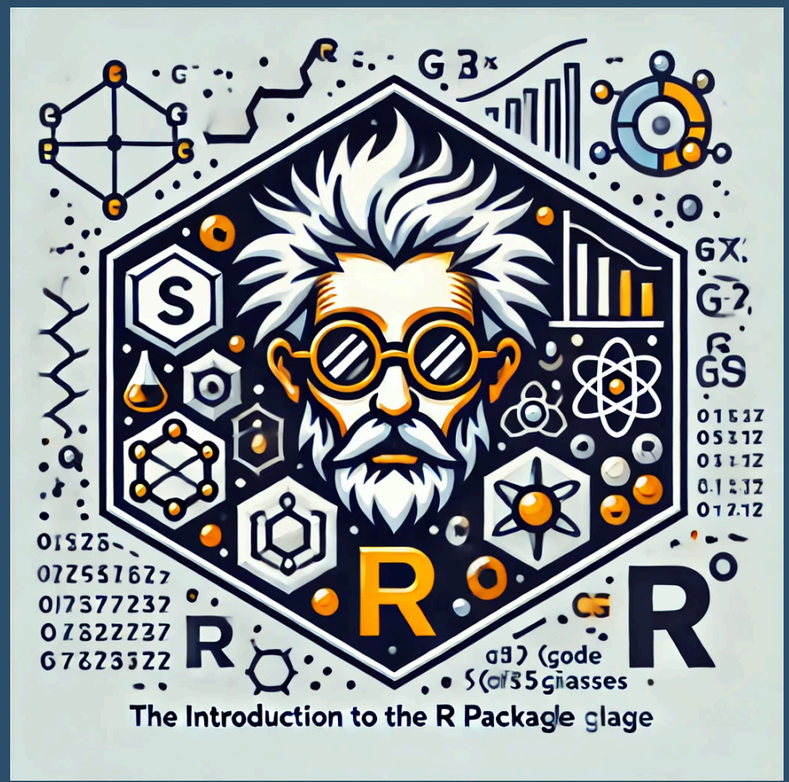
Teil IV - Themes und Data Handling

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

Prof. Dr. Daniel D. Schnitzlein



GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

1. Wiederholung aus dem letzten Jahr

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Aufgabenblatt III: Datentransformation und Modellierung

(siehe R.Skript auf Moodle und GitHub).

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

2. Themes: Erstellen von reproduzierbaren publikationsreifen Datenvisualisierungen

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Der Ausgangspunkt nach dem zweiten Termin

R Code | Start Over | Run Code

```r
install.packages('palmerpenguins')
library(palmerpenguins)
library(ggplot2) # Alternativ könnten wir auch das Meta Paket tidyverse einbinden.
library(ggthemes)
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Warum soll ich ein Theme nutzen?

- Themes haben `keinen` Einfluss darauf, wie die Daten dargestellt oder ob und wie sie transformiert werden.
- Themes ändern die Eigenschaften eines Plots `nicht`, aber sie helfen, den Plot ästhetisch ansprechend zu gestalten oder an einen bestehenden Styleguide anzupassen. Themes steuern Elemente wie z.B. Schriftarten, Ticks, Hintergründe etc.
- Themes vereinfachen es, Abbildungen ein einheitliches Erscheinungsbild innerhalb einer Publikation, innerhalb einer Organisation oder eines Web-Auftritts zu geben.
- Ein einheitliches Erscheinungsbild (das nicht einfach die Standardeinstellungen nutzt) wirkt professioneller und überzeugender verglichen mit wild zusammengewürfelten Designs.
- Individuelle Themes können in `R` vordefiniert werden und müssen daher nur einmal erstellt werden und können dann auf alle Arten von Abbildungen angewendet werden.

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Kernelemente des `ggplot2`-Theme-Systems

`elements`	Definiert das Objekt, das verändert werden soll.	z.B. `plot.title` oder `axis.ticks.x` usw. Eine vollständige Liste aller `elements` findet sich hier.
`element function`	Jedes `element` ist mit einer oder mehreren `element functions` verbunden.	z.B. `element_text(...)` definiert die Schriftgröße, Schriftart, Farbe etc. von `plot.title`.
`theme(...)`	Funktion um die Änderungen durchzuführen.	z.B. `theme(plot.title = element_text(colour = "red"))`
vollständige `Themes`	Vollständige `Themes`, die entweder bereits vordefiniert oder selbst definiert werden können.	z.B. `theme_bw( )`

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Vollständige ggplot2-Themes (1)

ggplot2 ermöglicht es anhand von Themes die Formatierung von Abbildungen weitgehend zu automatisieren (z.B. Anwendung von einheitlichem Corporate Design).

R Code | Start Over | Run Code

