen im Datensatz sind und löscht diese. Wenn die Variablen nicht im Datensatz sind, dann macht R nichts weiter (meldet auch keinen Fehler und stoppt Ihren Durchlauf nicht.). Seien Sie damit also mir Vorsicht. Beim Befehl «rename» können Sie das auch machen. Mit «any\_of» werden nur die Variablen umbenannt, die auch wirklich da sind. Manchmal kommt das bei jedem Durchlauf vor, weil zB bei join-Befehlen immer nochmal ein Datensatz gematcht wird und wie im Beispiel vorhandene Variablen wie «Nenn1» umbenannt werden in «Nenn1.x» und Nenn1.y” Wenn Sie jetzt die mit «.y» löschen und die mit «.x» umbenennen, dann haben Sie das Problem auch glöst, wenn Ihnen nicht ganz klar ist, warum die immerwieder auftauchen. Für’s Erste ist das also auch eine Lösung eines Problems. ;-)

```
# Manchmal macht es Sinn, von einer Variable eine Kopie mit besser lesbarem Namen anzulegen
DATEN_BF <- DATEN_BF |>
```

```

mutate(Alter = ALTER,
       Geschlecht = GESCHL)

## verschiebe (relocat) die Variable GESCHL hinter die SD02.
DATEN_BF <- DATEN_BF |>
  relocate(Geschlecht, .after = GESCHL)

# So können Sie Variablen löschen. Durch das -any_of gibt es keine Fehler, wenn R die Var
DATEN_BF <- DATEN_BF |>
  select(-any_of(c("Nenn1.y", "Nenn2.y", "Nann3.y")))

# So nenne Sie Variablen um, wenn es sie gibt (vorne die neue Bezeichnung).
DATEN_BF <- DATEN_BF |>
  rename(any_of(c("Nenn1" = "Nenn1.x", "Nenn2" = "Nenn2.x", "Nenn3" = "Nenn3.x")))

```

### 7.3 Label für Variablen und Ausprägungen (teils TF E)

Wenn Sie messen, dann stehen am Ende Zahlen für das was Sie gemessen, also zB erfragt haben. Zum Beispiel steht dann eine 1 für «trifft nicht zu» und 5 für «trifft vollkommen zu». In Ihren Auswertungen müssen Sie aber wieder die verbalen Entsprechungen nutzen, statt nur Tabellen oder Grafiken zu haben, wo nur Zahlen ausgewiesen werden (Sie können das noch verbinden, aber Ihre Leser:innen nicht). In anderen Programmen (wie SPSS) ist es daher üblich, dass die gemessenen Zahlen mit verbalen Labels versehen werden. In Base-R geht das nicht so toll. Da müssen Sie mit «factors» arbeiten, wo also statt der Zahlen die verbalen Entsprechungen verwendet werden. Mühsam an den Faktoren ist, dass man mit ihnen wieder nicht mehr rechnen kann, also z.B. weder einen Mittelwert ausgeben noch eine Korrelation. Da hilft das Paket `sjmisc` weiter. Mit dem können Sie den Zahlenwerten Ihrer Variablen ihre Merkmalsausprägungen als Label zuordnen. Das geht mit `set_label(labels = c(<trifft nicht zu> = 1, <trifft voll zu> = 5))`:

```

# So können Label vergeben werden

DATEN_BF |> sjmisc::frq(GESCHL)

# Hier wird das Variablenlabel vergeben

DATEN_BF <- DATEN_BF |>
  sjlabelled::var_labels(GESCHL = "Geschlecht")

# Hier werden für die einzelnen Ausprägungen die Label vergeben
DATEN_BF <- DATEN_BF |>
  sjlabelled::set_labels(GESCHL, labels = c("männlich" = 1, "weiblich" = 2, "divers" = 3,

# Mit dem Befehl für Häufigkeitstabellen (frq für frequencies) kann die Labelerei schnell a

DATEN_BF |>
  sjmisc::frq(GESCHL)

```



```

## Hier werden für einige Variablen Label vergeben, die zusammengehören und später als Au
# Ersteinmal die Variablen angucken. Die Variablen gehen von NACHRICHT_TITEL1 in einer Re
DATEN_BF |>
  select(NACHRICHT_TITEL1:M112_23) |> # Nur die interessierenden Variablen selektieren
  ufs::multiResponse() |> # Mehrfachantworten ansehen
  arrange(desc(Frequency)) |> # Sortieren nach Häufigkeit
  kableExtra::kable() |> # besser lesbare Tabelle daraus machen
  kableExtra::kable_styling() # Styling

# Mit set_variable_labels werden hier die Variablenlabel vergeben
DATEN_BF <- DATEN_BF |>
  sjlabelled::var_labels(NACHRICHT_TITEL1 = "NZZ",
                        NACHRICHT_TITEL2 = "TA",
                        NACHRICHT_TITEL3 = "20 Minuten",
                        NACHRICHT_TITEL4 = "Blick",
                        NACHRICHT_TITEL5 = "Watson",
                        NACHRICHT_TITEL6 = "SRF News",
                        M112_20 = "Social Media",
                        M112_21 = "Google News",
                        M112_23 = "anderes"
                        )

# Hier mal für den Nachrichtenfaktor "Kontroverse" aus der Inhaltsanalyse

DATEN_IA <- DATEN_IA |>
  sjlabelled::var_labels(KNTR = "Kontroverse") |>
  sjlabelled::set_labels(KNTR, labels = c("keine" = 0, "gering" = 1, "stark" = 2, "k.A." =

```

## 7.4 Befragungstag aus dem Datum

Mit diesem Befehl kann der Befragungstag aus der Datumsvariable erzeugt werden, die von SosciSurvey automatisch erzeugt wird.

```

DATEN_BF <- DATEN_BF |>
  mutate(Befrag_Tag = day(STARTED)) |> # mit day wird aus der Datumsvariable der Tag raus
  relocate(Befrag_Tag, .after = STARTED) # Hiermit wird die Variable für den Befragungsst

DATEN_BF |> sjmisc::frq(Befrag_Tag)

```

## 7.5 Umkodieren

### 7.5.1 Werte wechseln

Umkodieren bedeutet, dass Ausprägungen (also Werte) in einer Variablen verändert werden oder zu anderen Werten in einer neuen Variable werden. Typische Umkodierungen sind Veränderungen der Codes (also zB wenn man eine Skala umdrehen möchte von 1 bis 5 auf 5 bis 1) und Zusammenfassungen von Codes, also zB 1 und 2 werden neu in 1 zusammengefasst 3 wird die neue 2 und 4 sowie 5 werden als neue 3 kodiert. Dann hat man jeweils zwei Abstufungen in den Ausprägungen zu jeweils einer zusammengefasst.

```
DATEN_BF <- DATEN_BF |>
  rename(Bildung = Bildung)

## Hier machen wir aus den beiden tieferen Bildungsgruppen eine einzelne Gruppe. In den K

DATEN_BF |>
  sjmisc::frq(Bildung) # erstmal angucken

# Erstelle mit mutate eine neue Variable Bildung_gr, die eine rekodierte Bildung (zB 1 un
DATEN_BF <- DATEN_BF |>
  mutate(Bildung_gr = sjmisc::rec(Bildung, rec = "1:2 = 1; 3 = 2; 4 = 3; 5 = 4")) # die o

DATEN_BF <- DATEN_BF |>
  sjlabelled::set_labels(Bildung_gr, labels = c("tief" = 1, "mittel" = 2, "hoch" = 3, "no
  sjlabelled::var_labels(Bildung_gr = "Bildung (gruppiert)") # hier das Variablenlabel

DATEN_BF |>
  sjmisc::frq(Bildung_gr)
```

### 7.5.2 Mehrere auf einmal umkodieren

```
DATEN_BF |>
  sjmisc::frq(HAUFIGKEIT_01:HAUFIGKEIT_05)

# mutiere across die ganze Liste der Variablen von:bis, als Funktion (~) rec(.x) wobei .x
DATEN_BF <- DATEN_BF |>
  mutate(across(HAUFIGKEIT_01:HAUFIGKEIT_05,
    ~ sjmisc::rec(.x, rec = "1 = -2 [viel zu wenig]; 2 = -1 [zu wenig]; 3 = 0
      .names = "{.col}_um0")) |> # Hier werden an die Variablennamen Suf
  sjlabelled::var_labels(HAUFIGKEIT_01_um0 = "Corona", # Hier noch Label für die Variable
    HAUFIGKEIT_02_um0 = "Abstimmungen",
    HAUFIGKEIT_03_um0 = "Finanz",
    HAUFIGKEIT_04_um0 = "Klima",
    HAUFIGKEIT_05_um0 = "Geflüchtete")
```

```
DATEN_BF |>
  sjmisc::frq(HAUFIGKEIT_01_um0:HAUFIGKEIT_05_um0)
```

