

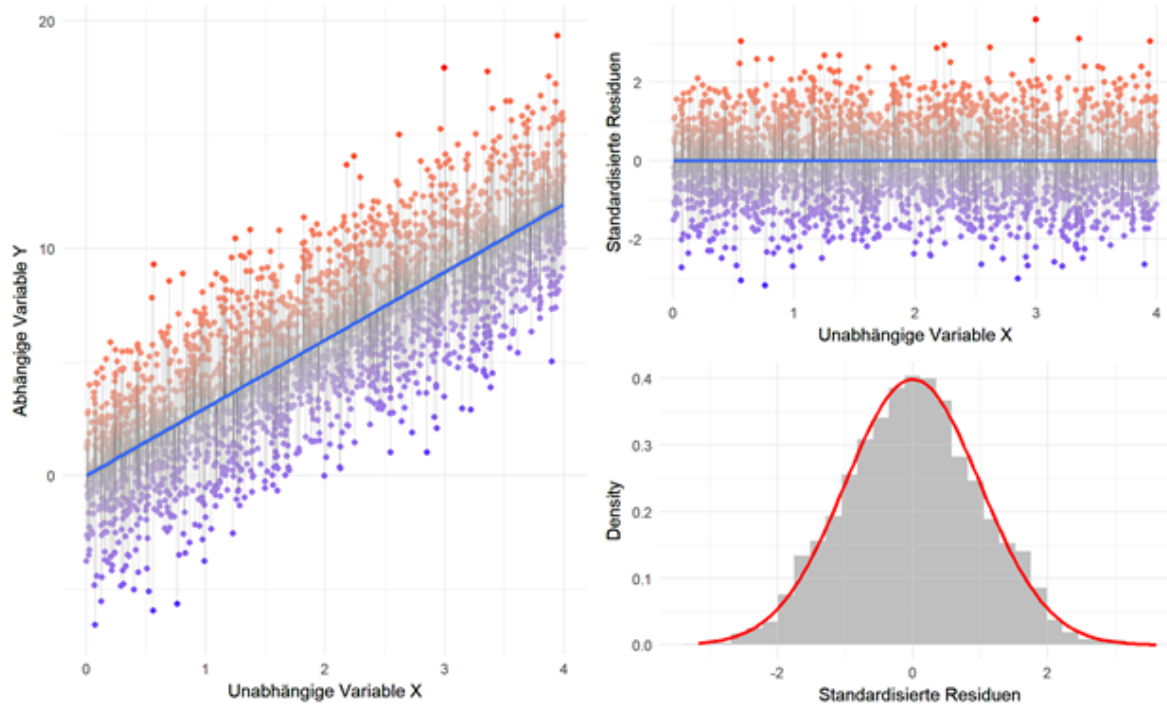


Universität Stuttgart

Abteilung IV (Prof. Urban)

Soziologie und empirische Sozialforschung

SM II: Tutorium - 2. Sitzung



Kontakt:

Fabio Votta

[favstats](#)
[favstats.eu](#)
[@FabioFavusMaxim](#)
fabio.votta@gmail.com

2018-25-10

Übersicht

1. Übungsaufgabe - SPSS
 - Wichtige Befehle
2. Übungsaufgabe - R
 - Projekte & Datenpfade
 - Hilfreiche Funktionen

[Link zum Datensatz](#)

Ziel der Übungsaufgabe

(Multivariate) Lineare Regression kennenlernen

Übungsaufgabe SPSS

[In SPSS]

