R - Befehlsliste

Thomas Haase

17. November 2022

FB03 - Sozialwissenschaften Matrikelnummer: 6033199

Modul: Übung Einstieg in R leicht gemacht - Philipp Kleer

Inhaltsverzeichnis

1 Grundlagen **1**
- 1.1 Environment . . . 1
- 1.2 Vektoren . . . 1
- 1.3 Faktoren . . . 1
- 1.4 Dataframes . . . 1
  - 1.4.1 Logische Verknüpfungen . . . 2

2 Importe, deskript. Statistik, Tidyverse **3**
- 2.1 Datenimport . . . 3
  - 2.1.1 .txt Datei . . . 3
  - 2.1.2 .csv Datei . . . 3
  - 2.1.3 R Datei Import . . . 3
  - 2.1.4 SPSS Import . . . 3
- 2.2 Deskriptive Statistik . . . 4
  - 2.2.1 Dimensionen und Länge . . . 4
  - 2.2.2 Minimum, Maximum, Spannweite . . . 4
  - 2.2.3 Modus, Median, arithm. Mittelwert . . . 5
  - 2.2.4 Quantile . . . 5
  - 2.2.5 Streuungsmaße . . . 5
  - 2.2.6 Standardabweichung und Varianz . . . 5
- 2.3 Tidyverse . . . 6
  - 2.3.1 Übersicht Häufiger Funktionen . . . 6
    - 2.3.1.1 select() - Spalten wählen . . . 6
    - 2.3.1.2 slice () - Zeilen wählen . . . 6
    - 2.3.1.3 filter() - Zeilen filtern . . . 6
    - 2.3.1.4 arrange() - Spalte Sortieren . . . 6
    - 2.3.1.5 mutate() - Variable bearbeiten . . . 6
    - 2.3.1.6 case_when() + mutate() . . . 7
    - 2.3.1.7 summarize() - Zusammenfassen . . . 7
    - 2.3.1.8 group_by() - gruppieren . . . 8
  - 2.3.2 Piping . . . 8
  - 2.3.3 Teilen eines Datensatzes NICHT EINSEHBAR IM BUCH . . . 8
  - 2.3.4 Datensätze zusammenführen - Fälle hinzufügen . . . 8
    - 2.3.4.1 bind_rows() - gleiche Variablen . . . 8
    - 2.3.4.2 full_join() - unterschiedliche Variablen . . . 8
  - 2.3.5 left_join(), right_join() - Datensätze kombinieren . . . 9
  - 2.3.6 rename() + fulljoin() - Variable hinzufuegen . . . 9
- 2.4 tidyR . . . 10
  - 2.4.1 Beispiel statclass . . . 10
  - 2.4.2 Beispiel statclass2 . . . 11
- 2.5 if, else, elseif . . . 12
- 2.6 for-loop . . . 12
- 2.7 function() . . . 13

3 Zusammenhangsmaße und Mittelwertvergleiche **15**
- 3.1 Zusammenhangsmaße . . . 15
  - 3.1.1 Tabellen . . . 15
  - 3.1.2 Kreuztabellen . . . 15
    - 3.1.2.1 Randhäufigkeiten . . . 16

3.1.2.2 gmodels . . . 16
3.1.3 Zusammenhangsmaße . . . 16
3.1.3.1 Chi2 Unabhängigkeitstest . . . 16
3.1.3.2 Annahmen . . . 16
3.1.3.3 Cramer's V . . . 17
3.1.4 Korrelationen berechnen . . . 17
3.1.4.1 Korrelationskoeffizient berechnen . . . 18
3.1.4.2 mehrere Korrelationen berechnen . . . 18
3.1.5 Grafiken . . . 19
3.1.5.1 Library psych - Korrelationsübersicht . . . 19
3.1.5.2 Library corrplot - Heatmap . . . 19
3.2 Mittelwertvergleiche . . . 19
3.2.1 Ein-Stichproben-T-Test . . . 19
3.2.1.1 zweiseitiger Test . . . 19
3.2.1.2 einseitiger Test . . . 20
3.2.2 Zwei-stichproben-T-Testf) ge . . . 20
3.2.2.1 ungepaart . . . 20
3.2.2.2 gepaart . . . 22
3.2.3 ANOVA/F-Test - mehr als 2 Gruppen . . . 22

4 Lineare Regression, Modellexport, Regressionsdiagnostik 23
4.1 Lineare Regression . . . 23
4.1.1 Bivariates Beispiel . . . 23
4.1.1.1 Koeffizienten Interpretation . . . 24
4.1.1.2 weitere Ausgaben . . . 24
4.1.2 Multivariates Beispiel . . . 25
4.1.3 Dichotome Kategorielle Variablen . . . 25
4.1.4 Polytome Kategorielle Variablen . . . 26
4.1.4.1 Ändern der Referenzkategorie . . . 27
4.2 Standardisierte Koeffizienten und Export . . . 28
4.2.1 Z-Standardisierung . . . 28
4.2.2 Export Tabellen und Grafiken . . . 28
4.2.2.1 Modelle in Textdateien exportieren . . . 28
4.3 Regressionsdiagnostik . . . 32
4.3.1 Annahme 1: lineare Beziehung . . . 33

5 Darstellungen 33
5.1 Plots . . . 33
5.1.1 Balkendiagramme . . . 33
5.1.1.1 Säulen und Balkendiagramme . . . 33
5.1.1.2 Farben . . . 34
5.1.1.3 Wertebeschriftungen . . . 35
5.1.1.4 Achsen, Legende, Titel . . . 36
5.1.1.5 Speichern . . . 37
5.1.1.6 Gruppieren . . . 37
5.1.2 Histogramme . . . 37
5.1.2.1 Start . . . 37
5.1.2.2 Gruppieren . . . 38
5.1.2.3 Ridgeline . . . 38
5.1.3 Boxplots . . . 39
5.1.3.1 geomboxplot() . . . 39

5.1.3.2 ggboxplot() 39
5.1.3.3 Rainclouds (are better boxplots) 40

1 Grundlagen

1.1 Environment

```
rm(datei1, dataframe2) #loescht Objekte aus dem Environment
```

1.2 Vektoren

```
logi <- c(TRUE, FALSE)
numbers <- c(1,2,3,4,5,6,7,8,9,10)
char <- c("Taipeh", "Seoul","Berlin","Taipeh")

char[3]
##[1] "Berlin"
```

1.3 Faktoren

Faktoren sind Vektoren, in denen jede Koordinate nur ein Mal vorkommt. Die Koordinaten sind ggf. geordnet.

```
char <- c("Taipeh", "Seoul","Berlin","Taipeh")
as.factor(char)
## [1] Taipeh Seoul Berlin Taipeh
## Levels: Berlin Seoul Taipeh



vector <- c("x","y","z","x")
vectorOrd <- factor(vector, #Dateninput
ordered = TRUE, #Soll geordnet werden?
levels = ("z","y","x") #Wie wird geordnet? z < y < x
)
vectorOrd
## [1] x y z y
## Levels: z < y < x
```

1.4 Dataframes

Ein Dataframe ist eine Matrix mit Spalten(Vektoren 1.2) und Zeilen.
Mehrere Vektoren können zu einem Dataframe verbunden werden:

```
v1 <- c("x","y","z")
v2 <_ c(1,2,3)

binded <-cbind(v1,v2)
binded
##          v1       v2
## [1,]      x         1
## [2,]      y         2
## [3,]      z         3

df <- data.frame(binded)
df
```

```
##       v1       v2
## 1      x        1
## 2      y        2
## 3      z        3
```

Ausgabe von Daten "df [*Zeile* , *Spalte*]"

```
df[1,1]
## [1] x

df[1, ]
##       v1       v2
## 1      x        1

df[, 2]
## [1] 1 2 3

df[  ,c(1,2,3)] #oder variablenbezeichnung der Spalte eintragen
df[  ,c("auswahl","preis","...")]

#Zeile 3 bis 5
df[3:5, ]
```

Umgang mit einem Dataframe:
structure ausgeben

```
str(df)
```

erste/letzte 6 Zeilen ausgeben

```
head(df)
tail(df)
```

bestimmte Variable aufrufen

```
df$variable
```

Datentabelle einer Variablen ausgeben

```
table(df$variable)
```

1.4.1 Logische Verknüpfungen

und	&
oder	|
gleich	==
ungleich	!=
größer	>
kleiner	<
kleiner gleich	<=
größer gleich	>=

2 Importe, deskript. Statistik, Tidyverse

2.1 Datenimport

2.1.1 .txt Datei

In Klammer angeben:

- Pfad
- Trennzeichen (Im Beispiel mit Tab getrennt)
- Header = Variablennamen importieren

```
df <- read.table("./data/MeineDatei.txt",
sep = "\t",
header = TRUE
)
```

2.1.2 .csv Datei

Es wird zwischen .csv und .csv2 unterschieden.

