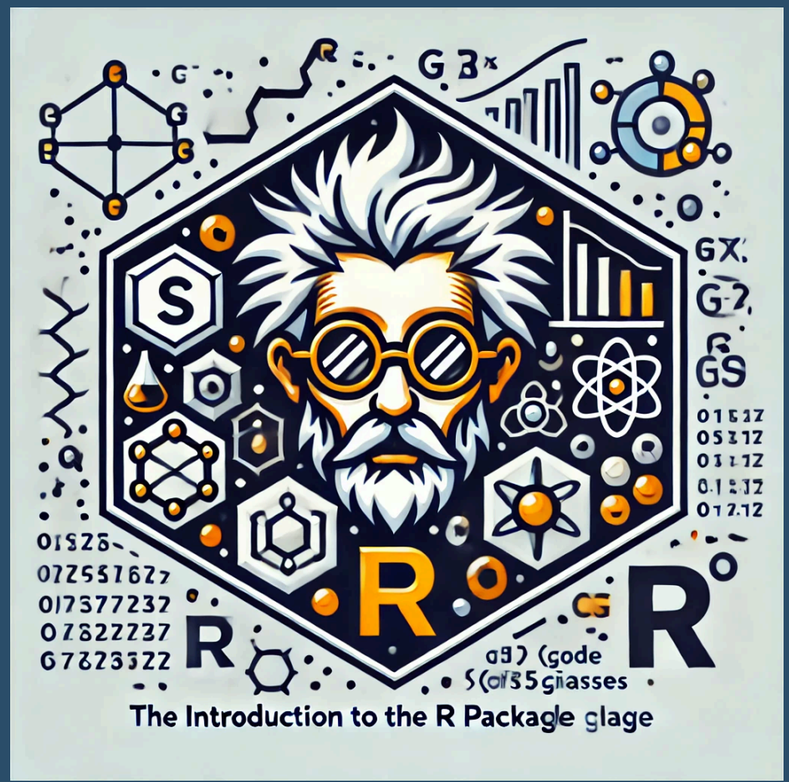
Teil V - Data Handling und Tabellen

Grundlagen der Datenanalyse und Statistik mit R | WS 2024/25

Prof. Dr. Daniel D. Schnitzlein



GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Dritter Teil von letzter Woche

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

3. Einlesen und Bereinigen von Daten

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Das Menü

- Die meisten Datensätze können interkativ über das Menü in R-Studio eingelesen werden.
- Dabei wird auch der entsprechende R-Code generiert, den Sie dann in Ihr Skript übernehmen können.
- Aufgrund der einfachen Bedienung und der komfortablen Vorschau-Funktion ist es empfehlenswert diesen Weg – mindestens – für das erste Mal des Einlesens eines Datensatzes zu wählen.
- Natürlich können alle Datensätze auch über ein Skript eingelesen werden.
- Bei spezielleren Datensätzen ist dieser Weg zu bevorzugen.
- Wir beschränken uns hier im Kurs auf “Rechtecksdaten” in den Formaten *.csv und *.xls bzw. *.xlsx

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

CSV-Datensätze (1)

R Code ⟳ Start Over ▷ Run Code

```r
library(tidyverse)
students <- read_csv("data/students.csv")
```

R Code ⟳ Start Over ▷ Run Code

```r
students
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Clean-up missing values

R Code | Start Over | Run Code

```
1 students <- read_csv("data/students.csv")
```

alternativ

R Code | Start Over | Run Code

```
1 students <- read_csv("data/students.csv", na=c("N/A", ""))
```

Einheitliche Namen der Variablen

R Code | Start Over | Run Code

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

students <- students |>
    clean_names()
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Variablentypen zuweisen

DIe Variable `meal_plan` ist eine kategoriale Variable die in R als Faktor angelegt sein sollte. Nur so kann R später für Modellierung etc. die Variablen korrekt verwenden.

R Code | Start Over | Run Code

```
1
2 students <- students |>
3   mutate(meal_plan = factor(meal_plan))
4
5 students
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Variablen bereinigen

DIe Variable age enthält aktuell Zahlen und Strings. Das wollen wir bereinigen.

R Code | Start Over | Run Code

```
1
2 students <- students |>
3   mutate(
4     age = parse_number(if_else(age == "five", "5", age))
5   )
6
7 students
```

Weitere Informationen und Optionen zum Umgang mit CSV-Files finden Sie unter ?read_csv.