```r
ggplot(data = penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(aes(color = species, shape = species)) +
  geom_smooth(method = "lm") +
  labs(title = "Body mass and flipper length",
       subtitle = "Dimensions for Adelie, Chinstrap, and Gentoo Penguins",
       x = "Flipper length (mm)",
       y = "Body mass (g)",
       color = "Species",
       shape = "Species") +
  theme_economist()
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Vollständige `ggplot2`-Themes (2)

`theme_gray()`	Das charakteristische ggplot2-Design mit grauem Hintergrund und weißen Gitterlinien, das die Daten hervorhebt und Vergleiche erleichtert.
`theme_bw()`	Variation des klassische „dark-on-light"-ggplot2-Themes. Eignet sich möglicherweise besser für die Anzeige mit einem Beamer.
`theme_linedraw()`	Theme nutzt nur schwarzen Linien unterschiedlicher Breite auf weißem Hintergrund.
`theme_light()`	Ein ähnliches Theme wie `theme_linedraw()`, aber mit hellgrauen Linien und Achsen, um mehr Aufmerksamkeit auf die Daten zu lenken.
`theme_dark()`	Der dunkle Cousin von `theme_light()`, mit ähnlichen Strichstärken, aber dunklem Hintergrund.
`theme_minimal()`	Ein minimalistisches Theme ohne Hintergrund.
`theme_classic()`	Ein klassisch aussehendes Theme mit x- und y-Achsenlinien und ohne Gitterlinien.
`theme_void()`	Ein komplett leeres Theme.
`theme_test()`	Ein Theme zum Testen visueller Komponenten. Es sollte sich idealerweise nie ändern, außer bei neuen Funktionen.

Probieren Sie es aus!

R Code ⟳ Start Over ▷ Run Code

```r
ggplot(data = penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(aes(color = species, shape = species)) +
  geom_smooth(method = "lm") +
  labs(title = "Body mass and flipper length",
       subtitle = "Dimensions for Adelie, Chinstrap, and Gentoo Penguins",
       x = "Flipper length (mm)",
       y = "Body mass (g)",
       color = "Species",
       shape = "Species") +
  theme_gray()
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Anpassen der integrierten `themes`

- Alle vordefinierten Themes haben einen Parameter `base_size()` mit dem die Basisschriftgröße (Achsentitel) aller Schriften mit einem Befehl geändert werden kann.
- Dabei wird auch die Achsenbeschriftung (0.8x) und der Titel (1.2x) angepasst.
- Mit `set_theme()` kann ein bestimmtes `theme` als Standardtheme gesetzt werden.
- Alle weitergehenden Änderungen müssen über die Ansprache der entsprechenden Elemente direkt erfolgen.

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Beispiel: Änderung der Farbe im Titel

R Code | Start Over | Run Code

```r
ggplot(data = penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(aes(color = species, shape = species)) +
  geom_smooth(method = "lm") +
  labs(title = "Body mass and flipper length",
       subtitle = "Dimensions for Adelie, Chinstrap, and Gentoo Penguins",
       x = "Flipper length (mm)",
       y = "Body mass (g)",
       color = "Species",
       shape = "Species") +
  theme_bw() +
  theme(plot.title = element_text(colour = "red"))
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Beispiel: Volle Individualisierung