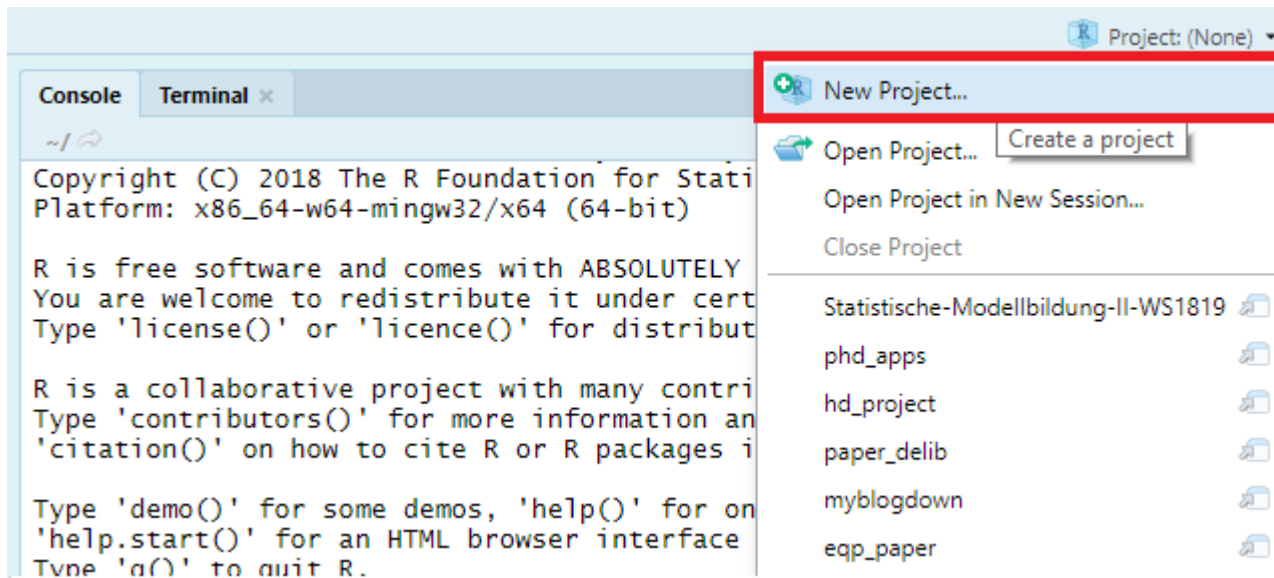
[Link zur SPSS Vorlage](#)

Übungsaufgabe R

Projekte & Datenpfade

Mit Hilfe von **Projekten** können wir Ordnung halten!

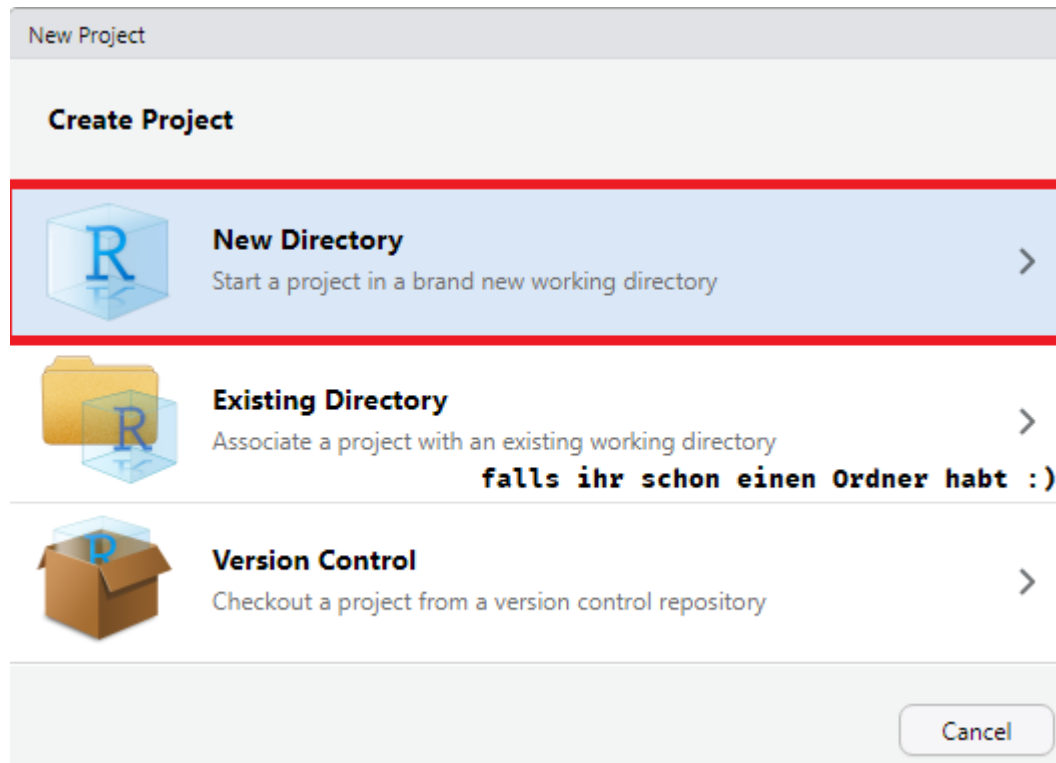
1: Neues Projekt öffnen



Projekte & Datenpfade

■ Mit Hilfe von **Projekten** können wir Ordnung halten!