Excel-Daten

- Microsoft Excel ist wahrscheinlich das meistgenutze Tabellenkalkulationsprogramm und wahrscheinlich auch das meistgenutze Programm zur (oberflächlichen) Datenanalyse.
- Die Excel-Datenformate sind daher auch häufig genutze **Output-Formate** für Datengenerierungen und Datenweitergaben (speziell in Unternehmen).

- R kann Excel-Daten u.a. über das Paket `readxl` lesen und via `writexl` schreiben. Die zugehörige Funktion ist `read_excel()`.
- R kann auch mit Stata, SPSS, SAS o.ä. Daten umgehen z.B. via des Pakets `haven`.
- Speziell auf die Arbeit mit gelabelten Daten spezialisiert ist das Paket `sjlabelled` : https://cran.r-project.org/web/packages/sjlabelled/index.html

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Excel-Daten einlesen (1)

R Code ⟳ Start Over ▷ Run Code

```
1 library(readxl)
2
3 students <- read_excel("data/students.xlsx")
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Excel-Daten einlesen (2)

R Code | Start Over | Run Code

```
1
2 students <- read_excel("data/students.xlsx",
3   col_names = c("student_id", "full_name", "favourite_food", "meal_plan", "age"),
4   skip = 1
5 )
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Excel-Daten aus Dateien mit mehreren Tabellenblättern einlesen

R Code | Start Over | Run Code

```
1
2 penguins_torgersen <- read_excel("data/penguins.xlsx", sheet = "Torgersen Island", na = "NA")
3 penguins_biscoe <- read_excel("data/penguins.xlsx", sheet = "Biscoe Island", na = "NA")
4 penguins_dream  <- read_excel("data/penguins.xlsx", sheet = "Dream Island", na = "NA")
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Daten zu einem Datensatz zusammenfügen

R Code ⟳ Start Over ▷ Run Code

```
1 penguins <- bind_rows(penguins_torgersen, penguins_biscoe, penguins_dream)
2 penguins
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Unsere Themenliste

1. **Einführung in R und R-Studio:** Überblick über die Installations- und Einrichtungsprozesse | Grundlegende Funktionen und Bedienung von R und RStudio
2. **Grundlagen der Statistiksprache R:** Syntax und Datenstrukturen in R | Einführung in Funktionen und Pakete
3. **Datenmanagement in R**: Methoden der Datenorganisation und -vorbereitung | Importieren, Bereinigen und Transformieren von Datensätzen
4. **Einführung in die Pakete des tidyverse:** Überblick über die wichtigsten tidyverse-Pakete wie z.B. dplyr und ggplot2 | Anwendung dieser Pakete zur effizienten Datenanalyse und -visualisierung
5. **Deskriptive Statistik in R:** Berechnung und Interpretation grundlegender statistischer Kennzahlen | Anwendung von deskriptiven Methoden zur Datenexploration | Einführung in die statistische Modellierung am Beispiel linearer Modelle
6. **Datenvisualisierung in R:** Erstellen von publikationsreifen Grafiken und Diagrammen mit ggplot2 | Gestaltung und Interpretation von Datenvisualisierungen zur Unterstützung der Datenanalyse

Noch offen: Einheitliche und publikationsreife Tabellen.

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Tabellen

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Verwendungszwecke von Tabellen (ähnlich zu Abbildungen)

Use-case 1. Im Datenanalyseprozess benötigen wir Tabellen um Ergebnisse zu organisieren, einen aggregierten Überblick über die Daten zu bekommen, Ergebnisse – *auf der Arbeitsebene* – zu diskutieren.

Für diesen Fall ist es sinnvoll auf eingebaute Tabellenfunktionen zurückzugreifen bzw. die entsprechenden R-Funktionen zu nutzen (z.B. `table()`). Gemeinsames Merkmal dieser Funktionen ist, dass die erzeugten Tabellen bevorzugt `in der Konsole` (manchmal im Viewer) `angezeigt werden`.

Use-case 2. Am Ende des Datenanalyseprozess benötigen wir Tabellen um die Ergebnisse – *nach extern* – zu kommunizieren. Für diesen Fall steht in R mit dem `gt()`-Paket eine Lösung zur Verfügung, die nach ähnlicher Logik wie `ggplot2` eine Erstellung druckreifer Tabellen ermöglicht. Gemeinsames Merkmal hier ist, dass die erzeugten Tabellen bevorzugt `in eigenen Dokumenten` (Word, LaTeX, HTML etc.) `verwendet werden`.