### 7.5.3 NA durch 0 ersetzen

Die Grundstruktur von `across`-Befehlen ist an erster Stelle, zum Beispiel in `mutate` der Aufruf `across` gefolgt von der Liste der Variablen, für die dieselbe Mutationsfunktion durchgeführt werden soll. Im folgenden Beispiel ist es `across(Nenn1:Nenn5)`. Dann folgt die Funktion, was durch `~` angezeigt beziehungsweise gemacht wird. Immer da, wo man die Variablen einsetzen würde (also zB, wenn man die Funktion nur für `Nenn1` schreiben würde, wo man die `Nenn1` hinsetzt), wird `.x` geschrieben.

```
DATEN_IA <- DATEN_IA |>
  mutate(across(Nenn1:Nenn5, ~ifelse(is.na(.x) == TRUE, 0, .x)))
```

### 7.5.4 Ausprägungen zusammenfassen

Mit dem sehr stark kombinierbaren Befehl `case_when` können Sie eine neue Variable erstellen (zB `ALTER_GR`) in der Klammer von `case_when()` die Bedingungen folgen (zB `ALTER < 0`) und dann das Funktionszeichen `~` und dann der Wert, der an der Stelle, wo die Bedingung stimmt, hingeschrieben werden soll, also zum Beispiel die 1, wenn `'ALTER <= 25`, dann (`~`) 1.

```
# Das Alter in Gruppen zusammenfassen
DATEN_BF <- DATEN_BF |>
  mutate(ALTER_gr = case_when(
    ALTER < 0 ~ NA_real_,
    ALTER <= 25 ~ 1,
    ALTER <= 45 ~ 2,
    ALTER <= 65 ~ 3,
    ALTER <= 85 ~ 4,
    ALTER > 85 ~ 5
  )) |>
  sjlabelled::set_labels(ALTER_gr, labels = c("bis 25" = 1, "26-45" = 2, "46-65" = 3, "66-85" = 4, "86-100" = 5))
  sjlabelled::var_labels(ALTER_gr = "Alter gruppiert")

DATEN_BF |> frq(ALTER_gr)

# Hier werden für die Inhaltsanalyse die Anzahl Worte in Gruppen zusammengefasst. Das ges

DATEN_IA <- DATEN_IA |>
  mutate(WORTE_gr = sjmisc::rec(TC,
    rec = "0:100 = 1 [Kurzmeldung];
          101:150 = 2 [kurzer Artikel];
          151:300 = 3 [länger];
          301:500 = 4 [lang];
          501:max = 5 [Langtext]")) |>
  relocate(WORTE_gr, .after = TC)
```

```
### Rekodierung der Anzahl Nennungen in wenige Gruppen
```

```
DATEN_IA <- DATEN_IA |>
  mutate(Nenn_Gr = sjmisc::rec(Nenn_Gesamt, rec = "1:3 = 1; 4:16 = 2; 17:100 = 3")) |>
  sjlabelled::set_labels(Nenn_Gr, labels = (c("selten" = 1, "mittel" = 2, "häufig" = 3)))
```

### 7.5.5 Medium umkodieren

```
# 1 SRF
# 2 20min
# 3 Watson
# 4 NZZ
# 5 Tagesanzeiger
# 6 Blick

DATEN_IA <- DATEN_IA |>
  mutate(Medium_Q = sjmisc::rec(MEDIUM,
    rec = "1,4,5 = 1 [Qualität]; 2,3,6 = 2 [Boulevard]"))

DATEN_IA |> sjmisc::frq(Medium_Q)
```

### 7.5.6 Bedingt umkodieren

```
DATEN_BF |>
  sjmisc::frq(ALTER, Bildung_gr)

DATEN_BF <- DATEN_BF |>
  mutate(Alt_Bild1 = if_else(ALTER < 35 & Bildung_gr == 3, 1, 2))

DATEN_BF |> sjmisc::frq(Alt_Bild1)

# Hier erstmal umkodieren und anschauen

DATEN_BF |>
  mutate(Alt_Bild2 = case_when(
    ALTER < 35 & Bildung_gr == 3 ~ 1,
    ALTER > 75 & BILDUNG_gr == 3 ~ 2,
  )) |>
  sjlabelled::var_labels(Alt_Bild2 = "Alter und Bildung") |>
  sjlabelled::set_labels(Alt_Bild2, labels = c("Junggebildet" = 1, "Altgebildet" = 2)) |>
  sjmisc::frq(Alt_Bild2) # erstmal nur eine Frequencies als Ergebnis und nicht gleich den

# Dann dem Datensatz hinzufügen
DATEN_BF <- DATEN_BF |>
```

```
mutate(Alt_Bild2 = case_when(
  ALTER < 35 & BILDUNG_gr == 3 ~ 1,
  ALTER > 75 & BILDUNG_gr == 3 ~ 2,
)) |>
sjlabelled::var_labels(Alt_Bild2 = "Alter und Bildung") |>
sjlabelled::set_labels(Alt_Bild2, labels = c("Junggebildet" = 1, "Altgebildet" = 2))
```

### 7.5.7 Variablen zu Indizes zusammenfassen

```
# Ein Summenindex für 10 Variablen zur Nachrichtennutzung Es wird zeilenweise die Summe f
DATEN_BF <- DATEN_BF |>
  mutate(MEDIEN_sum = rowSums(across(NACHRICHT_TITEL1:NACHRICHT_TITEL10), na.rm = TRUE))

DATEN_BF |>
  sjmisc::frq(MEDIEN_sum)

# Hier noch der Durchschnittswert als Index, was bei Dummies der Anteil der 1-er ist, also
DATEN_BF <- DATEN_BF |>
  mutate(MEDIEN_mean = rowMeans(across(NACHRICHT_TITEL1:NACHRICHT_TITEL10), na.rm = TRUE))

DATEN_BF |>
  sjmisc::frq(MEDIEN_mean)

mutate

DATEN_IA <- DATEN_IA |>
  mutate(across(c(PERS:KNTR, KAT:NAEHE), ~replace(.x, is.na(.x),0))) |>
  mutate(across(c(PERS:KNTR, KAT:NAEHE), ~replace(.x, .x == -9,0))) |>
  rowwise() |>
  mutate(Jour_Beacht = sum(c(PERS:KNTR, KAT:NAEHE)))

DATEN_IA |> sjmisc::frq(Jour_Beacht)
```

### 7.5.8 Umfangreiche Umkodierungstabellen (TF E)

Zum Beispiel für Ländereigenschaften über die Ländercodes

```
# Aus der Länderliste mit den Kodierungen kommen die Variablen für eine Laenderliste:
Laenderliste <- readxl::read_xlsx("data/Laender_Liste_Kodierung.xlsx", sheet = "Länderlis

# Die Codes für die Länder kommen aus einer anderen Untertabelle der Excel, darum steht h
Ländercodes <- readxl::read_xlsx("data/Laender_Liste_Kodierung.xlsx", sheet = "Ländercode
```

```
# Dann gibt es noch lauter weitere Codes, die wir jetzt anmatchen können, die können wir  
Land_AlphaISO <- readxl::read_xlsx("data/Laender_Liste_Kodierung.xlsx", sheet = "AlphaISO",  
  select(name, "alpha2", "alpha3", "CountryCode"))  
  
# Hier binden wir die zusammen, erst die Ländercodes an die Laenderliste und dann noch die  
Laender <- left_join(Laenderliste, Ländercodes, by = "Land") |>  
  left_join(Land_AlphaISO, by = "alpha2")  
  
# oh, und hier noch ein paar Handelsdaten, falls das jemanden interessiert  
WorldTrade <- economiccomplexity::world_gdp_avg_1998_to_2000  
  
# So, die kleben wir jetzt auch noch hinten an unsere Laendervariable  
Laender <- left_join(Laender, WorldTrade, by = c("alpha3" = "country"))  
  
# Schreibe die Laender-Daten raus in die Exceldatei "Laender_codes.xlsx"  
writexl::write_xlsx(Laender, "data/Laender_codes.xlsx")  
  
# Hier die Laender-Daten wieder aus der (eventuell ergänzten Laender-Excel) in R laden:  
Laender <- readxl::read_xlsx("data/Laender_codes.xlsx")  
  
# Jetzt kommt die Magie. Wie matchen die Laendervariable an die Inhaltsanalysedaten:  
  
# Hier lade ich die Inhaltsanalysedaten aus meiner Excel. Vermutlich heisst die Excel bei  
IA <- readxl::read_excel("data/IA_Gesamt.xlsx")  
  
# Hier werden alle Variablen aus der Laendertabelle hinten an die Inhaltsanalyse geklebt  
IA <- left_join(IA, Laender, by = c("LAND" = "Code"))
```