R Code | Start Over | Run Code

```r
ggplot(data = penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(aes(color = species, shape = species)) +
  geom_smooth(method = "lm") +
  labs(title = "Body mass and flipper length",
       subtitle = "Dimensions for Adelie, Chinstrap, and Gentoo Penguins",
       x = "Flipper length (mm)",
       y = "Body mass (g)",
       color = "Species",
       shape = "Species") +
  theme_bw() +
    theme(plot.title = element_text(colour = "red", family="Comic Sans MS"),
        plot.subtitle = element_text(colour = "blue", face="bold"),
        plot.title.position = "plot",
        axis.title = element_text(colour = "blue"),
        axis.text = element_text(colour = "purple"),
        legend.position = "top",
        legend.location = "plot",
        legend.margin = margin(0, 0, 0, 0), # turned off for alignment
        legend.justification.top = "left",
        legend.justification.left = "top",
        panel.grid.major = element_blank(),
```

GitLab Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

```
        panel.grid.minor = element_blank(),
        panel.border=element_blank(),
        axis.line.x = element_line(colour = "lightgray"),
        axis.line.y = element_line(colour = "lightgray")
    )
```

Definieren eigener Themes

R Code | Start Over | Run Code

```r
Daniels_theme <- function() {
  theme_bw() +
    theme(plot.title = element_text(colour = "red", family="Comic Sans MS"),
          plot.subtitle = element_text(colour = "blue", face="bold"),
          plot.title.position = "plot",
          axis.title = element_text(colour = "blue"),
          axis.text = element_text(colour = "purple"),
          legend.position = "top",
          legend.location = "plot",
          legend.margin = margin(0, 0, 0, 0), # turned off for alignment
          legend.justification.top = "left",
          legend.justification.left = "top",
          panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          panel.border=element_blank(),
          axis.line.x = element_line(colour = "lightgray"),
          axis.line.y = element_line(colour = "lightgray")
    )
}
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Nutzen eigener Themes

R Code ⟳ Start Over ▷ Run Code

```r

theme_set(Daniels_theme())

ggplot(data = penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(aes(color = species, shape = species)) +
  geom_smooth(method = "lm") +
  labs(title = "Body mass and flipper length",
       subtitle = "Dimensions for Adelie, Chinstrap, and Gentoo Penguins",
       x = "Flipper length (mm)",
       y = "Body mass (g)",
       color = "Species",
       shape = "Species")
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Wichtige Themeelemente: 1. Plotelemente

Element	Funktion	Beschreibung
plot.background	`element_rect()`	Hintergrund der Abbildung (ohne Daten)
plot.title	`element_text()`	Titel der Abbildung
plot.margin	`margin()`	Margins um die Abbildung
plot.title.position	“panel”, “plot”	Alignment des Titels entweder mit dem Panel oder Plot

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Wichtige Themeelemente: 2. Achsenelemente

Element	Funktion	Beschreibung
axis.line	`element_line()`	Achsenlinie (in std. Themes versteckt)
axis.text	`element_text()`	Labels der Achsenticks
axis.text.x	`element_text()`	x-Achsentickslabels
axis.text.y	`element_text()`	y-Achsentickslabels
axis.title	`element_text()`	Achsentitel
axis.title.x	`element_text()`	Achsentitel x-Achse
axis.title.y	`element_text()`	Achsentitel y-Achse
axis.ticks	`element_line()`	Achsentickstriche
axis.ticks.length	`unit()`	Achsentickstrichlänge

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Wichtige Themeelemente: 3. Legendenelemente

Element	Funktion	Beschreibung
legend.background	`element_rect()`	Legendenhintergrund
legend.key	`element_rect()`	Hintergrund der Legendeneinträge
legend.key.size	`unit()`	Größe der Legendeneinträge
legend.key.height	`unit()`	Höhe der Legendeneinträge
legend.key.width	`unit()`	Breite der Legendeneinträge
legend.margin	`unit()`	Margin um die Legende
legend.text	`element_text()`	Legenden Labels
legend.text.align	0–1	Legenden Label Alignment (0 = rechts, 1 = links)
legend.title	`element_text()`	Legendenname
legend.title.align	0–1	Legendname Alignment (0 = rechts, 1 = links)
legend.position	“top”, “bottom”, “left”, “inside”, “right”, “none”	Positionieren der Legende
legend.justification.top	“left”, “center”, “right”	Positionieren der Legende (ebenfalls für alle Pos.)