Argument	read.csv	read.csv2	Beschreibung
header	TRUE	TRUE	Variablenbezeichnungen mit Importieren
sep	","	";"	Seperator Komma oder semicolon
dec	"."	","	Dezimalstelle mit Komma oder Punkt getrennt
fill	TRUE	TRUE	Wenn die Zeilen nicht gleichlang sind, werden Blanks hinzugefügt

```
dfcsv <- read.csv("./data/MeineDatei.csv",
header = TRUE
)

dfcsv2 <- read.csv2("./data/MeineDatei.csv",
header = TRUE
)
```

2.1.3 R Datei Import

.rds Dateien beinhalten einzelne Objekte.

```
df <- readRDS("./data/MeineDatei.rds")
```

alternativ den gesamten Workspace laden

```
load("./data/exam.RData")
```

2.1.4 SPSS Import

Um SPSS Dateien zu importieren wird eine library benötigt. In diesem Beispiel wird "foreign"verwendet. read.spss hat neue Argumente:

- use.value.labels = FALSE –> schaltet aus, dass Wertelabels als R-factors mit entsprechenden leveln übernommen werden
- to.data.frame –> logical: return a data frame?

```
install.packages("foreign")
library("foreign")

dfspss <- read.spss("./data/MeineDatei.sav",
use.value.labels = FALSE,
to.data.frame = TRUE
)
```

2.2 Deskriptive Statistik

Für eine schnelle Übersicht ein paar deskriptiver Werte summary()

```
summary(df$variable)
```

2.2.1 Dimensionen und Länge

Länge eines Vektors und Dimensionen

```
length(df$variable)

dim(df)
```

2.2.2 Minimum, Maximum, Spannweite

Minimum

```
min(df$variable)

#ohne NA's (missing values)
min(df$variable,
na.rm = TRUE)
```

Maximum

```
max(df$variable)

#ohne NA's (missing values)
max(df$variable,
na.rm = TRUE)
```

Spannweite (gibt Minimum und Maximum aus)

```
range(df$variable)

#ohne NA's (missing values)
range(df$variable,
na.rm = TRUE)
```

2.2.3 Modus, Median, arithm. Mittelwert

Median (Der Wert, der genau in der Mitte einer Datenverteilung liegt.)

```
median(df$variable,
na.rm = TRUE)
```

Arithmetischer Mittelwert

```
mean(df$variable,
na.rm = TRUE)
```

Modus (Der häufigste Wert, der in der Stichprobe vorkommt.) Keine built-in-function ==> nur mit Workaround möglich.
Man lässt sich alle Werte ausgeben und filtert dann daraus den höchsten Wert. table() ist in 1.4 erklärt

```
max(table(df$variable))
```

2.2.4 Quantile

Mit Quantisierung gibt man Werte in bestimmten Bereichen aus.

```
seq(from = 0, to = 1, by = 0.1)

seq(0, 1, 0.1)

quantile(df$variable,
probs = seq(0,1, 0.1),
na.rm = TRUE
)
```

2.2.5 Streuungsmaße

Interquartilsabstand - Die breite des Intervalls, in dem die mittleren 50 Prozent der Stichprobeelemente liegen.

```
IQR(df$variable,
na.rm = TRUE
)
```

Alternativ (mit 2.2.4):

```
diff(quantile(df$variable,
probs = c(0.25, 0.75),
na.rm = TRUE
)
)
```

2.2.6 Standardabweichung und Varianz

Standardabweichung

```
sd(df$variable,
na.rm = TRUE
)
```

Varianz

```
var(df$variable,
na.rm = TRUE
)
```

2.3 Tidyverse

```
install.packages("tidyverse", dependencies = TRUE)
library("tidyverse")
```

2.3.1 Übersicht Häufiger Funktionen

2.3.1.1 select() - Spalten wählen

Spalten auswählen. Ausgabe erfolgt in tibble.

```
select(df, c(variable1, variable2))
```

2.3.1.2 slice () - Zeilen wählen

Zeilen auswählen.

```
slice(df, 50:55)

slice(df, seq(0,1000,100))
```

2.3.1.3 filter() - Zeilen filtern

Datensatz unter bestimmten Bedingungen ausgeben.
Filtern mit logischen Verknüpfungen 1.4.1

```
filter(df, variable == "Variable 1")

filter(df, variable == "Variable 1" & gndr == "male")
```

2.3.1.4 arrange() - Spalte Sortieren

Mit arrange() können Variablenwerte aufsteigend oder absteigend sortiert werden.

```
#aufsteigend
dfAsc <- arrange(df, variable)

#absteigend
dfDesc <- arrange(df, -variable)
dfDesc <- arrange(df, desc(variable))
```

2.3.1.5 mutate() - Variable bearbeiten

mutate() kann zum erstellen neuer Variablen, oder zum bearbeiten vorhandener Variablen verwendet werden.

```
mutate(df, variable2 = variable1 - mean(df,variable1))
```

2.3.1.6 case_when() + mutate()

case_when() beinhaltet vektorisierte if else() Statements. Es ist äquivalent zum CASE WHEN in SQL.

```
#hier wird fuer bestimmt-teilbare Zahlen ein Wort, statt der Zahl im Datensatz
    eingetragen
x <- 1:50
case_when(
x %% 35 == 0 ~ "fizz buzz",
x %% 5 == 0 ~ "fizz",
x %% 7 == 0 ~ "buzz",
TRUE ~ as.character(x)
)


#hier wird die Variable district recoded und aus der char variable eine integer
    variable gemacht.
mutate(df,districtRec = case_when(
district == "Distrikt 1" ~ 1,
district == "Distrikt 5" ~ 5,
district == "Distrikt 7" ~ 7,
district == "Distrikt 10" ~ 10,
district == "Distrikt 12" ~ 12,
)
)

#hier wird getestet, ob eine Person im Distrikt12 weiblich ist. Alle Variablen in
    denen die Bedingungen gelten werden mit 1 gekennzeichnet. Damit nicht alle
    anderen Kombinationen aufgeschrieben werden, werden mit TRUE alle anderen Zeilen
    mit einer 0 gekennzeichnet.

mutate(df, d12gndr = case_when(
district == "Distrikt12" & gndr == "female" ~ 1,
TRUE ~ 0))
```

2.3.1.7 summarize() - Zusammenfassen

Werte aus Spalten werden zusammengefasst. Nutzbar ist jede Funktion, die eine Spalte als Input verlangt.

```
summarize(df,first(variable))
summarize(df,last(variable))
summarize(df,nth(variable))
summarize(df,n(variable))
summarize(df,n_distinct(variable))
summarize(df,IQR(variable))
summarize(df,min(variable))
summarize(df,max(variable))
summarize(df,mean(variable))
summarize(df,median(variable))
summarize(df,var(variable))
summarize(df,sd(variable))
```

2.3.1.8 group_by() - gruppieren

Mit group()_by() wird die Ausgabe nach den Variablentypen gemacht.

```
group_by(df, variable1)

#mean(variable1) fuer m und f ausgeben
summarize(group_by(df,gndr), mean(variable1))
```

Die Gruppierung kann mit ungroup() entfernt werden

```
ungroup()
```

2.3.2 Piping

Piping übergibt den Datensatz an die entsprechenden Funktionen.
Der Hotkey ist STRG + SHIFT + M.

```
df <- df %>%
group_by(variable1) %>%
mutate(variable3 = mean(variable2, na.rm = TRUE)) %>%
ungroup()
```

LINKS: group_by() - gruppieren, mutate() - Variable bearbeiten

2.3.3 Teilen eines Datensatzes NICHT EINSEHBAR IM BUCH

2.3.4 Datensätze zusammenführen - Fälle hinzufügen

2.3.4.1 bind_rows() - gleiche Variablen

Für Datensätze mit gleicher Variablenanzahl.
.id = origin erstellt eine Variable, die die Herkunft eines Falles erfasst

```
dfALL <- df1 %>%
bind_rows(list(df2, df3, df4, df5)), .id = "origin") %>%
mutate(origin = factor(origin, labels = c(
"Beschreibung1", "Beschreibung 2", "Beschreibung 3", "Beschreibung 4", "Beschreibung
    5")))

table(dfALL$origin)
head(dfALL$origin)
```

2.3.4.2 full_join() - unterschiedliche Variablen

Datensätze mit Unterschiedlich benannten Variablen können mit bindrows() oder full_join() verbunden werden.

```
#Datensatz 1(df1)----------------
##    district gndr
## 1 Distrikt 1 male
## 2 Distrikt 1 male
## 3 Distrikt 1 male

#Datensatz 2(df2)----------------
##    District  gndr
## 1 Distrikt 1 male
```

```
## 2 Distrikt 5 female
## 3 Distrikt 7 female

dfTest <- df1 %>%
bind_rows(df2)

dfTest
##    district  gndr  District
## 1 Distrikt 1 male      <NA>
## 2 Distrikt 1 male      <NA>
## 3 Distrikt 1 male      <NA>
## 4      <NA>  male Distrikt 1
## 5      <NA> female Distrikt 5
## 6      <NA> female Distrikt 7
```