## 8 Ergebnisse

Das tidyverse wird geladen und die zwei Basispakete kableExtra sowie scales. Dann werden aus \_common.R ein paar Einstellungen geladen und dann die Datensätze für die Inhaltsanalyse und die Befragung.

```
library(kableExtra, scales)
suppressPackageStartupMessages("tidyverse")
```

```
[1] "tidyverse"
```

```
source("_common.R")
```

Loading required package: pacman

```
DATEN_IA <- readRDS("data/DATEN_IA.RDS")
DATEN_BF <- readRDS("data/DATEN_BF.RDS")
```

### 8.1 Wofür welcher Datensatz?

Für die meisten Auswertungen wird der Datensatz der Inhaltsanalyse schon reichen. Sie können damit beschreiben, in welchem Umfang welche Nachrichtenfaktoren vorkommen und welche Ausprägungen sie haben. Sie können auch testen, welche Nachrichtenfaktoren mit der journalistischen Beachtung (Platzierung und Umfang der Beiträge) in welchem Masse zusammenhängen (Kreuztabellen und Korrelationen). Sie können auch analysieren, wie die Nachrichtenfaktoren untereinander zusammenhängen und inwiefern sie in der Summe die journalistische Beachtung besser erklären als einzelne Nachrichtenfaktoren (Additivitätshypothese). Sie können auch untersuchen, ob bestimmte NF in Mediengruppen häufiger vorkommen und stärkere Durchschnittsintensitäten haben als in anderen. Mit den Daten der Inhaltsanalyse lassen sich also viele Hypothesen schon aufklären.

Was allein mit dem Datensatz der Inhaltsanalyse nicht geht, ist die Bedeutung der NF für die Aufmerksamkeit bzw. Erinnerung an Meldungen durch die Rezipienten. Dafür braucht es die Ergänzung durch die Erkenntnisse aus der Befragung. Darum werden die Nennungen von Meldungen an den Inhaltsanalysedatensatz angefügt, also die Datensätze fusioniert. Dann können Sie das, was Sie mit der journalistischen Beachtung gemacht haben, für die Nennungen der Meldungen wiederholen und vergleichen, welche NF für die journalistische Beachtung eine andere Rolle spielen als für die Rezipienten. Wenn Sie aus der Befragung noch Informationen haben, für wie wichtig oder interessant die Befragten die von ihnen genannten Meldungen finden oder worüber sich die Befragten mit anderen unterhalten haben, dann können Sie feststellen, welche NF mehr auf das Interesse abzielen und welche eher als wichtig empfunden werden und welche mitbestimmen, worüber man sich so unterhält (was also Gesprächswert hat). Mit dem fusionierten Datensatz, bei dem die Inhaltsanalyse primär ist und die Nennungen der Befragten angefügt wurden, können die meisten Hypothesen

überprüft werden, da hier zum reinen IA-Datensatz noch die Informationen der Befragten hinzukommen.

Der Befragungsdatensatz alleine enthält nicht allzu viele Informationen. Die quasiexperimentellen Fragen, was für Schlagzeilen die Leute wie spannend finden, sind in der Regel nicht sehr ergiebig, weil sie eben nicht sehr valide sind (zu abstrakt und artifiziell). Die Befragung entfaltet ihren eigentlichen Wert erst mit der Datenfusion wie im vorherigen Abschnitt beschrieben. Allerdings haben Sie auch Daten über die Befragten, die dabei helfen können, die Befragten in Gruppen einzuteilen und dann zu schauen, welche Nachrichtenfaktoren bei diese Gruppen eine besondere Bedeutung haben. Dazu braucht es die Datenfusion der NF aus der Inhaltsanalyse an den Befragungsdatensatz. Dann haben sie für jeden Befragten, die durchschnittlichen Intensitäten (von 0 bis irgendwas) der Nachrichtenfaktoren, wie sie in den Meldungen vorkommen, die die Befragten genannt haben. Sie könnten also zB feststellen, ob ältere Befragte eher Meldungen mit anderen NF nennen als die jüngeren Befragte. Das wird vor allem spannend, wenn Sie die Nutzer:innen von Medien unterteilen und schauen, ob vielleicht Promistatus oder individueller Schaden bei Leser:innen von Boulevard- oder reinen Onlinemedien besonders hoch im Kurs sind und bei SRF- oder NZZ-Nutzer:innen eine geringere Rolle spielen. Sie könnten sogar eine Faktorenanalyse der NF der Befragten durchführen und so herausfinden, welche Kombinationen von NF bei den Befragten stärker zusammenhängen. Und sie könnten explorativ eine Clusteranalyse durchführen und Gruppen von Befragten bilden, die deutlich machen, dass es unterscheidbare Nutzer:innengruppen bei der Rezeption von Nachrichten gibt. Sie müssen nicht, aber Sie können das machen.

Manche AGs werden fast nur mit dem Datensatz arbeiten, der aus der Datenfusion von IA + Befragung entstanden ist (mit der IA alleine, werden viele Hypothesen ungeklärt bleiben, was natürlich nicht gut wäre). Da es Gruppen gibt, die Hypothesen darüber haben, wie sich die Bedeutung der NF zwischen Nutzer:innengruppen unterscheiden, brauchen die auch den Datensatz aus der Fusion Befragung + IA. Da ist die Fusion allerdings aufwendiger, fehleranfälliger und im Ergebnis mühsamer. Für das Zeitmanagement sollten alle AGs schon früh die vorbereinigten Daten aus der Befragung und der Inhaltsanalyse haben, um starten zu können. Wenn die Datenfusion IA + Befragung funktioniert hat und fertig ist, sollte die zur Verfügung gestellt werden. Als letztes sollten die fusionierten Daten aus Befragung + IA ausgegeben werden. Aus der stufenweisen Verteilung der Daten folgt, dass Datenaufbereitungen wiederholt werden müssen, also zB die Umkodierungen (mutate) und Labeli nochmals durchlaufen muss, die für die unfusionierten IA-Daten gemacht wurden, wenn die fusionierten Daten kommen. Das sollte allerdings kein Probleme sein, da diese Transformationen in R- oder Rmd-Dateien gespeichert sind und einfach nochmal durchlaufen müssen.

## 8.2 Stichprobenbeschreibung

Hier werden die Ergebnisse vor allem in Tabellenform präsentiert (Für ein Beispiel eines Verweises auf eine Tabelle siehe Tabelle @ref(tab:Mehrfachantworten)). Im Unikontext sind Grafiken und Diagramme eher dann angebracht, wenn Tabellen weniger Informationen bereitstellen würden oder so unübersichtlich würden, dass man sie kaum noch lesen kann. Das ist im normalen Studium selten der Fall. Schöne Diagramme können eine Arbeit optisch aufwerten aber meistens nicht inhaltlich. Wenn Grafiken für die Ergebnisdarstellung genutzt werden, sollten immer die differenzierteren Tabellen im Anhang aufgeführt werden.

---





## 8.4 Vorkommen der NF

Es können am Anfang die wichtigsten Variablen für die Stichprobenbeschreibung auch als einfache Häufigkeitsauszählungen dargestellt werden. Häufigkeitsauszählungen stellen die komplette Information einer Variablen dar, da jede mögliche Ausprägung wiedergegeben wird und ihre Anzahl im Datensatz. Das sind relativ viele Informationen. Wenn man das für mehrere Variablen macht, wird das viel Aufwand.

```
### Wenn Sie die Label bei der Datenaufbereitung schon vergeben haben, müssen Sie das hier

KNTR_frq <- DATEN_IA |> # baue eine Häufigkeitstabelle
  select(KNTR) |> # nur KNTR auswählen
  sjlabelled::as_label() |> # die Label verwenden
  tidycomm::tab_frequencies(KNTR) |> # eine Häufigkeitstabelle erstellt wird
  janitor::adorn_totals() |> # mit dieser Zeile können Sie eine Gesamtzeile unter Ihrer T
  mutate(Prozent = paste0(as.character(round(percent * 100, 0)), "%")) |> # Wenn Sie Proze
  select(1, n, Prozent) # hier nehmen wir von den Daten nur die erste Spalte 1 und die Fa

KNTR_frq |>
  kableExtra::kable(caption = "Kontroverse",
    booktabs = T,
    align = c('l', rep('r', 2)), # die 10 steht dafür, dass 10 weitere Spalten alle re
    linesep = "", # ohne diese Zeile, macht knitr alle 5 Zeilen einen kleinen Abstand
    col.names = c("Kontroverse", "Häufigkeit", "Prozent"), # Damit können die Überschr
  ) |> # noch mehr Anpassungsmöglichkeiten für kable finden Sie bei google: r kabl
  kableExtra::footnote(general = "Hier könnten/sollten Sie einen allgemeinen Hinweis oder
    general_title = "", # damit Kable unten nicht "Note" schreibt
    threeparttable = T # damit längere Fussnoten umgebrochen werden
  ) |>
  kableExtra::kable_styling()

rm(KNTR_frq) # Wird nicht mehr gebraucht und kann daher removed werden.
```

Textbeispiel: In 60 Prozent der Onlinebeiträge kam der Nachrichtenfaktor «Kontroverse» nicht vor. In knapp einem Drittel der Artikel wurde eine «geringe» Kontroverse beschrieben. In 57 Artikeln von 499 wurde eine starke Kontroverse thematisiert (11%).