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Wichtige Themeelemente: 4. Panelelemente

Element	Funktion	Beschreibung
panel.background	`element_rect()`	Panel Hintergrund (unter den Daten)
panel.border	`element_rect()`	Panelrahmen
panel.grid.major	`element_line()`	Hauptgridlines
panel.grid.major.x	`element_line()`	Hauptgridlines vertikal
panel.grid.major.y	`element_line()`	Hauptgridlines horizontal
panel.grid.minor	`element_line()`	Minorgridlines
panel.grid.minor.x	`element_line()`	Minorgridlines vertikal
panel.grid.minor.y	`element_line()`	Minorgridlines horizontal
aspect.ratio	numeric	Plot aspect ratio

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Wichtige Themeelemente: 5. Facetinglemente

Element	Funktion	Beschreibung
strip.background	`element_rect()`	Hintergrund der Panel Strips
strip.text	`element_text()`	Strip Text
strip.text.x	`element_text()`	Horizontaler Strip Text
strip.text.y	`element_text()`	Vertikaler Strip Text
panel.spacing	`unit()`	Margin zwischen Facets
panel.spacing.x	`unit()`	Margin zwischen Facets (vertikal)
panel.spacing.y	`unit()`	Margin zwischen Facets (horizontal)

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

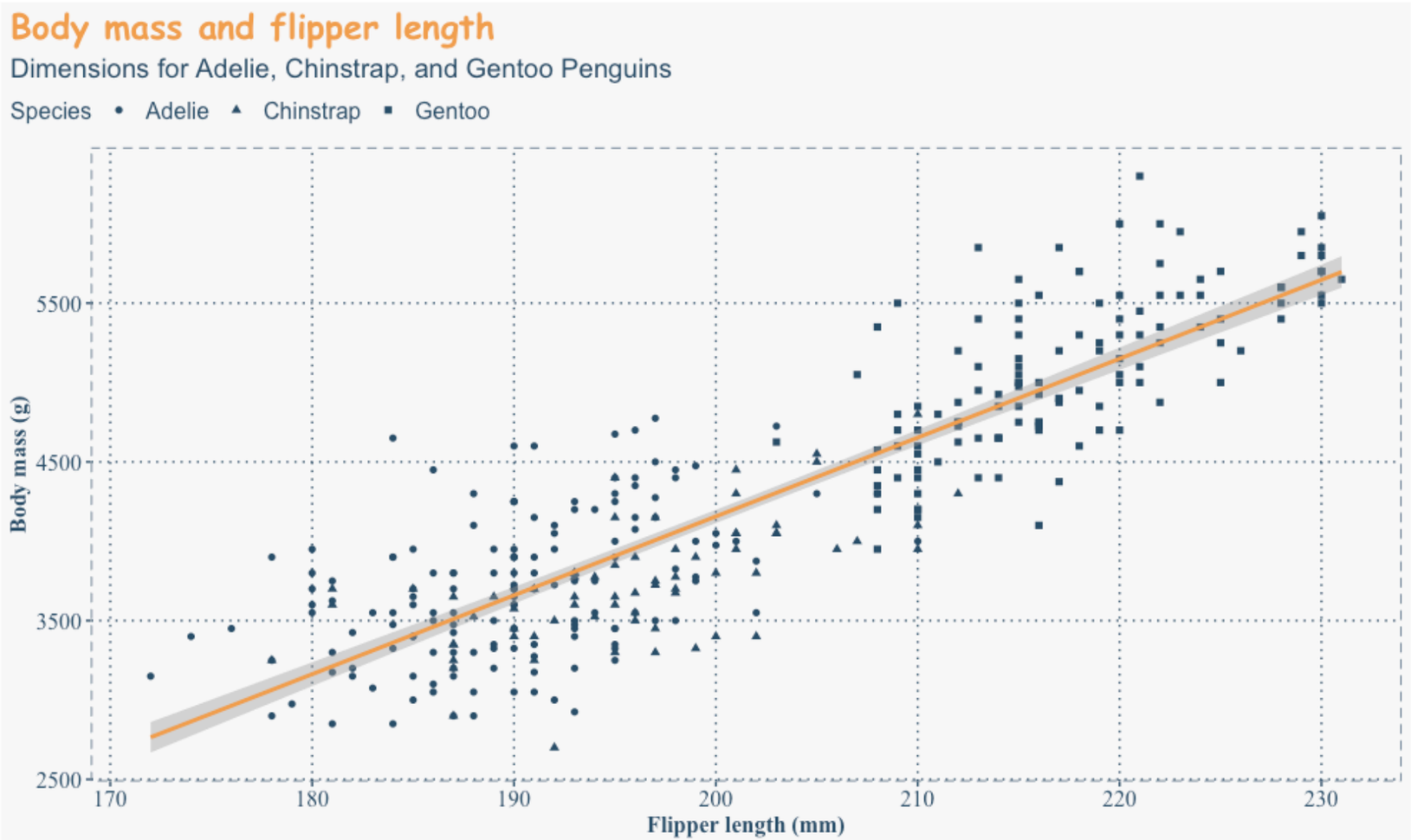
Wichtige Theme-Funktionen

`element_text(...)`	Kontrolliert Schriftart (`family`), Typ (`face`), Farbe (`colour`), Größe (`size`) usw.
`margin(0,0,0,0)`	Kontrolliert den Rand um ein Objekt
`element_line(...)`	Kontrolliert die Farbe (`colour`), Linienart (`linetype`) und Dicke (`linewidth`) einer Linie
`element_rect(...)`	Kontrolliert Rechtecke mit den Parametern `fill`, `linewidth`, `colour` und `linetype`
`element_blank(...)`	Zeichnet nichts (zum Deaktivieren von Elementen)