Jetzt mit full_join()

```
dfTest2 <- df1 %>%
full_join(df2,by = c(
"district" = "District",
"gndr" = "gndr")
)

head(dfTest2)

##    district  gndr
## 1 Distrikt 1 male
## 2 Distrikt 1 male
## 3 Distrikt 1 male
## 4 Distrikt 5 female
## 5 Distrikt 7 female
```

2.3.5 left_join(), right_join() - Datensätze kombinieren

BEISPIEL UNVOLLSTAENDIG Hinzufügen eines anderen Datensatzes zum ursprünglichen Datensatz.

```
dfMerged <- df %>%
left_join(dfEXTRA, by = "district")

dfMerged2 <- dfEXTRA %>%
right_join(df, by = "district")
```

2.3.6 rename() + fulljoin() - Variable hinzufuegen

Im Beispiel befinden sich 2 Datensaetze. Beide haben eine ID, aber die restlichen Variablen unterscheiden sich. Beide sollen zusammengefuehrt werden, aber die ID Variable ist unterschiedlich benannt.

```
head(sp)
##   socialpoints   id
## 1     88.00907 10000
## 2     90.38817 10001
## 3     88.43383 10002

head(df)
```

```
##    idno  district  gndr
## 1 10000 Distrikt 1 male
## 2 10001 Distrikt 1 male
## 3 10002 Distrikt 1 female
```

Nun wird in sp die Variable id umbenannt.
rename(neuer Name = alter Name)

```
sp <- sp %>%
rename(idno = id)
```

Jetzt können beide Datensaetze mit full_join 2.3.4.2 verbunden werden.

```
pss <- pss %>%
full_join(sp, by = "idno")
```

2.4 tidyR

Ein Datensatz ist *tidy* = im *tidy*Format = im *long*Format, wenn...

- ... jede Variable eine Spalte ist.
- ... jede Beobachtung eine Zeile ist.
- ... jede Beobachtungseinheit eine Tabelle formt.

2.4.1 Beispiel statclass

Im folgenden wird der Datensatz einer Statistikklasse aufgeräumt.

```
statclass
##     name stat1 stat2 r spss
## 1   momo    12     5  6    9
## 2    kim    14    10 13   15
## 3 sascha     7     4  4    1
```

Mit pivot_longer() werden die Spalten angegeben, welche neu geordnet werden sollen(hier stat1 bis spss).
Mit names_to() und values_to() wird festgelegt in welche neuen Variablen die ursprünglichen Variablennamen und Variablenwerte gespeichert werden sollen.

```
statclassTidy <- statclass %>%
pivot_longer(stat1:spss, names_to = "course", values_to = "grade") %>%
arrange(name,course)

statclassTidy
## # A tibble: 12 x 3
##    name   course grade
##    <chr>  <chr>  <dbl>
##  1 kim    r         13
##  2 kim    spss      15
##  3 kim    stat1     14
##  4 kim    stat2     10
##  5 momo   r          6
##  6 momo   spss       9
##  7 momo   stat1     12
```

```
##  8 momo  stat2     5
##  9 sascha r        4
## 10 sascha spss     1
## 11 sascha stat1    7
## 12 sascha stat2    4
```

Die Sortierung hat den Nachteil, dass keine Mittelwerte mehr von grade berechnet werden können. Um dies umzukehren benutzt man pivot_wider()

```
statclassRe <- statclassTidy %>%
pivot_wider(ames_from = course, values_from = grade)

statclassRe
```

2.4.2 Beispiel statclass2

Namenwerte sind Spaltennamen! Dies wird korrigiert:

```
##   test momo kim sascha exam
## 1 stat1  12  13      4 exam1
## 2 stat1  NA  NA      8 exam2
## 3 stat2   5  10      5 exam1
## 4 stat2  NA  NA     NA exam2
## 5    r    6  13      3 exam1
## 6    r   NA  NA      9 exam2
## 7 spss    9   4      7 exam1
## 8 spss   NA   7     NA exam2
```

```
statclass2Tidy <- statclass2 %>%
pivot_longer(momo:sascha, names_to = "names", values_to = "grade")

statclass2Tidy
```

```
## # A tibble: 24 x 4
##   test  exam  names  grade
##   <chr> <chr> <chr>  <dbl>
## 1 stat1 exam1 momo    12
## 2 stat1 exam1 kim     13
## 3 stat1 exam1 sascha   4
## 4 stat1 exam2 momo    NA
## 5 stat1 exam2 kim     NA
## 6 stat1 exam2 sascha   8
## 7 stat2 exam1 momo     5
## 8 stat2 exam1 kim     10
## 9 stat2 exam1 sascha   5
## 10 stat2 exam2 momo   NA
## # ... with 14 more rows
```

exam beinhaltet keine Werte, sondern Namen von Variablen, nämlich von exam1 und exam2! Variablen, die die Note in der Prüfung angeben, deren Wert noch in grade steht. Deshalb nutzen wir hier jetzt pivot_wider(), um die Daten final tidy zu machen:

```
statclass2Tidy <- statclass2Tidy %>%
pivot_wider(names_from = exam, values_from = grade) %>%
```

```
relocate(names) %>%
arrange(names,test)
```

Rueckgaengig machen:

```
statclass2re <- statclass2Tidy %>%
pivot_wider(names_from = test, values_from = c(exam1,exam2))
```

2.5 if, else, elseif

Die else if Bedingung wird nur geprüft, wenn die vorherigen Bedingungen FALSE ausgeben/ sind.

```
x <- 9
if (x >= 7) {print("x ist groesser als 7")}


y <- 5
if (y >= 7) {print("y ist groesser als 7")}
else if (y >= 4 & mot < 7) {print("y liegt zwischen 4 und 6")}
```

else beinhaltet alle anderen Möglichkeiten.

```
x <- 5
if (x >= 7) {print("sehr motiviert")}
else {print("nicht motiviert")}
```

2.6 for-loop

```
for (i in 0:5) {print(i)}


# Beispiel Zielwert
n <- 5
for (i in 0:n) {print(i)}

# Beispiel Startwert
start <- 3
for (i in start:n) {print(i)}
```

paste0() verbindet Objekte mit Text und setzt dazwischen keine Leerstellen.
paste() setzt zwischen allen Elementen ein Leerzeichen, oder ein im Argument sep bestimmtes Trennzeichen.

```
teacher <- c("Baecker", "Mueller", "Kaufmann", "Bauer", "Schuster")

for (i in 1:length(teacher))
{print(paste0("Dozent:in ", i, " ist ",teacher[i]))}
```

Ein weiteres Beispiel zur Benotung:
Im Beispiel sieht man, dass wir zwei if-Ausdrücke genutzt haben. Zuerst prüfen wir, ob die Beobachtung NA in grade2 aufweist. Die Funktion is.na() liefert einen logischen Wert (TRUE, FALSE). Trifft das zu (TRUE), weisen wir NA auf der neuen Variable zu und springen mit dem Befehl next zum nächsten Fall (Schleife beginnt mit nächstem i von vorne). Mit next überspringen wir die nächsten Anweisungen

und die Schleife beginnt mit der nächsten Iteration wieder von vorne. Wenn kein NA vorliegt, wird der zweite if-Ausdruck durchgeprüft, in dem wir mit mehreren else if-Ausdrücken die Textbeschriftungen zuweisen.

```
for (i in 1:length(statistics$grade2)) {
   if (is.na(statistics$grade2[i])) {
      statistics$g2text[i] <- NA
      next
   }

   if (statistics$grade2[i] >= 13) {
      statistics$g2text[i] <- "sehr gut"
   } else if (statistics$grade2[i] >= 10 & statistics$grade2[i] < 13) {
      statistics$g2text[i] <- "gut"
   } else if (statistics$grade2[i] >= 7 & statistics$grade2[i] < 10) {
      statistics$g2text[i] <- "befriedigend"
   } else if (statistics$grade2[i] >= 5 & statistics$grade2[i] < 7) {
      statistics$g2text[i] <- "ausreichend"
   } else {
      statistics$g2text[i] <- "nicht bestanden"
   }
}
```

2.7 function()

Syntax

```
my_function <- function(arg1, arg2, ..., argn){
   # Anweisungen
}
```

Beispiel Mittelwert

```
own_mean <- function(x){
   mean = sum(x) / length(x)
   print(mean)
}
own_mean(statistics$grade)
```

Beispiel IQR 1

```
own_iqr <- function(x){
   uGrenze <- quantile(x,
   probs = 0.25,
   na.rm = TRUE
   )

   oGrenze <- quantile(x,
   probs = 0.75,
   na.rm = TRUE
   )

   abstand <- oGrenze - uGrenze

   print(paste("Das untere Quartil liegt bei:",
   uGrenze
```

```
  )
  )

  print(paste("Das obere Quartil liegt bei:",
  oGrenze
  )
  )

  print(paste("Der Interquartilsabstand betraegt:",
  abstand
  )
  )
}

own_iqr(statistics$grade)
```