## 8.5 Vergleiche des Vorkommens

Da das einzelne Vorkommen von Nachrichtenfaktoren im Grunde nichts über ihre Bedeutung sagt, sollten das Vorkommen der NF verglichen werden. Da die Nachrichtenfaktoren aber in der Regel mit unterschiedlichen Ausprägungen gemessen wurden, müssen sie erstmal auf einen gemeinsamen Standard gebracht werden. Das geht zum Beispiel indem man sie in Dummyvariablen umwandelt, also immer dann eine 0 setzt, wenn ein NF nicht vorkam (Wert 0 im Datensatz) und 1, wenn er grösser war als 0, also 1, 2, 3 usw. Mit solchen Dummyvariablen kann man ganz gut rechnen. So ist der Mittelwert einer Dummyvariable gleich dem Proporz der 1er (Summe der 1 = Vorkommen der 1 dividiert durch die Anzahl). Wenn man das mal 100 rechnet, dann hat man die Prozentwerte.

```
## Hier die Label vergeben, die nachher im Säulendiagramm erscheinen sollen.
DATEN_IA <- DATEN_IA |>
  sjlabelled::var_labels(PERS = "Personalisierung", KNTR = "Kontroverse", SCHAD = "Schaden")

DATEN_IA |>
  select(PERS, KNTR, SCHAD, NAEHE, MEDIUM) |> # Auswahl Ihrer NFs
  sjlabelled::label_to_colnames() |>
  mutate(across(everything(), ~if_else(.x > 0, 1, 0))) |> # Hier alle gleich behandeln, i
  summarise(across(everything(), ~round(mean(.x, na.rm = TRUE), 2) * 100)) |> # Mittelwert v
  pivot_longer(cols = (everything()),
               names_to = "NFs", values_to = "Prozent") |> # Die Tabelle muss noch ins L
  ggplot(aes(NFs, Prozent)) + # ACHTUNG bei ggplot für die Diagramme wird das + verwendet
  geom_col() +
  geom_text(aes(label = paste0(format(Prozent), "%")), vjust = -.4) +
  xlab("Nachrichtenfaktoren")
```

Textbeispiel: Personalisierung kam in 78 Prozent der Artikel vor. Das bedeutet, dass Personalisierung für die Berichterstattung ein wichtiger Nachrichtenfaktor ist. Ohne Personen geht es offenbar nicht. Die «Nähe» der berichteten Ereignisse ist in 50 Prozent der Artikel gegeben. Das deutet darauf hin, dass die Belange von besonderer Bedeutung sind, die eine höhere Betroffenheit für die Schweizer Bevölkerung haben, weil sie in ihre Nähe passieren. Medien berichten vor allem auch über Kontroversen oder kontroverse Themen. In der Stichprobe waren 40 Prozent der Artikel von Kontroversen geprägt. 38 Prozent der Artikel thematisierten einen «Schaden». Das bedeutet einerseits, dass 62% der Artikel keinen Schaden thematisiert haben, also dennoch wichtig genug waren. Andererseits bedeutet das, dass Ereignisse mit der Thematisierung von Schäden häufig in den Medien aufgegriffen werden. Um die Bedeutung der NFs im Vergleich beurteilen zu können, muss noch untersucht werden, ob sie alleine stark genug sind, um einen hohen Nachrichtenwert zu begründen oder erst durch das Zusammenspiel mit anderen Nachrichtenfaktoren eine Rolle spielen.

## 8.6 Bedeutung der Nachrichtenfaktoren für die journalistische Selektion

```
DATEN_IA <- DATEN_IA |>
  mutate(Nenn_Gr = sjmisc::rec(Nenn_Gesamt, rec = "1:3 = 1; 4:16 = 2; 17:100 = 3")) |>
  sjlabelled::set_labels(Nenn_Gr, labels = (c("selten" = 1, "mittel" = 2, "häufig" = 3)))

# DATEN_IA |> frq(Nenn_Gr) Immer mal schnell checken mit frq, aber nie in den Output!

SCHAD_Nenn_KREUZ <- DATEN_IA |>
  sjlabelled::as_label() |>
  tidycomm::crosstab(SCHAD, Nenn_Gr, add_total = TRUE, percentages = TRUE, chi_square = T
  select("<NA>") |>
  janitor::adorn_totals() |> # Wieder die Gesamtzeile
  mutate(across(c(-1), ~paste0(as.character(round(.x * 100, 0)), "%"))) # Wenn Sie Prozen
```

```

SCHAD_Nenn_KREUZ |> # Jetzt wird die Tabelle noch hübsch gemacht
  kableExtra::kable(caption = "Kreuztabelle für Schaden und Rezipienten-Nennungen", # n
    booktabs = T, # das sorgt für gute Bücherqualitätstabellen
    align = c('l', rep('r', NCOL(SCHAD_Nenn_KREUZ)-1)), # wenn ab der zweiten Spalte
    linesep = "", # ohne diese Zeile, macht knitr alle 5 Zeilen einen kleinen Abstand
    col.names = c("Nennungen", "keiner", "gering", "gross", "Gesamt"), # Damit können
  ) |> # noch mehr Anpassungsmöglichkeiten für kable finden Sie bei google: r kabl
  kableExtra::kable_styling() |> # hier können Sie stylen, aber bleiben Sie seriös :
  kableExtra::footnote(general = "Hier könnten/sollten Sie einen allgemeinen Hinweis oder
    general_title = "", # damit Kable unten nicht "Note" schreibt
    threeparttable = T # damit längere Fussnoten umgebrochen werden
  ) |> kableExtra::add_header_above(c("", "Schaden" = 3, "")) |>
  kableExtra::row_spec(4, bold = TRUE)

```

Textbeispiel: Wenn Onlinebeiträge keinen Schaden aufweisen, werden sie am häufigsten von den Rezipienten nur selten genannt (14%). Nennungen mittlerer Häufigkeit korrespondieren am häufigsten mit grossem Schaden (46%). Die häufigsten Nennungen kommen dann zustande, wenn Meldungen einen geringen «Schadenswert» aufweisen. Das kann daran liegen, dass am häufigsten das Thema «Corona» genannt wurde, wo häufig zwar ein Schaden thematisiert wird, aber eher ein geringer. (Kreuztabellen sind schwer zu texten!)

### 8.6.1 Nachrichtenwert und Resonanz mehrerer Medien

Mit dieser Analyse wird die Bedeutung der Nachrichtenfaktoren für die Selektion durch verschiedene Medien untersucht. Der Grundgedanke baut darauf auf, dass Ereignisse mit einem sehr hohen Nachrichtenwert im Grunde in allen Medien aufgegriffen werden. Wir untersuchen hier also, ob über die thematisierten Ereignisse in wenigen oder vielen Medien berichtet wurde. Dazu fassen wir die kodierten Themen pro Tag zusammen und schauen, in wie vielen Medien sie aufgegriffen wurden.

```

SELEKT <- DATEN_IA |>
  group_by(TC, TAG, MEDIUM) |> # Dieser Schritt ist notwendig, damit Merfachthematisieru
  summarise(n = n(), # Anzahl der Themen pro Tag in einem Medium
    PERS_m = max(PERS), # Hier werden die höchsten Werte je NF genommen, die an e
    KNTR_m = max(KNTR),
    SCHAD_m = max(SCHAD),
    NAEHE_m = max(NAEHE)) |>
  group_by(TC, TAG) |> # Jetzt kann auf Tag aggregiert werden, weil oben jedes Thema pro T
  summarise(SELEKT = n(), # Anzahl der Medien, die die jeweiligen Themen aufgegriffen hab
    Personalisierung = max(PERS_m), # Hier wird das Maximum des jeweiligen NF gen
    Kontroverse = max(KNTR_m),
    Schaden = max(SCHAD_m),
    Nähe = max(NAEHE_m)) |>
  mutate(across(everything(), ~replace(.x, is.na(.x), 0))) |>
  ungroup()