Anmerkung: Im Fall von Abbildungen entspricht *Use-case 1* der Verwendung von `plot()` und *Use-case 2* der Verwendung von `ggplot2`.

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Use-case 1. Arbeitsebene

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Deskriptive Statistiken

R Code ⟳ Start Over ▷ Run Code

```r
summary(penguins)
```

alternativ

R Code ⟳ Start Over ▷ Run Code

```r
library(psych)
describe(penguins)
```

alternativ

R Code ⟳ Start Over ▷ Run Code

```r
library(stargazer)
stargazer(as.data.frame(penguins), type= "text", title="Descriptive statistics", digits=1)
```

viele weitere Möglichkeiten: z.B. die easystats-Pakete .

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Regressionen (1)

R Code | Start Over | Run Code

```
1 penguins |>
2   lm(formula = flipper_length_mm ~ body_mass_g) |>
3   summary()
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Regressionen (2)

R Code | Start Over | Run Code

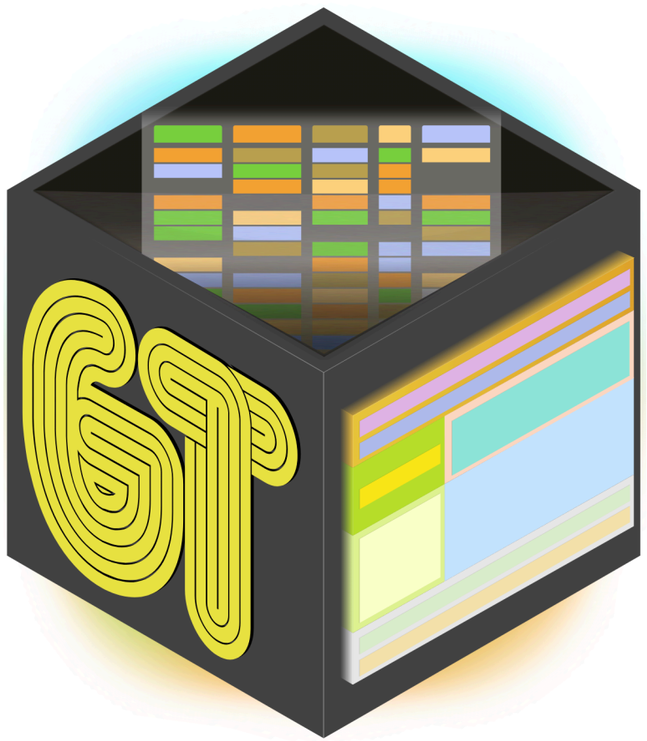
```r
model2 <- penguins |>
  lm(formula = flipper_length_mm ~ body_mass_g + bill_length_mm)

model3 <- penguins |>
  lm(formula = flipper_length_mm ~ body_mass_g + bill_length_mm + bill_depth_mm)

stargazer(model1, model2, model3, type="text")
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Use-case 2. Publikationsreife Tabellen

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Das `gt()`-Paket und `gtsummary()`



GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

`gtsummary()`

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Anwendung (1)

R Code ⟳ Start Over ▷ Run Code

```r
install.packages("gt")
install.packages("gtsummary")

library(gt)
library(gtsummary)

penguins |>
  lm(formula = flipper_length_mm ~ body_mass_g) |>
  tbl_regression()  |>
  bold_p(t = 0.10) |>
  add_glance_table(include = c(nobs, r.squared))
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Anwendung (2)

R Code | Start Over | Run Code

```r
model1 <- penguins |>
  lm(formula = flipper_length_mm ~ body_mass_g) |>
  tbl_regression()  |>
  bold_p(t = 0.10) |>
  add_glance_table(include = c(nobs, r.squared))

model2 <- penguins |>
  lm(formula = flipper_length_mm ~ body_mass_g + bill_length_mm) |>
  tbl_regression()  |>
  bold_p(t = 0.10) |>
  add_glance_table(include = c(nobs, r.squared))

model3 <- penguins |>
  lm(formula = flipper_length_mm ~ body_mass_g + bill_length_mm + bill_depth_mm) |>
  tbl_regression()  |>
  bold_p(t = 0.10) |>
  add_glance_table(include = c(nobs, r.squared))
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Anwendung (3)