Beispiel IQR 2

```
own_iqr_return <- function(x){
  uGrenze <- quantile(x,
  probs = 0.25,
  na.rm = TRUE
  )

  oGrenze <- quantile(x,
  probs = 0.75,
  na.rm = TRUE
  )

  abstand <- oGrenze - uGrenze

  print(paste("Das untere Quartil liegt bei:",
  uGrenze
  )
  )

  print(paste("Das obere Quartil liegt bei:",
  oGrenze
  )
  )

  print(paste("Der Interquartilsabstand betraegt:",
  abstand
  )
  )

  results <- list()

  results$uGrenze <- uGrenze[[1]]

  results$oGrenze <-oGrenze[[1]]

  results$abstand <- abstand[[1]]

  return(results)
}
```

```r
test <- own_iqr(statistics$grade)

test
```

3 Zusammenhangsmaße und Mittelwertvergleiche

3.1 Zusammenhangsmaße

3.1.1 Tabellen

Tabelle ausgeben (wie in 1.4).

```r
table(df$variable)
```

Anzahl gültige Faelle

```r
sum(table(df$variable))
```

Anzahl NAs

```r
sum(is.na(df$variable))
```

Gesamtlänge = gültige Faelle + NAs

```r
length(df$variable)
```

alternativ

```r
table(df$variable, useNA = "ifany")
```

Tabelle mit summarytools

```r
install.packages("summarytools")
library("summarytools")

freq(df$variable)
```

3.1.2 Kreuztabellen

table(Zeilenvariable, Spaltenvariable)

```r
table(df$variable1, df$gndr)
##
##     female male
##   0    117  109
##   1    133  135
##   2    211  225
##   3    287  331
```

Prozentuale Angaben werden mit prop.table(..., x) ausgegeben.
x = 1 –> Zeilen in Prozenten
x = 2 –> Spalten in Prozenten

```r
prop.table(table(df$variable1, df$gndr), 1)
```

```
prop.table(table(df$variable1, df$gndr), 2)
```

3.1.2.1 Randhäufigkeiten

Randhäufigkeiten werden mit margin.table(x) ausgegeben.
x = 1 –> Zeilen in Prozenten
x = 2 –> Spalten in Prozenten

```
margin.table(table(df$var1, df$var2), 1)

margin.table(table(df$var1, df$var2), 2)
```

3.1.2.2 gmodels

```
if(!require("gmodels")) install.packages("gmodels")
library("gmodels")

CrossTable(df$var1, df$var2)
```

3.1.3 Zusammenhangsmaße

3.1.3.1 Chi2 Unabhängigkeitstest

H_0 = Variablen sind statistisch unabhängig

```
chi <- chisq.test(df$var1, df$var2)
chi
##
##  Pearson's Chi-squared test
##
## data: mytable
## X-squared = 14.123, df = 10, p-value = 0.1674
```

Interpretation:

p-value = $0.1674 > 0.05$

Es liegt kein signifikanter Testwert vor

⇒Nullhypothese kann beibehalten werden

⇒Variablen statistisch unabhängig

3.1.3.2 Annahmen

Um einen χ^2 Test durchzuführen müssen 2 Bedingungen erfüllt sein.

1. $\geq$10 Beobachtungen pro Zelle
2. $\geq$5 erwartete Beobachtungen pro Zelle

Ausgabe der Liste eines χ-Tests

```
chi <- chisq.test(df$var1, df$var2)
str(chi)
```

Einzelne Teile der Liste ausgeben und aufrufen

```
ls(chi)

chi$expected
```

3.1.3.3 Cramer's V

Cramer's V wird genutzt um die Stärke des Zusammenhangs auszurechnen.
conf.level ist das Konfidenzintervall.

```
install.packages("DescTools")
library("DescTools")

CramerV(df$var1, pss$gndr, conf.level = 0.95 )

##   Cramer V    lwr.ci    upr.ci
## 0.05365996 0.00000000 0.06613029
```

Interpretation:

erste Spalte - Cramer's V

zweite & dritte Spalte - Konfidenzintervall

Wenn das Intervall 0 schneidet, ist es nicht signifikant.
Hier eine Tabelle für die Interpretation:

unteres Ende	oberes Ende	Interpretation
0	0.1	kein Zusammenhang
0.1	0.3	gering
0.3	0.6	mittel
0.6	1	stark

3.1.4 Korrelationen berechnen

Pearson's r	(pseudo-)metrische Variablen, linear, monotone Beziehung, Varianzgleichheit, bivariate Normalverteilung
Spearman's ρ	mind. ordinale Variablen, monotone Beziehung

Zunächst Pearson's r:
Dafür müssen die Annahmen für die Variablen getestet werden:

- Stichprobe von verbundenen Werten ✓
- beide Variablen metrisch ✓
- Beziehung zwischen Variablen ist linear

Scatterplot erstellen:

```
plot(df$var1, df$var2)
```

Wenn sich zu viele Datenpunkte überlappen kann man nichts erkennen. Durch streuen der Punkte lässt sich dies vermeiden.

```
plot(jitter(df$var1, 3) ~
jitter(df$var1, 3)
)
```

- Beziehung zwischen Variablen ist linear ✓

Nun kann Pearson's r berechnet werden!

3.1.4.1 Korrelationskoeffizient berechnen

```
cor(
df$var1, df$var2,
method = "pearson", # alternativ hier "spearman"
use = "complete.obs"
)
## [1] 0.2318401
```

Die Ausgabe zeigt nur den r-Wert und keinen p-Wert an. Ohne den p-Wert kann keine Aussage über die Signifikanz getroffen werden. **ALTERNATIV:**

```
install.packages("psych")
library("psych")

corr.test(
df$var1, df$var2,
method = "pearson",
use = "complete.obs"
)
```

Der p-wert ist an der letzten Stelle eingetragen.
Interpretation:

p-value $< 0.05 \Rightarrow$ statistisch signifikant

r $= 0.23 \Rightarrow$ schwacher Zusammenhang

Dieser Test generiert drei Matrizen, die du später für die Visualisierung nutzen kannst.

- Korrelationsmatrix
- Matrix der Stichprobengroesse
- Matrix der p-Werte

3.1.4.2 mehrere Korrelationen berechnen

```
cor(
df[c("var1", "var2", "var3", "var4")],
method = "pearson",
use = "complete.obs"
)


library("psych")
corr.test(
df[c("var1", "var2", "var3", "var4")],
```

```
method = "pearson",
use = "complete.obs"
)
```

3.1.5 Grafiken

3.1.5.1 Library psych - Korrelationsübersicht

```
pairs.panels(
df[c("var1", "var2", "var3", "var4")],
method = "pearson",
jiggle = TRUE, #fuer pseudometrische Daten
stars = TRUE #Konvention fuer Signifikanzen
)
```

Im oberen Drittel befinden sich die Korrelationskoeffzienten, in der diagonalen die univariate Verteilung der jeweiligen Variable und im unteren Drittel die bivariate Verteilung des Variablenpaars.

3.1.5.2 Library corrplot - Heatmap

Hier sind alle Funktionen der Library corrplot beschrieben:
https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html Heatmap:

```
install.packages("corrplot")
library("corrplot")

cor2 <- corr.test(
df[c("var1", "var2", "var3", "var4")],
method = "pearson",
use = "complete.obs"
)

cor2
ls(cor2)


corrplot(
cor2$r,
p.mat = cor2$p,  # Matrix mit p-Werten
insig = "blank", # nicht signifikante = leer
type = "upper",  # auch lower moeglich
method = "circle" # verschiedene Optionen moeglich
)
```

3.2 Mittelwertvergleiche

3.2.1 Ein-Stichproben-T-Test

3.2.1.1 zweiseitiger Test

Beispiel: Es wird das Alter(age) eines Datensatzes(df) berechnet.

```
mean(df$age, na.rm = TRUE)
```

Das Durchschnittsalter liegt bei 42.83006 Jahren
Da im Datensatz nur Personen ab 16 Jahren befragt wurden weicht das Durchschnittsalter ab.
Ist diese Abweichung statistisch signifikant?
Da es keine Annahme über die Richtung gibt wird ein **zweiseitiger Test** durchgeführt.

```
t.test(
df$age,
mu = 36.8,
alternative = "two.sided")
```

Als Ausgabe erscheint der t-value, p-value, confidence interval und der mean. Der p-Wert ist kleiner als 0.05 und somit ist das Ergebnis signifikant.
Die Abweichung lässt sich dann folgendermaßen berechnen:

```
diff_age <- mean(df$age, na.rm = TRUE) - 36.8

diff_age
```

3.2.1.2 einseitiger Test

```
#one-sided, greater
t.test(
df$age,
mu = 36.8,
alternative = "greater"
)

#one-sided, lower
t.test(
df$age,
mu = 36.8,
alternative = "less"
)
```

3.2.2 Zwei-stichproben-T-Testf) ge

3.2.2.1 ungepaart

In einem ungepaarten zwei-Stichproben-T-Test werden zwei unabhängige Gruppen voneinander getestet.
Dafür müssen 2 Bedingungen getestet werden