```

```

SELEKT |>
  select(SELEKT:Nähe) |>
  cor() |> # Hier wird wieder eine Tabelle mit kable produziert
  kableExtra::kable(caption = "Korrelationen zwischen NFs", digits = 2,
    booktabs = T,
    align = c('rrrrr'), # die 5 steht für 5 Variablen
    linesep = "", # ohne diese Zeile, macht knitr alle 5 Zeilen einen kleinen Abstand
    ## col.names = c("Anstellung", "Lokal", "Regional", "National", "Transnational", "
  ) |> # noch mehr Anpassungsmöglichkeiten für kable finden Sie bei google: r kabl
  kableExtra::kable_styling() |> # hier können Sie stylen, aber bleiben Sie seriös :
  kableExtra::footnote(general = "Hier könnten/sollten Sie einen allgemeinen Hinweis oder
    general_title = "", # damit Kable unten nicht "Note" schreibt
    threeparttable = T # damit längere Fussnoten umgebrochen werden
  )

```

Textbeispiel: Untersucht man, wie stark die Nachrichtenfaktoren die Selektion der Themen über die Medien hinweg steuern, können relativ starke Korrelationen festgestellt werden. Die Korrelationen zwischen der Personalisierung und der Auswahl durch mehrere Medien ist mit .38 am stärksten.

Alternativ und vielleicht schöner kann das in einem Korrelationsplot dargestellt werden:

```

Corr_SELEKT <- SELEKT |>
  select(SELEKT:Nähe) |>
  cor()

Corr_SELEKT_test <- SELEKT |>
  select(SELEKT:Nähe) |>
  corrrplot::cor.mtest(conf.level = .95)

corrrplot::corrplot(Corr_SELEKT, p.mat = Corr_SELEKT_test$p,
  type = "lower", method = "square",
  addCoef.col = 'black',
  insig = "pch", diag = FALSE ,
  tl.cex=0.8, pch.cex = 0.8, tl.srt = 30)

rm(SELEKT,Corr_SELEKT, Corr_SELEKT_test)

```

Textbeispiel: (wie oben mit Fokus auf die erste Spalte mit der Selektion)

## 8.7 Thematisierung in Medien und Publikumsresonanz

Wenn Sie eine Hypothese zum Zusammenhang zwischen der Häufigkeit eines Themas in den Medien und der Häufigkeit der Nennungen durch das Publikum haben oder zwischen der Stärke eines NF in der Berichterstattung und den Nennungen durch das Publikum, dann müssen Sie die Daten zunächst aggregieren, weil Ihre Analyseeinheit grösser ist als Ihre Kodiereinheit (CE). Das kommt daher, dass Sie nicht Eigenschaften von Beiträgen in Beziehung setzen wollen, sondern die Anzahl von Beiträgen der jeweiligen Themen in Verbindung setzen wollen mit den Nennungen des Publikums. Die Anzahl der Beiträge muss daher eine Aggregationsstufe über den Artikeln liegen.

```

DATEN_IA <- DATEN_IA |>
  mutate(Nennungen = rowSums(across(Nenn1:Nenn5), na.rm = TRUE)) # hier nochmal zur Sicherung

SELEKT <- DATEN_IA |>
  group_by(TC) |> # Hier wird anhand der Themenvariable TC gruppiert
  summarise(Them_n = n(), # Anzahl der Thematisierungen
            PERS_m = max(PERS), # Hier werden die höchsten Werte je NF genommen, die zu j
            KNTR_m = max(KNTR),
            SCHAD_m = max(SCHAD),
            NAEHE_m = max(NAEHE),
            Nenn_m = max(Nennungen)) |>
  mutate(across(everything(), ~replace(.x, is.na(.x), 0))) |> # hier noch die na durch 0
  ungroup()

SELEKT |>
  select(Them_n:Nenn_m) |>
  cor() |> # Hier wird wieder eine Tabelle mit kable produziert
  kableExtra::kable(caption = "Korrelationen zwischen Thematisierung und Nennungen durch
  booktabs = T,
  align = c('rrrrr'), # die 5 steht für 5 Variablen
  linesep = "", # ohne diese Zeile, macht knitr alle 5 Zeilen einen kleinen Abstand
  ## col.names = c("Them_n", "PERS_m", "KNTR_m", "SCHAD_m", "NAEHE_m", "Nenn_m"), #
  ) |> # noch mehr Anpassungsmöglichkeiten für kable finden Sie bei google: r kabl
  kableExtra::kable_styling() |> # hier können Sie stylen, aber bleiben Sie seriös:
  kableExtra::footnote(general = "Hier könnten/sollten Sie einen allgemeinen Hinweis oder
  general_title = "", # damit Kable unten nicht "Note" schreibt
  threeparttable = T # damit längere Fussnoten umgebrochen werden
  )

# auch hier können Sie natürlich die alternativen Correlationsdarstellungen nutzen

```

## 8.8 Zusammenspiel der Nachrichtenfaktoren

Das Zusammenspiel der Nachrichtenfaktoren lässt sich als Kreuztabellen oder als Korrelationen darstellen. Wenn sehr wenige Nachrichtenfaktoren verglichen werden sollen, geben die Kreuztabellen einen differenzierteren Einblick. Bei mehreren NFs ist es deutlich effizienter die Stärke der Zusammenhänge über Korrelationen abzubilden.

```

Corr_NF <- DATEN_IA |>
  select(PERS:NAEHE) |>
  as.matrix() |>
  cor()

Corr_NF_test <- DATEN_IA |>
  select(PERS:NAEHE) |>

```

```

corrplot::cor.mtest(conf.level = .95)

# mehr zu den corrplots hier: https://cran.r-project.org/web/packages/corrplot/vignettes/
corrplot::corrplot(Corr_NF, p.mat = Corr_NF_test$p,
  type = "lower", method = "square",
  insig = "pch", diag = FALSE ,
  tl.cex=0.6, pch.cex = 0.8, tl.srt = 45)

```

Textbeispiel: An den Korrelationen ist gut zu erkennen, dass manche Nachrichtenfaktoren eher zusammen auftreten, manche sich eher widersprechen und viele unabhängig voneinander sind («x» im Feld für fehlende Signifikanz). Eine starke positive Korrelation findet sich zwischen «Elite-Personen» (ELIT\_PERS) und «Prominenz» (PRO). Das ist sehr plausibel, weil Elitepersonen oft auch prominent sind. Auf der anderen Seite gibt es eine relativ starke negative Korrelation zwischen «gesellschaftlicher Relevanz» (RELE\_GESELL) und «Personalisierung» (PERS). Themen mit gesellschaftlicher Relevanz sind also oft nicht personengebunden sondern von allgemeinerer und damit auch abstrakterer Natur.