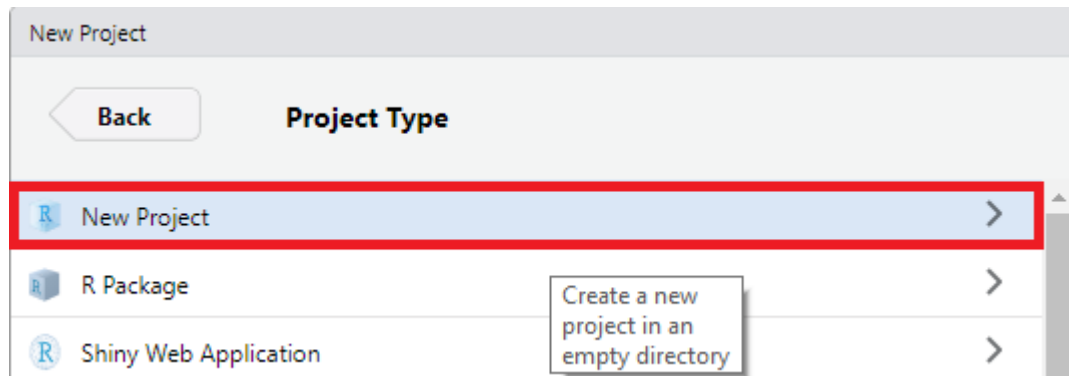
2: Auf New Directory klicken



Projekte & Datenpfade

■ Mit Hilfe von **Projekten** können wir Ordnung halten!

3: Auf New Project klicken



Projekte & Datenpfade

■ Mit Hilfe von **Projekten** können wir Ordnung halten!

4: Details eures Projektes ausfüllen

New Project

Back

Create New Project

Directory name: 1. Name des Projektordners
smll_aufgaben

Create project as subdirectory of:
~/git_proj Browse...

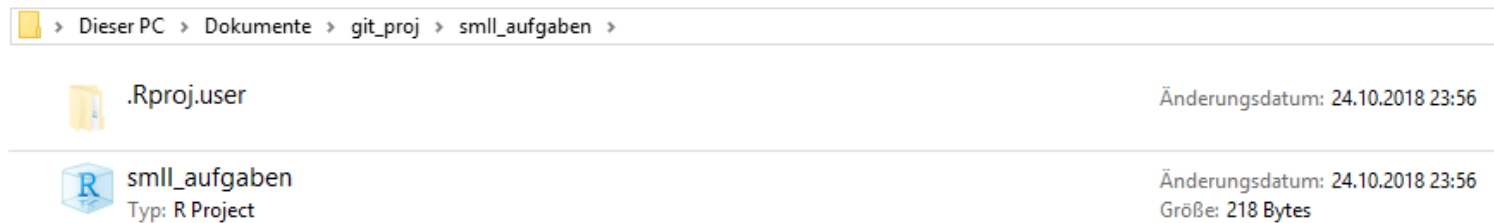
☐ Create a git repository 2. Gib einen Pfad an
☐ Use packrat with this project

3. Neues Projekt erstellen :)
☐ Open in new session Create Project Cancel

Projekte & Datenpfade

■ Mit Hilfe von **Projekten** können wir Ordnung halten!





5: So oder so ähnlich sollte euer Ordner jetzt aussehen



Projekte & Datenpfade

■ Mit Hilfe von **Projekten** können wir Ordnung halten!