1. Varianzgleichheit (Levene-Test)
2. Normalverteilung der metrischen Variablen (uV)

1. Varianzgleichheit - Levene Test

```
install.packages("car")
library("car")

leveneTest(
df$var1,
df$gndr,
center = "mean")
```

```
## Levene's Test for Homogeneity of Variance (center = "mean")
##        Df F value Pr(>F)
## group   1  0.5405 0.4623
##       4998
```

Interpretation:

H_0: Beide Gruppen haben in der metrischen Variable diesselbe Varianz.

p-value = 0.5405 > 0.05

⇒ Varianzgleichheit wird angenommen

Durchführung des Tests

```
t.test(
df$var1 ~ df$gndr, #metrische variable ~ kategorielle variable
mu = 0,
alternative = "two.sided",
paired = FALSE,  # ungepaarte Stichproben!
var.equal = TRUE # Option des Levene-Tests!
)
##
##  Two Sample t-test
##
## data: df$var1 by df$gndr
## t = 1.3509, df = 4998, p-value = 0.1768
## alternative hypothesis: true difference in means between group female and group
    male is not equal to 0
## 95 percent confidence interval:
##  -0.1436357 0.7803096
## sample estimates:
## mean in group female mean in group male
##           34.46080           34.14246
```

Werte:

- t-value = 1.3509
- p-value = 0.1768
- Confidence Interval = [-0.1436357, 0.7803096]
- group female = 34.46080
- group male = 34.14246

Im Durchschnitt ist var1 bei Männern 0.31834 kleiner. Das ist aber statistisch nicht relevant. **polytome Variablen - mehr als 2 Ausprägungen** Die Variable "Bildungsabschluss"hat 5 Ausprägungen(bildung1,bildung2,...3,4,5).

```
#Test of homogeneity of variances
leveneTest(
df$var1,
df$bildung,
```

```
center = "mean"
)
## Levene's Test for Homogeneity of Variance (center = "mean")
##        Df F value Pr(>F)
## group   4  0.4981 0.7372
##       4643

t.test(
df$bildung[df$bildung == "bildung 1"],
df$bildung[df$bildung == "bildung 5"],
mu = 0,
alternative = "two.sided",
paired = FALSE,
var.equal = TRUE
)
##
##  Two Sample t-test
##
## data: df$var1[df$bildung == "bildung 1"] and df$var1[df$bildung == "bildung 5"]
## t = 9.723, df = 1078, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  4.492021 6.763452
## sample estimates:
## mean of x mean of y
##  36.19636 30.56863
```

Interpretation:
Bei Personen mit niedriger Bildung (mean of x) ist var1 höher als bei Personen mit höherer Bildung (mean of y). Der Effekt ist signifikant und die Differenz beträgt 5.62773.

3.2.2.2 gepaart

Bei gepaarten 2-Stichproben-T-Tests sind die Variablen der beiden Gruppen voneinander abhängig.
Annahmen:

1. Variablen sind metrisch ✓
2. Differenz folgt einer Normalverteilung (relevant für n $\leq$ 30) ✓

```
t.test(
df$var1,
df2$var1,
alternative = "two.sided",
paired = TRUE
)
```

3.2.3 ANOVA/F-Test - mehr als 2 Gruppen

Annahmen:

1. abhängige Variable metrisch ✓
2. unabhängige Variable kategoriell ✓
3. Gruppen sind unabhängig voneinander ✓

4. für n $\leq$ 25n $\Rightarrow$ Normalverteilung der metr. Variable in jeder Gruppe ✓
5. Varianzgleichheit zwischen Gruppen

```
leveneTest(
df$var1 ~ df$bildung,
center = "mean"
)
```

Interpretation:
Zwischen keinem Gruppenpaar gibt es signifikante Differenzen, daher wird Varianzgleichheit angenommen.
H_0: Alle Gruppen haben dieselbe Varianz in der metrischen Variable.
Berechnen des Tests(ungleiche Varianzen)

```
oneway.test(
df$var1 ~ df$bildung,
var.equal = TRUE
)
##
##  One-way analysis of means
##
## data: df$var1 and df$bildung
## F = 40.16, num df = 4, denom df = 4643, p-value < 2.2e-16
```

Interpretation:
p-value $<$ 2.2e-16 $<$ 0.05 $\Rightarrow$ es gibt mindestens zwischen 2 Gruppen signifikante Differenz.
Feststellen der Gruppen, zwischen denen die signifikante Differenz liegt:

```
pairwise.t.test(
df$var1, # metrische Variable zuerst
df$bildung, # Gruppenvariable als zweites
p.adj = "holm" # Korrektur (Standardverwendung)
)
```

4 Lineare Regression, Modellexport, Regressionsdiagnostik

4.1 Lineare Regression

Ziel: Abhängigkeiten prüfen
Bedingungen:

- abhängige Variable = pseudometrisch
- unabhängige Variable = pseudometrisch/ kategoriell
- Beziehung zwischen unabhängiger und abhängiger Variable muss linear sein

4.1.1 Bivariates Beispiel

- Personen, die ökonomisch zufrieden sind, sind auch zufriedener mit dem politischen System im Ganzen.

  $\text{stfdem} = \hat{\beta}_0 + \hat{\beta}_1 \cdot \text{stfeco} + \hat{\epsilon}$

- Personen, die *unabhängige Variable* sind, sind auch *abhängige Variable*

  abhängige Variable $= \hat{\beta}_0 + \hat{\beta}_1 \cdot$ unabhängige Variable $+ \hat{\epsilon}$

```
olsModel <- lm(
stfdem ~ 1 + stfeco, #Formel der Regression
data = pss # Datenobjekt
)
summary(olsModel)


olsModel <- lm(y ~ 1 + x, data = df)
summary(olsModel)
```

4.1.1.1 Koeffizienten Interpretation

```
coef(olsModel)
## (Intercept)    stfeco
##   0.4808475  0.8727124
```

Wenn stfeco um eine Einheit ansteigt, steigt stfdem um 0.8727124 an.
Wenn eine Person stfeco $= 0$ hat, dann ist stfdem $= 0.4808475$.

```
confint(olsModel)
##                2.5 %    97.5 %
## (Intercept) 0.3448187 0.6168763
## stfeco     0.8461641 0.8992608
```

Das Konfidenzintervall ist $[0.8461641, 0.8992608] \neq 0 \Rightarrow$ signifikant $\Rightarrow$ Nullhypothese$(H_0 : \beta_1 = 0)$ verwerfen

4.1.1.2 weitere Ausgaben

```
olsModel$coefficients # Koeffizienten
coef(olsModel) # Koeffizienten
## (Intercept)    stfeco
##   0.4808475  0.8727124

head(olsModel$fitted.values) # geschaetzte Werte
head(fitted(olsModel)) # ersten 6 geschaetzten Werte
##        1        2        3        4        5        6
## 5.717122 6.589834 5.717122 3.971697 4.844410 5.717122

head(olsModel$residuals) # Residuen
head(resid(olsModel)) # Residuen
##          1          2          3          4          5          6
##  1.2828780 1.4101656 0.2828780 1.0283029 -0.8444096 0.2828780

confint(olsModel) # Konfidenzintervalle
##                2.5 %    97.5 %
## (Intercept) 0.3448187 0.6168763
## stfeco     0.8461641 0.8992608
```

Um das Modell global zu vergleichen nutzen wir R^2, das lässt sich aus der vorletzten Zeile von ablesen.

```
summary(olsModel)
## Multiple R-squared: 0.4587, Adjusted R-squared: 0.4585
```

Das Modell erklärt 45.85% der Varianz von der abhängigen Variable Y.
Wenn stfeco um eine Einheit ansteigt, steigt stfdem um 0.8727124 an.

4.1.2 Multivariates Beispiel

Gleichung:

$$\text{stfdem} = \beta_0 + \beta_1 \cdot \text{stfeco} + \beta_2 \cdot \text{trstlgl} + \epsilon$$

$$\mathbf{Y} = \beta_0 + \beta_1 \cdot \mathbf{X}_1 + \beta_2 \cdot \mathbf{X}_2 + \epsilon$$

```
olsModel2 <- lm(
stfdem ~ 1 + stfeco + trstlgl,
data = pss
)
summary(olsModel2)

olsModel2 <- lm(Y ~1 + x + z, data=df)
summary(olsModel2)

## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.67658   0.09318   7.261 4.44e-13 ***
## stfeco       0.87361   0.01355 64.468 < 2e-16 ***
## trstlgl    -0.04212   0.01319 -3.194 0.00141 **
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.732 on 4890 degrees of freedom
##   (107 observations deleted due to missingness)
## Multiple R-squared: 0.4598, Adjusted R-squared: 0.4596
## F-statistic: 2081 on 2 and 4890 DF, p-value: < 2.2e-16
```

- Das Modell kann 45.96% der Varianz in stfdem erklären(vorletzte Zeile).
- stfeco steigt um 1 $\Rightarrow$ stfdem steigt um 0.87361 (Coefficients)
- trstlgl steigt um 1 $\Rightarrow$ stfdem sinkt um -0.04212
- Beide Effekte sind signifikant $p<0.05$