R Code ⟳ Start Over ▷ Run Code

```
1 tbl_merge(
2   tbls = list(model1, model2, model3),
3   tab_spanner = c("Model 1", "Model 2", "Model 3")) |>
4   modify_table_body(~.x %>% arrange(row_type == "glance_statistic"))
```

	Model 1			Model 2			Model 3		
Characteristic	**Beta**	**95% CI**[1]	**p-value**	**Beta**	**95% CI**[1]	**p-value**	**Beta**	**95% CI**[1]	**p-value**
body_mass_g	0.02	0.01, 0.02	**<0.001**	0.01	0.01, 0.01	**<0.001**	0.01	0.01, 0.01	**<0.001**
bill_length_mm				0.55	0.39, 0.71	**<0.001**	0.59	0.45, 0.73	**<0.001**
bill_depth_mm							-1.7	-2.0, -1.3	**<0.001**
No. Obs.	342			342			342		
R^2	0.759			0.788			0.831		

[1] CI = Confidence Interval

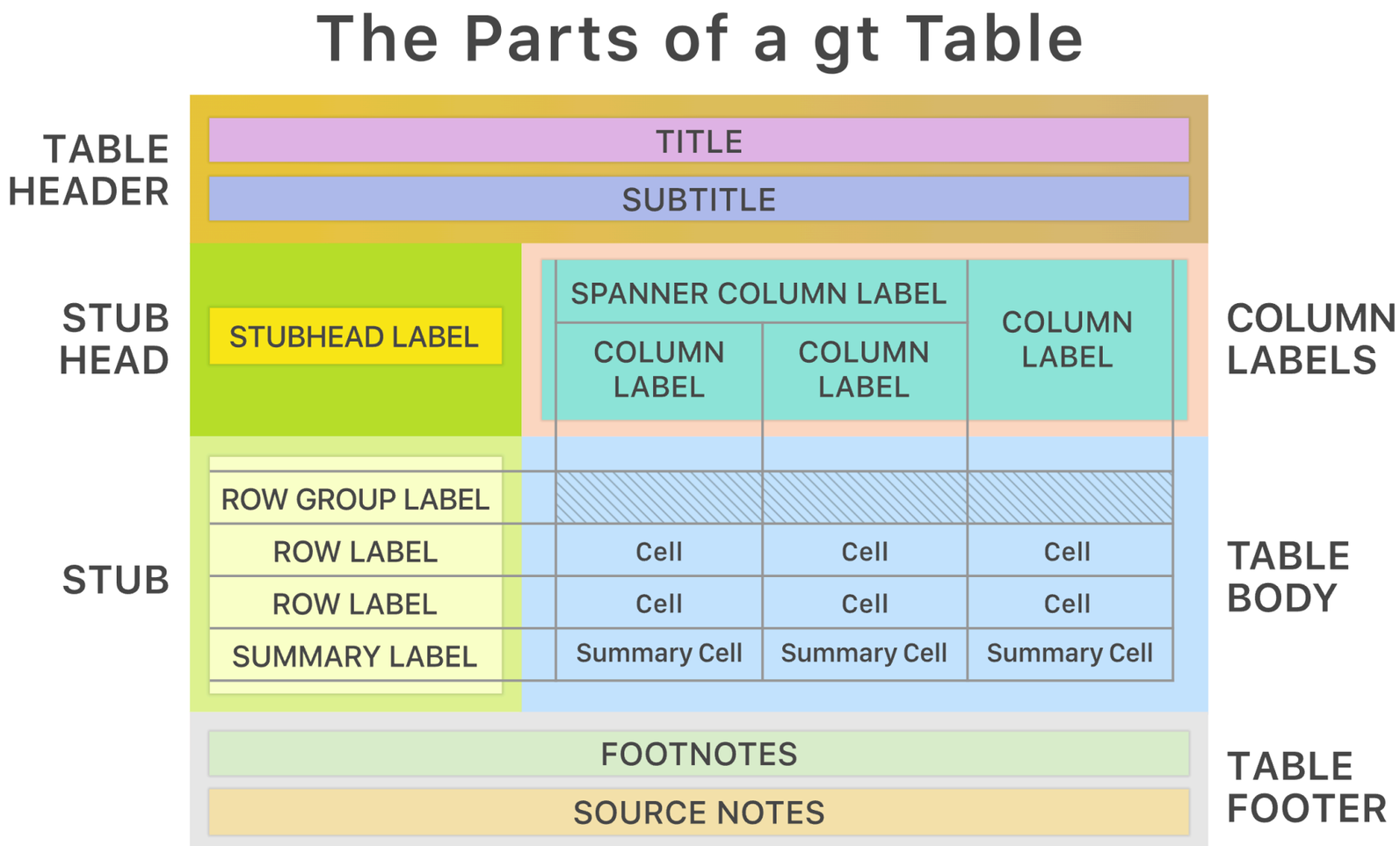
- Mit der Funktion `as_gt()` kann ein `gtsummary` Objekt in ein `gt()`-Objekt umgewandelt werden und als normale `gt()`-Tabelle bearbeitet werden.