### 8.8.1 Netzwerkplot der Korrelationen

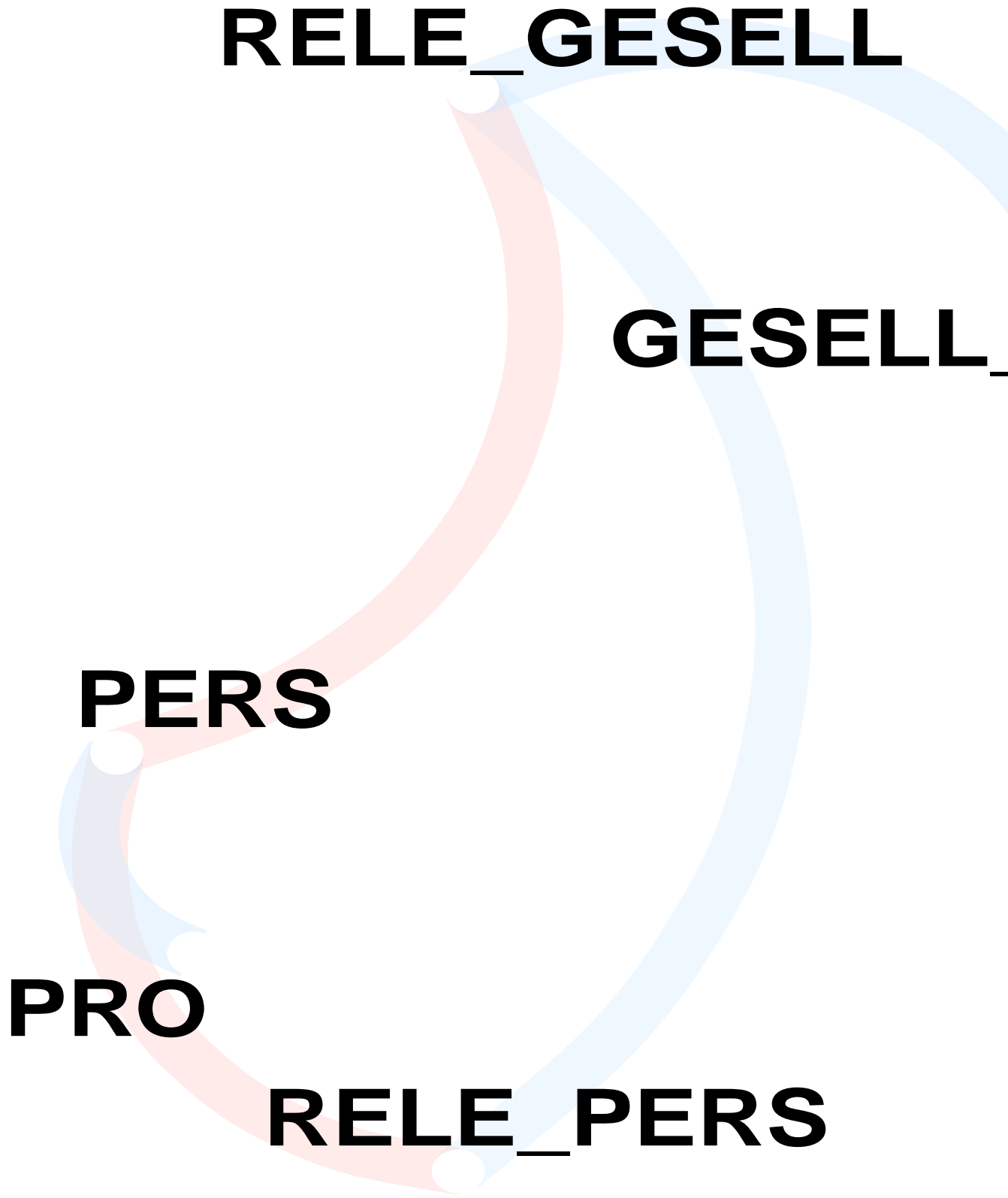
Es gibt auch die Möglichkeit, die Nachrichtenfaktoren in so einem Netzwerkdiagramm darzustellen. Damit kann man sehen, welche Nachrichtenfaktoren stärker zusammenhängen (nahe beieinander) und welche eher unabhängig sind (weiter voneinander entfernt).

```

Corr_NF <- DATEN_IA |>
  select(RELE_GESELL, RELE_PERS, PRO, PERS, KNTR, KONF, GESELL_SCHAD, GWLT, SCHAD) |>
  corrr::correlate(diagonal = NA) # oder 1, wenn Sie lieber die 1er in der Diagonalen haben
## Correlation computed with
## * Method: 'pearson'
## * Missing treated using: 'pairwise.complete.obs'

Corr_NF |>
  corrr::network_plot() # mehr zur Formatierung der network-plots erfahren Sie hier: http

```





Die näher beieinanderliegenden Nachrichtenfaktoren hängen stärker miteinander zusammen als die stärker entfernten. In diesem Netzwerkdiagramm ist gut zu erkennen, dass «Kontroverse» (KNTR), «Konflikt» (KONF), «Gewalt» (GWLTL) und «Schaden» (SCHAD) stärker miteinander zusammenhängen als mit «Personalisierung» (PERS) oder «gesellschaftlicher Relevanz» (RELE\_GESSELL). Auf der anderen Seite gibt es eine relativ starke negative Korrelation zwischen «gesellschaftlicher Relevanz» (RELE\_GESSELL) und «Personalisierung» (PERS).

## 8.9 Wirkung der NF auf die Beachtungsindikatoren

### 8.9.1 Regressionstabellen

```
Beacht_Rez_1 <- lm(Nenn_Gesamt ~ PERS + KNTR + GESELL_SCHAD + Population, DATEN_IA)

Beacht_Jour_1 <- lm(Jour_Beacht ~ PERS + KNTR + GESELL_SCHAD + Population, DATEN_IA)

# mehr zu tab_model finden Sie hier: https://strengjacke.github.io/sjPlot/articles/tab_model
# Der Output erscheint rechts im Viewer und kann dort kopiert und zB nach Word übertragen

sjPlot::tab_model(Beacht_Rez_1, Beacht_Jour_1, show.se = FALSE, show.std = TRUE, show.est = TRUE)

# Leider gibt es nur einen vollkommen unbrauchbaren Output im PDF also RMarkdown. Dafür gibt es
# Wer noch mehr Kontrolle über die Tabelle möchte, kann sich mit lm.beta, tidy und kable
Beacht_Rez_1_std <- lm.beta(Beacht_Rez_1)

# Damit können das R2 und die Freiheitsgrade aus den Summaries des Modells extrahiert und
R2 <- summary(Beacht_Rez_1_std)$r.squared
N <- summary(Beacht_Rez_1_std)$df[2]

tidy(Beacht_Rez_1_std) |>
  select(-estimate, -std.error) |>
kable(caption = "Regression auf Beachtung der Rezipienten", digits = 2,
      booktabs = T,
      align = c('lrrrrrrrrr'), # die 5 steht für 5 Variablen
      col.names = c("UVs", "BETAs", "t", "p"),
      linesep = "", # ohne diese Zeile, macht knitr alle 5 Zeilen einen kleinen Abstand
      ## col.names = c("Anstellung", "Lokal", "Regional", "National", "Transnational", "International")
      ) |> # noch mehr Anpassungsmöglichkeiten für kable finden Sie bei google: r kable
kable_styling() |> # hier können Sie stylen, aber bleiben Sie seriös :-)
footnote(general = paste0("$R^2$: ", round(R2, 2), " Freiheitsgrade: ", N, "\n Hier können Sie
      general_title = "", # damit Kable unten nicht "Note" schreibt
      threeparttable = T # damit längere Fussnoten umgebrochen werden
      )
```

Mit stargazer wird direkt LaTeX-Code erzeugt, der dann in den PDFs relativ schön angezeigt

wird. Stargazer hat sehr viele Funktionen und kann sehr genau angepasst werden. Im HTML, wie auf dieser Seite, ist der Output nicht ganz so schön:

Eine weitere Alternative ist jtools. Damit werden recht umfangreich Informationen über die Modelle inline in R-Studio ausgegeben und passabel im PDF rausgelassen. Nachteil ist die mässige bzw. umständliche Anpassbarkeit.

Mehr dazu finden Sie hier.

### 8.9.2 Regressionsdiagramm (Whyskerplot)

Tabellen sind einigermassen informativ. Besser lesbar ist allerdings die folgende Darstellung mit einem Whiskerplot, die auch sehr gut im PDF funktioniert, weil es eine Abbildung ist!

```
# mehr zu plot_models hier: https://strengjacke.github.io/sjPlot/articles/plot_model_est
sjPlot::plot_models(Beacht_Rez_1, Beacht_Jour_1,
  std.est = "std", # Für die (vergleichbaren) standardisierten Regressionskoeff
  title = "NFs und Beachtung",
  legend.title = "Beachtung",
  show.values = TRUE) +
ylim(-.25, .6) # müssen Sie vielleicht anpassen. Der erste Wert muss kleiner sein als d
```

Textbeispiel: «Personalisierung» wirkt auf die Beachtung durch Rezipienten nur minimal und eher negativ (-.06). Insgesamt nennen die Befragten also etwas seltener Meldungen, wo die Personalisierung hohe Werte hat. Allerdings liegt o im Konfidenzintervall dieses kleinen Effekts; er ist also nicht signifikant. Im Unterschied dazu besteht ein starker positiver Zusammenhang zwischen der Personalisierung und der journalistischen Beachtung (.36). Personalisierung ist also ein stark journalistischer Nachrichtenfaktor.

Der Nachrichtenfaktor «Kontroverse» zeigt kleine positive Zusammenhänge mit der Beachtung durch die Rezipienten (.04) und mit der journalistischen Beachtung (.09). Allerdings ist die Wirkung auf die Rezipientenerinnerung nicht signifikant. Das Gewicht des Nachrichtenfaktors «Kontroverse» in Bezug auf die journalistische Beachtung ist signifikant von 0 verschieden.