4.1.3 Dichotome Kategorielle Variablen

$\mathbf{X}_3 = \text{gndr}$ ist hier eine dichotome kategorielle Variable.
$\text{gndr} = \text{male} \vee \text{female}$

$$\text{stfdem} = \beta_0 + \beta_1 \cdot \text{stfeco} + \beta_2 \cdot \text{trstlgl} + \beta_3 \cdot \text{gndr} + \epsilon$$

$$\mathbf{Y} = \beta_0 + \beta_1 \cdot \mathbf{X}_1 + \beta_2 \cdot \mathbf{X}_2 + \beta_3 \cdot \mathbf{X}_3 + \epsilon$$

```
olsModel3 <- lm(
stfdem ~ 1 + stfeco + trstlgl + gndr,
data = pss
)
summary(olsModel3)


olsModel3 <- lm(Y ~ 1 + a + b + c)

## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.70975  0.09539  7.440 1.18e-13 ***
## stfeco       0.87435   0.01356 64.496 < 2e-16 ***
## trstlgl    -0.04137   0.01319 -3.136 0.00173 **
## gndrmale   -0.08020   0.04957 -1.618 0.10573
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.732 on 4889 degrees of freedom
##   (107 observations deleted due to missingness)
## Multiple R-squared: 0.4601, Adjusted R-squared: 0.4598
## F-statistic: 1389 on 3 and 4889 DF, p-value: < 2.2e-16
```

- Die Referenzkategorie ist female, also weibliche Befragte.
- Maennliche Personen haben also eine um -0.08020 niedrige Zufriedenheit als weibliche Personen.

Das Modell erklärt 45.98% der Varianz in der Variable stfdem. Die Zufriedenheit mit der ökonomischen Leistung (stfeco) sowie das Vertrauen in das Rechtssystem (trstlgl) haben einen signifikanten Effekt auf die Zufriedenheit mit der Demokratie (stfdem). Der Effekt von stfeco ist positiv (β_1=0.87435), der Effekt von Vertrauen in das Rechtssystem (β_2 = -0.04137) und der Effekt für männliche Personen ($\beta_3 = -0.08020$) sind beide negativ. Personen, die ein höheres Vertrauen haben oder männlich sind, haben also einge leicht geringere Zufriedenheit.
Eigentlich berechnen wir also
$\text{stfdem} = \beta_0 + \beta_1 \cdot \text{stfeco} + \beta_2 \cdot \text{trstlgl} + \beta_3 \cdot \text{male} + \epsilon$

4.1.4 Polytome Kategorielle Variablen

```
olsModel4 <- lm(
stfdem ~ 1 + stfeco + trstlgl + gndr + edu,
data = pss
)
summary(olsModel4)

## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.535964  0.112218  4.776 1.84e-06 ***
## stfeco         0.854641  0.014166 60.332 < 2e-16 ***
## trstlgl       -0.044393  0.013572 -3.271 0.00108 **
## gndrmale       0.001836  0.051229  0.036 0.97142
## eduES-ISCED II 0.168395  0.076925  2.189 0.02864 *
## eduES-ISCED III 0.343037 0.076832  4.465 8.21e-06 ***
## eduES-ISCED IV 0.419061  0.085739  4.888 1.06e-06 ***
## eduES-ISCED V  0.870502  0.125865  6.916 5.29e-12 ***
```

```
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.716 on 4542 degrees of freedom
##   (450 observations deleted due to missingness)
## Multiple R-squared: 0.4643, Adjusted R-squared: 0.4635
## F-statistic: 562.3 on 7 and 4542 DF, p-value: < 2.2e-16
```

Die niedrigste Ausprägung wird als Referenzkategorie gesetzt.
$\text{stfdem} = \beta_0 + \beta_1 \cdot \text{stfeco} + \beta_2 \cdot \text{trstlgl} + \beta_3 \cdot \text{male} + \beta_4 \cdot \text{eduLevelII} + \beta_5 \cdot \text{eduLevelIII} + \beta_6 \cdot \text{eduLevelVI} + \beta_7 \cdot \text{eduLevelV} + \epsilon$

4.1.4.1 Ändern der Referenzkategorie

Ziel: Die mittlere Kategorie als Referenzkategorie setzen:

```
pss$edu <- relevel(
pss$edu,
ref = "ES-ISCED III"
)

olsModel5 <- lm(
stfdem ~ 1 + stfeco + trstlgl + gndr + edu,
data = pss
)

summary(olsModel5)
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.879001  0.106299  8.269  < 2e-16 ***
## stfeco         0.854641  0.014166 60.332  < 2e-16 ***
## trstlgl       -0.044393  0.013572 -3.271  0.00108 **
## gndrmale       0.001836  0.051229  0.036  0.97142
## eduES-ISCED I -0.343037 0.076832 -4.465 8.21e-06 ***
## eduES-ISCED II -0.174643 0.066577 -2.623 0.00874 **
## eduES-ISCED IV 0.076024 0.075825  1.003  0.31610
## eduES-ISCED V 0.527465  0.119052  4.431 9.62e-06 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.716 on 4542 degrees of freedom
##   (450 observations deleted due to missingness)
## Multiple R-squared: 0.4643, Adjusted R-squared: 0.4635
## F-statistic: 562.3 on 7 and 4542 DF, p-value: < 2.2e-16
```

Das Modell erklärt 46.35% der Varianz in stfdem. Der Effekt von stfeco ist positiv und signifikant ($p<0.001$). Personen mit höherem Vertrauen in das Rechtssystem haben eine geringe Zufriedenheit ($\beta_2 = -0.044393$, $p<0.01$). Männliche und weibliche Befragte haben keine unterschiedliche Zufriedenheit ($\beta_3 = 0.001836$, $p>0.05$). Im Vergleich zu Personen mit mittlerem Bildungsabschluss, haben Personen mit sehr niedrigem Abschluss (ES-ISCED I) und niedrigem Bildungsabschluss (ES-ISCED II) ein geringeres Vertrauen ($\beta_4 = -0.343037 \beta_5 = -0.174643$). Beide Effekte sind signifikant. Personen mit dem höchsten Bildungsabschluss haben ein deutlich höheres Vertrauen als Personen mit mittlerem Bildungsabschluss ($\beta_7 = 0.527465$,$p<0.001$). Personen mit dem zweithöchsten Abschluss haben ein geringfügig höheres Vertrauen ($\beta_6 = 0.076024$), dieser Effekt ist aber nicht signifikant.

4.2 Standardisierte Koeffizienten und Export

4.2.1 Z-Standardisierung

scale() ist Teil des tidyverse 2.3.

```
pss <- pss %>%
mutate(
stfdemZ = scale(stfdem),
stfecoZ = scale(stfeco),
trstlglZ = scale(trstlgl)
)

olsModel2Z <- lm(
stfdemZ ~ 1 + stfecoZ + trstlglZ,
data = pss
)

summary(olsModel2Z)

## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.004028  0.010511   0.383  0.70155
## stfecoZ     0.697244  0.010815 64.468 < 2e-16 ***
## trstlglZ   -0.033592  0.010517 -3.194  0.00141 **
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7352 on 4890 degrees of freedom
##   (107 Beobachtungen als fehlend gelscht)
## Multiple R-squared: 0.4598, Adjusted R-squared: 0.4596
## F-statistic: 2081 on 2 and 4890 DF, p-value: < 2.2e-16
```

Mit jedem Anstieg um eine Standardabweichung in stfeco steigt stfdem um 0.6972440.6972440.697244 Standardabweichungen. Es wird sichtbar, dass der Effekt von stfeco stärker ist als der von trstlgl ($0.697244 > |-0.033592|$).

4.2.2 Export Tabellen und Grafiken

4.2.2.1 Modelle in Textdateien exportieren

```
install.packages("modelsummary",
dependencies = TRUE)
library("modelsummary")

modelsummary(
list(olsModel,olsModel2)
)

models <- list(olsModel, olsModel2)
modelsummary(models)
```

Exportformate Die Datei wird automatisch im working directory gespeichert.

```
modelsummary(
models,
```

```
output = "table.docx" # Word-Datei (package flextable noetig)
)

modelsummary(
models,
output = "table.html" # HTML-Datei
)

modelsummary(
models,
output = "table.tex" # Tex-Datei
)

modelsummary(
models,
output = "table.md" # Markdown-Datei
)

modelsummary(
models,
output = "table.txt" # Text-Datei
)

modelsummary(
models,
output = "table.png" # Bild-Datei
)
#-------------------------------------------------------------
#Ausgabe in Konsole:
#-------------------------------------------------------------
modelsummary(
models,
output = "html" # html in Konsole
)

modelsummary(
models,
output = "latex" # latex in konsole
)

modelsummary(
models,
output = "markdown" # markdown in konsole
)
```

AUSGABE anpassen

```
# 4 Dezimalstellen nach dem Dezimaltrennzeichen
modelsummary(
models,
fmt = 4)

# 4 Dezimalzeichen und 0 am Ende
modelsummary(
models,
fmt = "%.4f")
```

```
# wissenschaftliche Notation
modelsummary(
models,
fmt = "%.4e")
```