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

```
gt()
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Elemente einer `gt()` - Tabelle



GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Anwendung (1)

R Code ⟳ Start Over ▷ Run Code

```
library(tidyverse)
library(gt)
library(gtExtras)
library(readxl)

Spotify_Top_10_GER_2025_01_15 <- read_excel("Spotify_Top_10_GER_2025-01-15.xlsx",
                                            col_types = c("numeric",
                                                          "text",
                                                          "text",
                                                          "numeric",
                                                          "date"))
```

R Code ⟳ Start Over ▷ Run Code

```
Spotify <- Spotify_Top_10_GER_2025_01_15 |>
  rename(Dauer_alt = Dauer) |>
  mutate(minutes = minute(Dauer_alt),
         seconds = second(Dauer_alt),
         Dauer = minutes*60 + seconds) |>
  select(-c(Dauer_alt))
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Anwendung (2)

R Code | Start Over | Run Code

```r
Spotify |>
  select(-c("minutes", "seconds")) |>
  gt() |>
  cols_label(Kuenstler_erster="Erste(r) Interpret(in)",
             Wiedergaben = "Anzahl Wiedergaben") |>
  tab_style(
    style = "font-weight: bold",
    locations = cells_column_labels()
  ) |>
  cols_align(align = "left",
             columns = c("Rang")) |>
  gt_plt_bar(column = Wiedergaben, color="#F29E4C", scale_type = "number", width=50) |>
  fmt_duration(
    columns = c("Dauer"),
    input_units = "seconds",
    output_units = c("minutes", "seconds"),
    duration_style = c("colon-sep"),
    use_seps = TRUE,
    sep_mark = ","
  ) |>
  tab_header(
    title = md("**Top 10 Songs auf Spotify in Deutschland**"),
    subtitle = "14. Januar 2024"
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

```
  ) |>
  tab_source_note(source_note = md(
    "Note: Die Tabelle zeigt die 10 meistgespielten Songs auf Spotify am 14. Januar 2025. Datenabruf am 15. Januar
  opt_stylize(style=3) |>
  gtsave_extra("Spotify_Table.png", vwidth=850, vheight=570, zoom=4)
```

Anwendung (3)

Top 10 Songs auf Spotify in Deutschland

14. Januar 2024

Rang	Titel	Erste(r) Interpret(in)	Anzahl Wiedergaben	Dauer
1	SABÍA QUE NO	reezy	703 112	03:03
2	APT.	ROSÉ	473 635	02:50
3	atzen & barbies (feat. ski aggu)	Shirin David	385 224	02:26
4	That's So True	Gracie Abrams	380 085	02:46
5	Die With A Smile	Lady Gaga	348 750	04:12
6	The Emptiness Machine	Linkin Park	330 194	03:10
7	Fata Morgana	Nina Chuba	285 576	02:45
8	Ma Baby	Jazeek	265 798	02:18
9	Messy	Lola Young	257 175	04:44
10	Butcher	Sosa La M	239 656	02:18

Note: Die Tabelle zeigt die 10 meistgespielten Songs auf Spotify am 14. Januar 2025. Datenabruf am 15. Januar 2025.
Quelle: Spotify.

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Wo geht's weiter?

- Introduction to creating `gt()`-Tables: https://gt.rstudio.com/articles/gt.html
- Overview and tutorials on `gtsummary`: https://www.danieldsjoberg.com/gtsummary/index.html
- Create your own table theme with the `gt()`-Package Video and Blog: https://rfortherestofus.com/2023/08/table-theme-gt
- Vollständige Berichte mit R: https://quarto.org
- Vorträge auf der `posit::conf 2024`: Link zur Playlist auf YouTube

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/

Viel Erfolg bei Ihren Analysen!!

GitHub-Repo: https://github.com/dschnitzlein/GDSR_5_Part_1/