6: Ordnerstruktur erstellen

» Dieser PC » Dokumente » git_proj » smll_aufgaben »		
	.Rproj.user	Änderungsdatum: 24.10.2018 23:56
	data	1. allbus2014.sav in den data Ordner verschieben Änderungsdatum: 24.10.2018 23:58
	ua1	2. 01ua_vorlage.Rmd in den 01ua Ordner verschieben Änderungsdatum: 24.10.2018 23:58
	smll_aufgaben Typ: R Project	Änderungsdatum: 24.10.2018 23:56 Größe: 218 Bytes

Projekte & Datenpfade

- In einem R Projekt ist das `Working Directory` immer dorthin gelegt wo das Projekt liegt
 - Noch genauer: Immer wo das jeweilige Skript (`.Rmd` oder `.R file`) innerhalb des Projekts liegt

Projekte & Datenpfade

- In einem R Projekt ist das Working Directory immer dorthin gelegt wo das Projekt liegt
 - Noch genauer: Immer wo das jeweilige Skript (.Rmd oder .R file) innerhalb des Projekts liegt
- So erspart man sich lange hässliche Pfade zu kopieren :)

```
getwd()
```

```
## [1] "C:/Users/Fabio/Documents/git_proj/smII_aufgaben/01ua"
```

- In unserem Fall wollen wir noch den folgenden Part zu unserem Pfad hinzufügen:

Projekte & Datenpfade

- In einem R Projekt ist das Working Directory immer dorthin gelegt wo das Projekt liegt
 - Noch genauer: Immer wo das jeweilige Skript (.Rmd oder .R file) innerhalb des Projekts liegt
- So erspart man sich lange hässliche Pfade zu kopieren :)

```
getwd()
```

```
## [1] "C:/Users/Fabio/Documents/git_proj/smII_aufgaben/01ua"
```

- In unserem Fall wollen wir noch den folgenden Part zu unserem Pfad hinzufügen:

```
## [1] "../"
```

- Dadurch springt der Pfad einen Ordner nach oben, wo sich der data Ordner befindet :)

Projekte & Datenpfade

Der riesige Vorteil: kein Datenwirrwarr und alle Pfade sind gelegt!

```
allbus <- read_sav("../data/allbus2014.sav")  
  
allbus %>% select(V1:V15) %>% head() %>% kable("html")
```

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
5240	1	1	2	1	1	2	2	2	3	2	2	1	1	2
5240	2	1	1	2	1	2	3	2	3	2	4	1	1	2
5240	3	1	2	2	1	1	2	2	3	3	5	2	2	5
5240	4	2	1	2	1	2	2	2	3	2	2	1	4	5
5240	5	1	1	1	1	2	3	3	3	3	3	1	5	5
5240	6	2	1	1	1	2	2	2	3	3	4	2	1	5

Jetzt noch ein paar hilfreiche Funktionen für die Übungsaufgabe

Hilfreiche Funktionen für Übungsaufgabe

■ Datensatz einladen

```
allbus <- read_sav("../data/allbus2014.sav")
```

■ Datensatz inspizieren mit binocularR

```
# devtools muss dazu einmal installiert werden  
install.packages("devtools")  
  
# jetzt kann binocularR über GitHub installiert werden  
devtools::install_github("systats/binocularR")
```

Anwendung von **binocularR**

```
binocularR(allbus)
```

Hilfreiche Funktionen für Übungsaufgabe

▮ Variablen auswählen

```
select(data, var1, var2, var3)
```

▮ Variablen umbenennen

```
rename(data, var1_new = var1,  
        var2_new = var2,  
        var3_new = var3)
```

▮ Neue Variable erstellen und Rekodieren

```
mutate(data, var1_new2 = var1_new + 5,  
        var2_new2 = ifelse(var2_new == 1, NA, var2_new),  
        var3_new2 = ifelse(var3_new == 1 | var3_new == 2, 1, var
```

Hilfreiche Funktionen für Übungsaufgabe

Neue Variable erstellen und Rekodieren

```
mutate(data, var1_new2 = var1_new + 5,  
        var2_new2 = ifelse(var2_new == 1, NA, var2_new),  
        var3_new2 = ifelse(var3_new == 1 | var3_new == 2, 1, var
```

Remember:

`ifelse` funktioniert nach folgender Logik:

1. Argument: logischer Test
2. Argument: was soll passieren wenn TRUE
3. Argument: was soll passieren wenn FALSE

Falls ihr nochmal Anwendungsbeispiele braucht, schaut doch in das `01_intro.Rmd` aus der ersten Sitzung :)

Alles gemeinsam mit der Pipe %>%

```
data %>%  
  select(var1, var2, var3) %>%  
  rename(data, var1_new = var1,  
          var2_new = var2,  
          var3_new = var3) %>%  
  mutate(data, var1_new2 = var1_new + 5,  
          var2_new2 = ifelse(var2_new == 1, NA, var2_new),  
          var3_new2 = ifelse(var3_new == 1 | var3_new == 2, 1, var
```