## 8.10 Vergleich der NF-Gewichte nach Mediengruppen

Hier werden die NF-Gewichte nach Mediengruppe verglichen.

```
# mehr zu plot_models hier: https://strengjacke.github.io/sjPlot/articles/plot_model_est
Beacht_Quali_1 <- lm(Nenn_Gesamt ~ PERS + KNTR + GESELL_SCHAD + Population, subset(DATEN_
Beacht_Boulevard_1 <- lm(Nenn_Gesamt ~ PERS + KNTR + GESELL_SCHAD + Population, subset(DA
sjPlot::plot_models(Beacht_Quali_1, Beacht_Boulevard_1,
  std.est = "std", # Für die (vergleichbaren) standardisierten Regressionskoeff
  title = "Gewichte der NFs nach Medientypen",
  legend.title = "Beachtung",
  m.labels = c("Qualitätsmedien", "Boulevard"),
  axis.title = "Regressionsgewichte",
```

```
show.values = TRUE) +
ylim(-.25, .6)
```

## 8.11 Unterschiede der Gewichte von NF nach Befragtenmerkmalen

Zu diesen Auswertungen und den Voraussetzungen in der Datenaufbereitung habe ich noch ein Video erstellt:

### 8.11.1 Zwei Gruppen als UV

Sie können Unterschiede in der Bedeutung (Gewicht) der Nachrichtenfaktoren mit einem Unterschiedstest untersuchen. Dazu müssen Sie die NF aus der Inhaltsanalyse an die Befragung matchen (siehe [@ref\(IAanBefragung\)](#)).

```
#DATEN_BF <- readRDS("data/BEFuIA.RDS")

t_test <- t.test(KNTR_m ~ Geschlecht, DATEN_BF) # t-Test ausführen und in t_test schreiben

t_Wert <- round(t_test$statistic,2)
p_Wert <- round(t_test$p.value, 3)
df <- round(t_test$parameter,0)

### Hier werden die Werte aus dem t-Test in einen Text geschrieben, den Sie in einer Grafik
t_Test_Geschlecht_KNTR = paste0("t-Test: \n",
  "p-Wert = ", p_Wert, # den p-Wert aus der t_test-List-Ausgabe
  "\n (t = ", t_Wert, # den T-Wert (statistic) aus der t_test-List-Ausgabe
  ", df = ", df, ")") # und die df (parameter) aus der t_test-List-Ausgabe

df.summary <- DATEN_BF %>% ## Daten der Befragung mit Ihren angehängten NF
  select(Geschlecht, KNTR_m) |>
  group_by(Geschlecht) |>
  summarise(
    ci_d = 1.96*sd(KNTR_m, na.rm = TRUE)/sqrt(n()),
    KNTR_m = mean(KNTR_m, na.rm = TRUE))

df.summary |>
  ggplot(aes(x = Geschlecht, y = KNTR_m, ymin = KNTR_m-ci_d, ymax = KNTR_m+ci_d, color = Geschlecht)) +
  geom_errorbar(width = 0.2) +
  geom_point(size = 1.5) +
  labs(x = "Geschlecht", y = "KNTR_m",
    title = "Bedeutung des NF Tierbezug nach Geschlecht",
    subtitle = "Mittelwertvergleich mit Konfidenzintervall") +
  theme(legend.position = "none") +
  annotate("text", x = Inf, y = Inf, label = t_Test_Geschlecht_KNTR, vjust = 1, hjust = 1)
```

```

theme_minimal()

#
# DATEN_BF %>% ## Daten der Befragung mit Ihren angehängten NF
#   as_label %>% ## die Label für die Grafik verwenden, statt der Werte ("männlich", "weiblich")
#   ggpubr::ggerrorplot(., x = "Geschlecht", y = "KNTR_m", # die UV ist x und die AV ist y
#     desc_stat = "mean_ci", color = "Geschlecht", # zeigt die Konfidenzintervalle um
#     palette = "uchicago", size = .9) + # bei den Paletten können Sie sich umsch
#   labs(x = "Geschlecht", y = "Kontroverse",
#     title = "Bedeutung des NF Kontroverse nach Geschlecht",
#     subtitle = "Mittelwertvergleich mit Konfidenzintervall") +
#   theme(legend.position = "none") +
#   annotate("text", x = Inf, y = Inf, label = t_Test_Geschlecht_KNTR, vjust = 1, hjust = 1)

rm(t_Wert, p_Wert, df)
rm(t_Test_Geschlecht_KNTR)

```

### 8.11.2 Mehrere Gruppen als UV

Hier werden die Mittelwerte für den Nachrichtenfaktor KNTR (Kontroverse) nach Altersgruppen verglichen (gebaut wurde KNTR\_m in der Datenaufbereitung siehe @ref(IAanBefragung)). Sie können schauen, ob die Mittelwerte sich zwischen den Gruppen unterscheiden, wenn Sie die Konfidenzintervalle (KI) anschauen. Wenn die KIs sich nicht überschneiden, dann sind die Unterschiede signifikant.

```

# Erstmal eine Analysis Of Variance = ANOVA für die Variable KNTR_m nach Altersgruppen (ANOVA)
ANOVA <- aov(KNTR_m ~ Alter, data = DATEN_BF)

# summary(ANOVA)
p_Wert <- round(summary(ANOVA)[[1]][1,5],3) # hier wird der p-Wert aus der Summary der ANOVA
F_Wert <- round(summary(ANOVA)[[1]][1,4],1) # hier der F-Wert
df <- round(summary(ANOVA)[[1]][2,1],0) # hier die df

### Hier werden die Werte aus dem F-Test in einen Text geschrieben, den Sie in einer Grafik
ANOVA_Alter_KNTR = paste0("F-Test: \n",
  "p-Wert = ", p_Wert, # den p-Wert aus der t_test-List-Ausgabe
  "\n (F = ", F_Wert, # den F-Wert (statistic) aus der t_test-List-Ausgabe
  ", df = ", df, ")") # und die df (parameter) aus der t_test-List-Ausgabe

DATEN_BF %>% ## Daten der Befragung mit Ihren angehängten NF
  sjlabelled::as_label() %>% ## die Label für die Grafik verwenden, statt der Werte ("männlich", "weiblich")
  ggpubr::ggerrorplot(., x = "Alter", y = "KNTR_m", # die UV ist x und die AV ist y (#Met
    desc_stat = "mean_ci", color = "Alter", # zeigt die Konfidenzintervalle um
    palette = "uchicago", size = .9) + # bei den Paletten können Sie sich umsch
  labs(x = "Alter", y = "Kontroverse",
    title = "Bedeutung des NF Kontroverse nach Alter",
    subtitle = "Mittelwertvergleich mit Konfidenzintervall") +

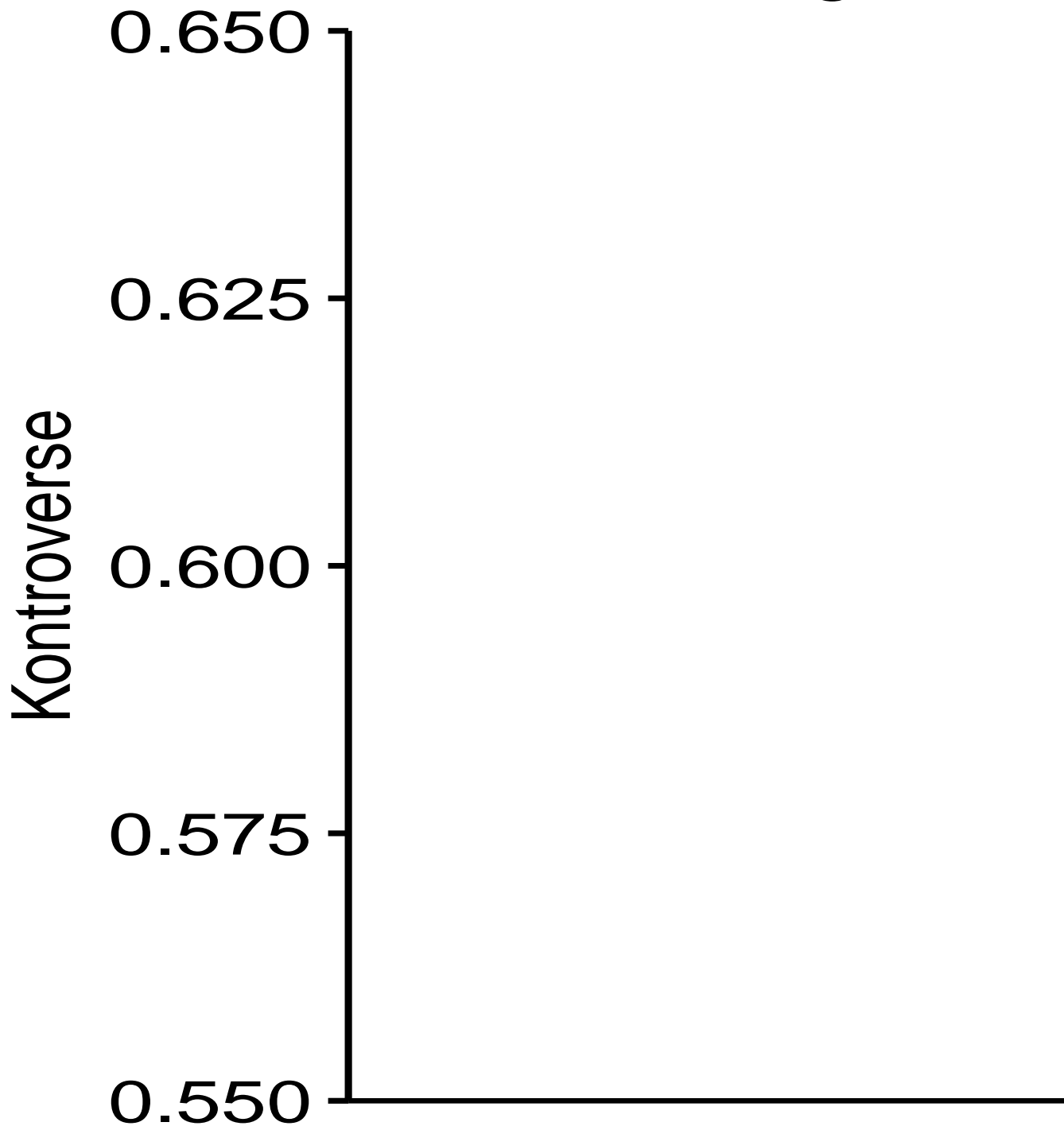
```

```
# theme(legend.position = "none") + # mit oder ohne Legende ok lesbar, was Sie schöner f
  annotate("text", x = Inf, y = .6, label = ANOVA_Alter_KNTR, vjust = 1, hjust = 1) # hier
## Warning: Computation failed in `stat_summary()`
## Caused by error in `get()`:
## ! object 'mean_ci' of mode 'function' was not found
## Warning: Position guide is perpendicular to the intended axis
## i Did you mean to specify a different guide `position`?

rm(F_Wert, p_Wert, df)
rm(ANOVA_Alter_KNTR)
```