Änderung . zu ,

```
options(OutDec = ",")

modelsummary(
olsModel,
fmt = 2)#

options(OutDec = ".")
```

Koeffizienten anpassen Alle Einstellungen sind hier nachzulesen: https://cran.r-project.org/web/packages/modelsummary/modelsummary.pdf
Bei estimate geben wir in geschweiften ... Klammern an, was angezeigt werden soll in der Zeile der jeweiligen Variable. Mit dem Argument statistics = NULL schalten wir die vorher vorhandene zweite Zeile nach jeder Variablen aus.

```
modelsummary(
models,
fmt = 1,
estimate = "{estimate} [{conf.low}, {conf.high}]",
statistic = NULL)

modelsummary(
models,
fmt = 1,
estimate = "{estimate}{stars}",
statistic = NULL)

modelsummary(
models,
fmt = 1,
estimate = "{estimate}{stars}",
statistic = NULL,
notes = list('$^{*} p< 0.1$, $^{**} p<0.05$, $^{***} p<0.01$'))

modelsummary(
models,
stars = TRUE # standardmaessig an!)

modelsummary(
models,
fmt = 1,
estimate = "{estimate}{stars}",
statistic = "[{conf.low}, {conf.high}]")
#----------------------------------------
#Hilfreiche Standardausgaben fuer statistic
#----------------------------------------
modelsummary(
models,
fmt = 1,
stars = TRUE,
statistic = "conf.int",
```

```
conf_level = .95)

modelsummary(
models,
statistic = c("conf.int",
"s.e. = {std.error}",
"t = {statistic}",
"p = {p.value}"))

modelsummary(
models,
coef_rename=c("stfeco" = "Zufriedenheit OEkonomie",
"trstlgl" = "Vertrauen Rechtssystem"))
```

DOTWHISKER

Link zu olsModel2: 4.1

```
install.packages("dotwhisker")
library("dotwhisker")

dwplot(olsModel2)

dwplot(
list(
olsModel,
olsModel2))


dwplot(
list(
olsModel,
olsModel2
)
) +
# Linie bei 0
geom_vline(
xintercept = 0,
linetype = "dashed"
) +
# Renaming der y-Achse
scale_y_discrete(labels = rev(
c(
"Zufriedenheit \n OEkonomie",
"Vertrauen \n Rechtssystem"
)
)
) +
# x-Achse modifiziert
scale_x_continuous(
breaks = seq(
-1,
1,
0.2
),
limits = c(-1, 1)
) +
# Titel und Caption hinzugefuegt
```

```
labs(title = "Lin. Regressionsmodelle auf stfdem",
caption = "Data: Panem Social Survey."
)
```

4.3 Regressionsdiagnostik

Ausgangspunkt ist das Modell olsModel2:

$$\text{stfdem}_i = \beta_0 + \beta_1 \cdot \text{stfeco}_i + \beta_2 \cdot \text{trstlgl}_i + \epsilon_i$$

Nach Anwendung der linearen Regression ergibt sich:

$$\text{stfdem}_i = 0.67658 + 0.87361 \cdot \text{stfeco}_i + 0.04212 \cdot \text{trstlgl}_i + \epsilon_i$$

Für die Vorhersage wird das ϵ_i vernachlässigt.
Annahmen der linearen Regression:

1. lineare Beziehung (4.3.1)
2. $\epsilon_i \sim \mathcal{N}(\mu, \sigma^2)$
3. $Var(\epsilon_i) = \sigma^2$(**Homoskedastizität**)
4. $Cov(\epsilon_i, \epsilon_j) = 0, i \neq j$(**Auto-Korrelation**)
5. keine lineare Abhängigkeit unter den unabhängigen Variablen (**Multikollinearität**)
6. keine einflussreichen Fälle

Wirkung bei Verletzung:

1. verzerrte Schätzer
2. invalide Signifikanztest, verzerrte Schätzer
3. verzerrte Schätzer
4. ineffiziente Schätzung
5. verzerrte Schätzer
6. verzerrte Schätzer

Mögliche Lösung:

1. nicht-lineare Transformation
2. *nichts!*
3. Transformation der abhängigen Variable (oder der linearen Gleichung)
4. nicht-lineare Transformation
5. Variablen entfernen
6. Beobachtungen entfernen (muss aber theoretisch begründbar sein!)

Test vorgehen:

1. Plot Inspektion
2. Plot Inspektion (Shapiro-Wilk-Test)
3. Breusch-Pagan-Test
4. Durbin-Watson-Statistik
5. Korrelation zwischen unabhängigen Variablen sowie VIF / Tolerance factor
6. Cook's D Cook's D

4.3.1 Annahme 1: lineare Beziehung

Argument [1] in plot wählt Residuals vs. Fitted

```
plot(
olsModel2,
1)
```

⇒ der Plot sollte uns eine nahezu horizontale Linie anzeigen; dann ist die Annahme erfüllt und wir können eine lineare Beziehung annehmen.

5 Darstellungen

5.1 Plots

Benötigt wird „ggplot". Dieses ist Im tidyverse enthalten. 2 Möglichkeiten „ggplot" zu laden:

```
library("tidyverse")
library("ggplot2")
```

5.1.1 Balkendiagramme

5.1.1.1 Säulen und Balkendiagramme

Sinnvoll für eine einzige kategoriale Variable
Es ist sinnvoll den Plot in einem Objekt zu speichern, wenn man später noch Layer hinzufügen möchte.

leerer Plot

```
ggplot(data = df,
mapping = aes(x = variablenname))
```

Balken hinzufügen

```
barplot1 <- ggplot(
           data = pss,
           mapping = aes(x = variablenname)
             ) +
         geom_bar()

barplot1
```

Aktuell ist jeder Balken auf sich selbst bezogen 100%. Für Prozentuale Balken gibt man an, dass alle zu einer Gruppe gehören.

```
barplot2 <- ggplot(
           data = pss,
           mapping = aes(
              x = edu,
              y = ..prop..,
              group = 1
              )
           ) +
         geom_bar()

barplot2
```

Säulendiagramm zu Balkendiagramm:

```
     barplot1 +
    coord_flip()
```

5.1.1.2 Farben

Color umrandet die Balken:

```
barplot <- ggplot(pss,
              aes(variable, color = edu)) +
         geom_bar()

barplot
```

fill füllt die balken mit Farbe

```
    barplot <- ggplot(pss,
    aes(variable, fill = edu)) +
    geom_bar()

    barplot
```

Farben selbst festlegen

```
barplot <- ggplot(pss, aes(edu, fill = edu) )
 +
geom_bar(fill = c("steelblue",
              "darkgoldenrod",
              "seagreen",
              "red4",
              "orange",
              "darkslategray2")
      )

barplot
```

```
# a colourblind-friendly palettes
   cbp1 <- c("#999999",
             "#E69F00",
             "#56B4E9",
             "#009E73",
             "#F0E442",
```

```
            "#0072B2",
            "#D55E00",
            "#CC79A7")

barplotCb <- ggplot(pss, aes(edu, fill = edu)) +
         geom_bar() +
         scale_fill_manual(values = cbp1)

barplotCb
```

Für schwarz-weiß drucke:

```
barplotGray <- ggplot(pss, aes(edu, fill = edu)) +
           geom_bar() +
           scale_fill_grey()

barplotGray
```

Vorgefertigte Farbpalette:

```
install.packages("RColorBrewer")
# Installation Entwicklertool
install.packages("devtools",dependencies = TRUE)
# Install beyonce palette
devtools::install_github("dill/beyonce")


library("RColorBrewer")
display.brewer.all()
```

Farbpalette auswählen und nutzen

```
barplotBrewer <- ggplot(pss, aes(edu, fill = edu)) +
geom_bar() +
scale_fill_brewer(palette = "Dark2")

barplotBrewer
```

Beyonce Farbpaletten:

```
library("beyonce")

# Ausgabe aller Farbpaletten [beginnend bei 1]
par(mfrow=c(26,5))
for(i in 1:130) print(beyonce_palette(i))


barplotBeyonce <- ggplot(pss, aes(edu, fill = edu)) +
geom_bar() +
scale_fill_manual(values = beyonce_palette(25))

barplotBeyonce
```

5.1.1.3 Wertebeschriftungen

```
barplotBeyonce <- ggplot(pss, aes(edu, fill = edu)) +
geom_bar() +
scale_fill_manual(values = beyonce_palette(25))

+ geom_text(stat = "count", aes(label = ..count..))
```

Lage der Zahlen im Balken korrigieren und Schriftfarbe korrigieren

```
barplotBeyonce +

geom_text(stat = "count",
          aes(label = ..count..),
          vjust = 1.5,
          color = "white")
```

Schriftgröße anpassen

```
barplotBeyonce +
geom_text(stat = "count", aes(label = ..count..), vjust = 1.5,

size = 8,

color = "white")
```