Hilfreiche Funktionen für Übungsaufgabe

▮ Variablen inspizieren

```
frq(allbus, V58)
```

```
##  
## # MUSIK: VOLKSMUSIK HOEREN (V58) <numeric>  
## # total N=3471 valid N=3464 mean=3.34 sd=1.30  
##  
##   val      label frq raw.prc valid.prc cum.prc  
##   1  SEHR GERN 323   9.31    9.32    9.32  
##   2      GERN 725  20.89   20.93   30.25  
##   3 WEDER NOCH 710  20.46   20.50   50.75  
##   4    UNGERN 863  24.86   24.91   75.66  
##   5 SEHR UNGERN 843  24.29   24.34  100.00  
##   9 KEINE ANGABE  0   0.00    0.00  100.00  
##  NA           NA   7   0.20    NA    NA
```

Lineare Regression

```
modell1 <- lm(y ~ x1 + x2, data = example_data)
htmlreg(modell1)
```

Model 1	
(Intercept)	38.15*** (3.35)
x1	-0.68 (0.35)
x2	0.02*** (0.00)
R ²	0.55
Adj. R ²	0.54
Num. obs.	100
RMSE	19.67
***p < 0.001, **p < 0.01, *p < 0.05	

Statistical models

Eine Regression spezifizieren wir in R mit der `lm()` Funktion

Syntax:

1. Die AV, hier y
2. `~` = "wird erklärt durch"
3. Die UVs `x1 + x2`
4. Zu guter letzt: der Datensatz:
`data = example_data`

Letztlich können wir uns das Modell anzeigen lassen:

- `screenreg` in Rstudio
- `htmlreg` für Webseiten
- `texreg` für PDF Reports

Lineare Regression II

Mit `list()` können wir auch mehrere Modelle nebeneinander darstellen

```
model2 <- lm(y ~ x1 + x2 + x3, data = example_data)
model3 <- lm(y ~ x1 + x2 + x3 + x4, data = example_data)

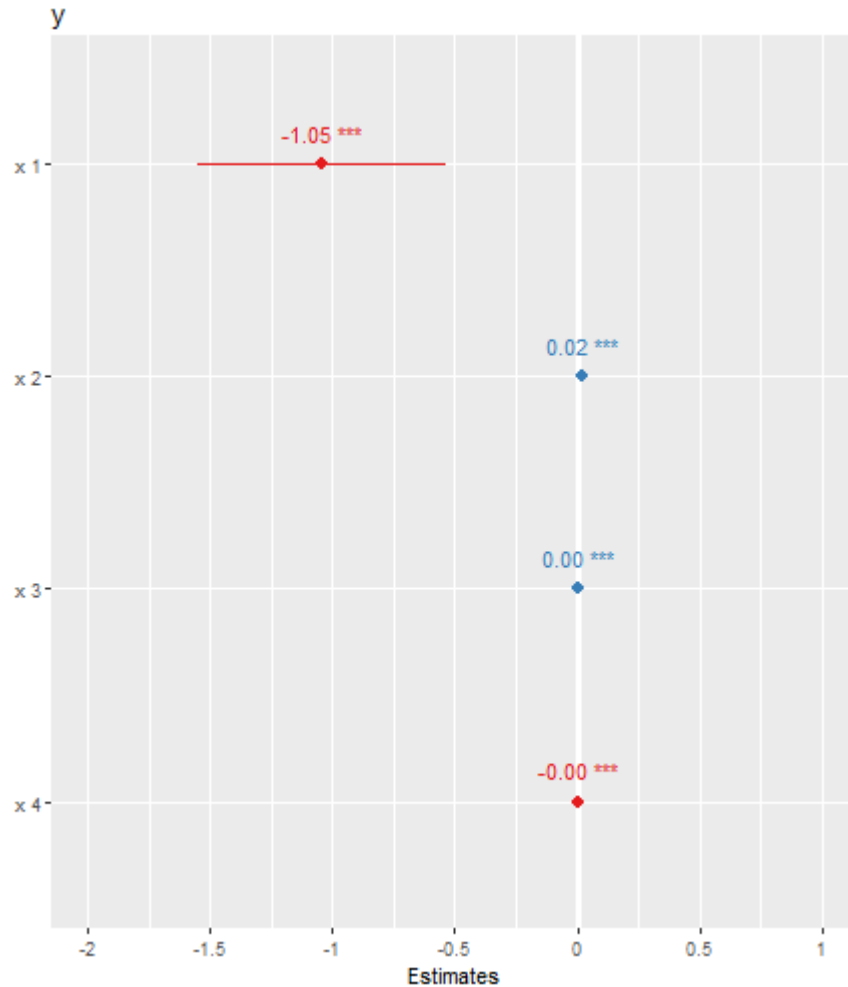
htmlreg(list(model1, model2, model3), single.row = T)
```