# Bedeutung des N

## Mittelwertvergleich



## 9 Forschungsbericht mit Quarto/Word

In der modernen quantitativen Forschung arbeiten die meisten Kollegen mit R (oder Python, aber die sind komisch). Mehr und mehr schreiben ihre Forschungsberichte in R-Markdown beziehungsweise neuerdings Quarto, was (fast) dasselbe ist. Wenn sie das tun, dann können sie Ihre Arbeit später als Word-Datei rauslassen, als HTML (um zum Beispiel eine Onlinedokumentation oder Onlineanhänge zu Ihren Arbeiten bereitzustellen) und als PDF um sie verteilen oder auch drucken zu können. Die PDF basiert auf einem der ausgereiftesten Drucksatzsysteme, das auch noch kostenlos ist: LaTeX. Dieses Drucksatzsystem ist kompliziert, aber Markdown nicht. Markdown wurde entwickelt, damit man Webseiten einfach erstellen kann und die Texte auch im Rohzustand noch lesbar sind. Darum ist es viel einfacher Markdown zu verwenden als Word, HTML, LaTeX oder gleich PostScript. Quarto ist eine Weiterentwicklung von Markdown für wissenschaftliche Texte. Dieses System ermöglicht Open Source Science und Publishing.

Wissenschaftliches Arbeiten bedeutet auch immer zitieren. Die Zitationen können natürlich irgendwie händisch eingebaut und später in einem Literaturverzeichnis ausgeführt werden. Das ist allerdings 1. aufwendig, 2. fehleranfällig und 3. sehr unflexibel (weil man zum Beispiel alles ändern müsste, statt nur einen Zitationsstil anzupassen.) Für Zitationen in RMarkdown (und Quarto) kann das Format BibTeX verwendet werden. Das ist schon vor langer Zeit für TeX und LaTeX entwickelt worden und ist sehr verbreitet (sie finden fast immer für Zitationen die sogenannten Bib-Tex-Schlüssel). Ein kostenloser Editor, für den es für alle Systeme und Online ein Frontend gibt ist Zotero (Citavi ist auch ok, aber nicht überall kostenlos und darum eigentlich nicht so verbreitet). Dort können Sie einfach ISBN-Nummern oder auch DOIs eingeben, um sämtliche Zitationsangaben abrufen zu können. Sie müssen die also nicht selbst eingeben (können aber auch das tun). Zotero lässt sich gut in RStudio integrieren (Tools -> Project Options -> R Markdown -> Visual Mode Zotero). Hier gibt es eine Einführung in Zotero, die zwar mehr auf Word gemünzt ist, aber gut. Im Youtube-Video unten von Ulrik Lyngs wiederum finden Sie eine komplette Zusammenfassung unseres ganzen Workflows mal von einem anderen Typen. Sie erkennen schnell, dass das Setup, das ich Ihnen ans Herz legen möchte, nicht nur eine absonderliche Marotte meinerseits ist, sondern State of the Art wissenschaftlichen Publizierens, das Sie ja im Studium lernen sollten. :-)

```
vembedr::embed_youtube("aU0uetWJXis") |>
  vembedr::use_rounded()
```

### 9.1 Quarto ist besser und einfacher

Zu Quarto zwingen Sie nicht, aber die hier gezeigte Methode zur Erstellung von Hausarbeiten, Forschungsberichten und sonstigen Texten ist schon sehr leistungsstark, weil Sie nicht nur tolle Berichte in PDF produzieren können, sondern Ihren Bericht auch gleich in HTML rauslassen können oder auch als ebook oder auch als Word (mit Einschränkungen). Wenn Sie Ihren Bericht in Quarto verfassen, dann bekommen Sie am Ende des Semesters mit Ihren Task-Force-Zeugnissen eine Bestätigung, dass Sie auch dieses Tool beherrschen. Quarto-Markdown zu beherrschen ist eine Qualifikation, die Sie sich gut in Ihr CV schreiben können. Auf dieser Seite <https://quarto.org> finden Sie einige

Bücher zu bookdown, die alle auch in bookdwon verfasst wurden. Hier <https://quarto.org> finden Sie Lösungen zu den meisten Fragen zu Quarto.

Diese R-Introduction ist übrigens auch in bookdown verfasst.

```
vembedr::embed_youtube("7_WjjClQGUw") |>  
  vembedr::use_rounded()
```

## 9.2 Tabellen und Grafiken nach Word schaffen

Es ist eine Herausforderung, Berichte in Markdown bzw. bookdown zu setzen und (wie alles in R) braucht es viel Zeit. Die Alternative besteht in vertrauteren Textverarbeitungsprogrammen wie zB Word, Page oder Open-Office. Berichte mit Titelblatt, Inhalts-, Abbildungs- sowie Tabellenverzeichnis, automatisch beschrifteten Grafiken und Tabellen sowie Querverweisen und einem ordentlichen Literaturverzeichnis hinzubekommen ist allerdings auch nicht einfach, wenn man diese Textverarbeitungsprogramme benutzt. Ganz zu schweigen von schwer kontrollierbaren Platzierungen von Grafiken und Tabellen. Die fliegen einem in solchen Programmen schnell um die Ohren.

Alle Bedenken beiseite, gab es den Wunsch, dass auch gezeigt wird, wie man die Grafiken aus R in hinreichender Qualität nach Word rüberschafft (in anderen Programmen funktioniert es vergleichbar; Word nehme ich hier als das gängigste Beispiel). Screenshots weisen keine akzeptable Qualität auf und sind deshalb keine Lösung – vor allem nicht für Tabellen.

Tabellen können schlicht in Wordtabellen abgeschrieben werden, was allerdings zu Übertragungsfehlern führt. Die Tabellen können aber auch schlicht kopiert und nach Word übertragen werden. Dort müssen Sie allerdings noch in eine annehmbare Form gebracht werden, was mit Tabellenformatierung oder Tabellenvorlagen einigermassen funktioniert. Die Tabellenüberschriften werden mit «Beschriftung einfügen» gemacht.

Noch komplizierter ist die Übertragung von Diagrammen und anderen Bildformaten aus R nach Word. Eine Möglichkeit bietet ggplot2, da bei ggplot-Grafiken mit ggsave die Grafiken im pdf-Format oder als png gespeichert werden können (wo möglich, ist pdf deutlich besser). Alternativ kann der «Visual» Mode im RStudio-Editor verwendet und dort Grafiken per drag-and-drop direkt nach Word rübergezogen werden. Die verschiedenen Methoden zeige ich hier im Video.

```
vembedr::embed_youtube("Ogy2bIfJpnk") |>  
  vembedr::use_rounded()
```