5.1.1.4 Achsen, Legende, Titel

Achsen Grenzen mit limits=c() anpassen und AChsenmarkierungen mit breaks=() anpassen

```
barplotbeyonce +
scale_y_continuous(limits = c(0, 1750)) +
scale_x_continuous(breaks = c(0, 1750, 100))
```

Achsenvariablen die nicht $\in \mathbb{R}$, sondern diskret sind werden folgendermaßen bearbeitet:

```
barplotBeyonce <- barplotBeyonce +
scale_x_discrete(limits = c("ES-ISCED I", "ES-ISCED II",
                            "ES-ISCED III", "ES-ISCED IV",
                            "ES-ISCED V", NA)
                 )

barplotBeyonce
```

Legende

```
barplotBeyonce +
theme(legend.position = "bottom") # left, right, top oder none
```

Beschriftungen

```
barplotBeyonce +
scale_fill_manual(
   name = "Bildungsniveau",
   labels = c("sehr niedrig", "niedrig","mittel","hoch","sehr hoch", "NA"),
   values = beyonce_palette(25)
)
```

Titel

```
barplotBeyonce +
labs(
x = "Bildungsniveau",
y = "Haeufigkeiten",
title = "My first try with ggplot2",
subtitle = "")
```

5.1.1.5 Speichern

plots speichert man am besten mit ggsave(). R speichert die Dateien im working directory. Wenn man nicht weiß wo das ist: getwd().
Tipp: erstelle einen Ordner „img“ für deine Bilder!

```
getwd()

ggsave("./img/mffggplot.png",
      width = 8, height = 6,
      units = "in", dpi = 450)
```

5.1.1.6 Gruppieren

Die Variable sollte eine *factor* Variable sein, ansonsten wird sie zur factorvariable coerced,w as zu Problemen führen kann.
Balken übereinander:

```
    barGroup <- ggplot(pss, aes(edu, fill = gndr)) +
            geom_bar()
```

Balken nebeneinander:

```
barGroup <- ggplot(pss, aes(edu, fill = gndr)) +
geom_bar(position = position_dodge())
```

5.1.2 Histogramme

5.1.2.1 Start

Histogramme werden für einzelne kontinuierliche Variablen genutzt.

```
hist <- ggplot(pss, aes(agea)) +
geom_histogram()

hist
```

Mit Dichteverteilung:

```
histDens <- ggplot(pss, aes(agea)) +

geom_histogram(aes(y = ..density..), color = "lightgray", fill = "gray") +

geom_density(alpha = 0.2, fill = "lightblue") +

labs(x = "Age in years", y = "Density", title = "Histogram of Age (PSS)")
```

5.1.2.2 Gruppieren

In facetgrid() wird mit einer Formel (x y) angegeben, nach welcher Variable x bzw. y Achse getrennt werden.

```
histGroup <- ggplot(pss,aes(agea, fill = gndr)) +

geom_histogram(aes(y = ..density..),alpha = 0.2, bins = 30, position = "identity") +

geom_density(alpha = 0.2) +

facet_grid(.~ gndr ) +

scale_fill_manual(values = beyonce_palette(72)) +

labs(x = "Age in years", y = "Density", title = "Histogram of Age") +

theme(legend.position = "none")
```

5.1.2.3 Ridgeline

Ridgelineplots zeigen wie Histogramme die Dichte einer Variablen an und sind deshalb eine gute Alternative zu Histogrammen.
X-Achse ist agea, Y-Achse ist gndr

```
install.packages("ggridges")
library("ggridges")


ridgeline <- ggplot(pss, aes(agea,gndr,fill = gndr)) +

geom_density_ridges(scale = 0.9,alpha = 0.4) +

theme_ridges() +

theme(legend.position = "none")


ridgeline
```

Angepasst:

```
ridgeline +

labs(title = "Ridgeline Plot", x = "Age in years",y = "") +

theme_ridges(grid = TRUE, center_axis_labels = TRUE) +

scale_fill_cyclical(labels = c("female", "male"),
                    values = cbp1,
                    guide = "legend",
                    name = "Gender") +

theme(axis.text.y = element_blank())
```

5.1.3 Boxplots

5.1.3.1 geomboxplot()

```
boxplot <- ggplot(pss, aes(agea)) +
geom_boxplot()

boxplot
```

angepasst und gedreht

```
boxplot <- ggplot(pss, aes(agea)) +

geom_boxplot() +

coord_flip() +

scale_y_continuous(breaks = NULL)

boxplot
```

Boxplot in Abhängigkeit vom District Gruppiert ausgeben:

```
boxplotDistrict <- ggplot(pss, aes(district, agea, fill = district)) +

geom_boxplot()
```

angepasst:

```
boxplotDistrict +

scale_fill_manual(name = "Distrikt", values = cbp1) +

scale_x_discrete(limits = c("Distrikt 10", "Distrikt 7", "Distrikt 12",
"Distrikt 5","Distrikt 1")) +

scale_y_continuous( breaks = seq(0, 100, 5)) +

labs(x = "District", y = "Age in years",title = "Boxplots of Age by District") +

coord_flip()
```

5.1.3.2 ggboxplot()

Hierzu wird die library ggpubr gebraucht. Im Beispiel wird Arbeitszeit nach Geschlecht ausgegeben.

```
library(ggpubr)

ggbox <- ggboxplot(pss,
              x = "gndr",
              y = "wkhtot",
              color = "ivory4",
              fill = "gndr",
              palette = c("lightblue","pink"),
              ylab = "Total work hours (incl. overtime)",
              xlab = "Gender (1:male, 2:female)",na.rm = TRUE)
```

```
ggbox
```

Mit horizontaler Linie den Mittelwert jeder Gruppe anzeigen:

```
ggbox +

geom_hline(aes(yintercept = mean(wkhtot[gndr == "male"], na.rm = TRUE)),
        lty = 2,
        lwd = 1,
        color = "lightblue") +

geom_hline(aes(yintercept = mean(wkhtot[gndr == "female"], na.rm = TRUE)),
        lty = 2,
        lwd = 1,
        color = "pink") +

theme(legend.position = "none")
```

5.1.3.3 Raincloud (are better boxplots)

Warum Boxplots oft schlecht sind ist hier erklärt: https://www.cedricscherer.com/2021/06/06/visualizing-distributions-with-raincloud-plots-and-how-to-create-them-with-ggplot2/

Hier ein anschauliches Beispiel, warum und wie Boxplots Missverständnisse fördern.
erstellen eines fiktiven Datensatzes:

```
data &lt;- tibble(group = factor(c(rep(Group 1 ,100),
                        rep( Group  2   ,250),
                        rep( Group  3   ,25))),
                  value = c(seq(0,20,length.out = 100),
                      c(rep(0,5),
rnorm(30,2,.1),
rnorm(90,5.4,.1),
rnorm(90,14.6,.1),
rnorm(30,18,.1),
rep(20,5)),
rep(
seq(
0,
20,
length.out = 5
),
5
)
)
) %&gt;%
rowwise() %&gt;%
mutate(value = if_else(group == Group 2  , value + rnorm(1, 0, .4), value))
```

hier die grafik:

```
ggplot(data, aes(x = group, y = value)) +
geom_boxplot(fill = "grey92") +
geom_point(size = 2,alpha = .3,position = position_jitter(seed = 1,width = .2))
```

Um Missverständnisse mit Boxplots zu vermeiden sollte man also die Rohdaten mit Verteilung angeben. Das macht man mit dem ggdist Paket

```
install.packages("ggdist")
library("ggdist")
```

Boxplot + Verteilung:

```
ggplot(data, aes(x = group, y = value)) +
## add half-violin from {ggdist} package
stat_halfeye(
## custom bandwidth
adjust = .5,
## adjust height
width = .6,
## move geom to the right
justification = -.2,
## remove slab interval
.width = 0,
point_colour = NA
) +
geom_boxplot(
width = .12,
## remove outliers
outlier.color = NA ## ‘outlier.shape = NA‘ works as well
)
```

Boxplot + Verteilung + Rohdaten

```
ggplot(data, aes(x = group, y = value)) +

stat_halfeye(adjust = .5, width = .6, justification = -.2, .width = 0, point_colour =
    NA) +

geom_boxplot(width = .12, outlier.color = NA ) +

## Rohdatenpunkte hinzfuegen
stat_dots(
# in welche Richtung die Punkt sich tuermen sollen, probiere right einfach aus!
side = "left",
# leichtes Einruecken von geom_boxplot()
justification = 1.1,
#  Groee  der Punkte
binwidth = .25
)
```

Verschönern durch anpassen der X-Achse (whitespace entfernen)

```
ggplot(data, aes(x = group, y = value)) +

stat_halfeye(adjust = .5, width = .6, justification = -.2, .width = 0, point_colour =
    NA) +

geom_boxplot(width = .12, outlier.color = NA ) +

## Rohdatenpunkte hinzfuegen
stat_dots(side = "left", justification = 1.1, binwidth = .25) +
```

```
# Entferne white space
coord_cartesian(xlim = c(1.2, NA))
```

...Es fehlen die letzten 2 Codebeispiele der Raincloudseite...