	Model 1	Model 2	Model 3
(Intercept)	40.90 (3.63)***	41.22 (3.60)***	39.11 (3.28)***
x1	-0.92 (0.33)**	-1.10 (0.35)**	-1.43 (0.32)***
x2	0.02 (0.00)***	0.02 (0.00)***	0.03 (0.00)***
x3		-0.00 (0.00)	0.00 (0.00)***
x4			-0.00 (0.00)***
R ²	0.52	0.53	0.62
Adj. R ²	0.51	0.52	0.61
Num. obs.	100	100	100
RMSE	20.33	20.14	18.21

***p < 0.001, **p < 0.01, *p < 0.05

Lineare Regression III

```
plot_model(model3, show.p = T, show.values = T)
```

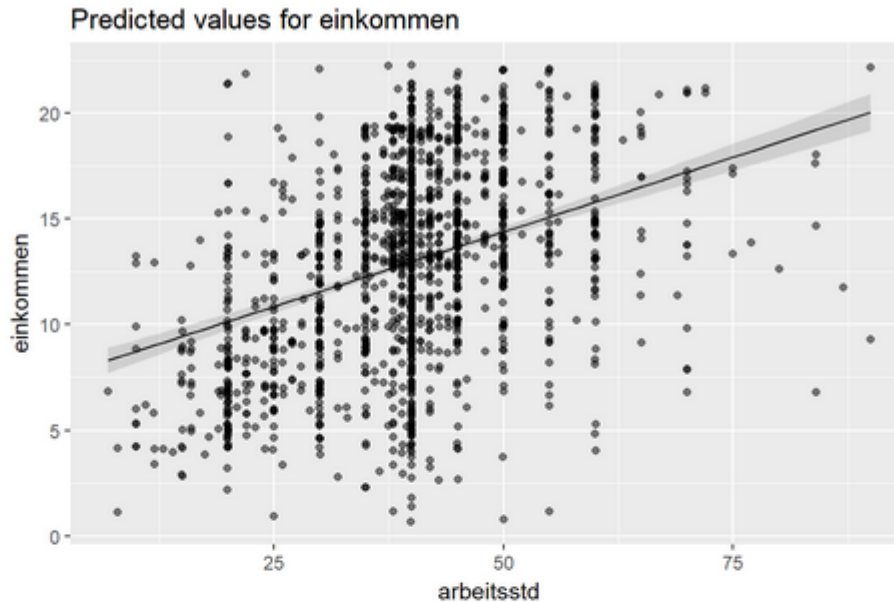


Lineare Regression IV

Mehr Anwendungsfälle für `plot_model`

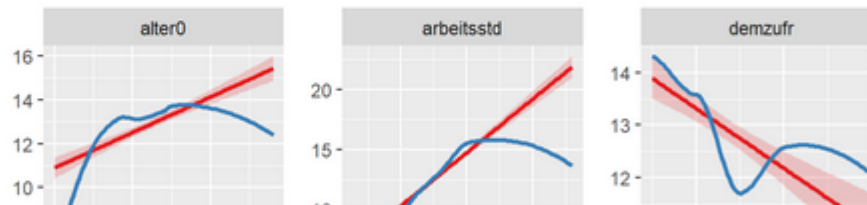
Mit `show.data = TRUE` können wir die Datenpunkte anzeigen

```
plot_model(mod1, type = "eff", terms = "arbeitsstd", show.data = TRUE)
```



Bivariate scatterplots

```
plot_model(mod1, type = "slope")
```



Jetzt können wir loslegen :)

[Link zur R Vorlage](#)