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Aufgabe 1

Erstellen Sie ihr eigenes Theme um die Abbildung folgendermaßen aussehen zu lassen:



GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Aufgabe 1 - Teil I

Erstellen Sie bitte ein eigenes Theme mit dem Namen `GDSR_theme`, das auf dem `theme_bw()` basiert und folgendem Design folgt:

- Die gesamte Abbildung (Plot, Legende, und Panel soll einen leicht angegrauten Hintergrund haben: #fafafa
- Alle Schriftelemente mit Ausnahme des Titels sollen ein dunkles Blau als Farbe haben: #2A4D69
- Alle Schriftelemente mit Ausnahme des Titels (Achsenbeschriftung, Achsentitel, Achsenticks, Legendentext, Legendentitel, Subtitle) sollen in **Times New Roman** gesetzt werden.
- Der Titel soll in **Comic Sans MS** gesetzt werden und folgende Farbe aus dem Design haben: #F29E4C
- Achsentitel und Abbildungstitel sollen fett gedruckt sein.
- Legendentext und Legendentitel sowie Achsentext und Achsentitel haben Schriftgröße 12. Der Titel hat Schriftgröße 18 und der Untertitel hat Schriftgröße 14.
- Das Panel enthält nur Major-Gridlines in Farbe #2A4D69 und gestrichelt (linetype 3) in size = 0.5.
- Die Abbildung hat einen Rahmen (panel.boarder) ebenfalls in Farbe #2A4D69 und mit linetype 2.
- Die Achsenticks sind ebenfalls in Farbe #2A4D69
- Die Legende soll oben links platziert sein.
- Titel und Legende sollen linksbündig mit der Achsenbeschriftung ausgerichtet werden.

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Aufgabe 1 - Teil II

R Code | Start Over | Run Code

```r
library(palmerpenguins)
library(ggplot2)
ggplot(data = penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(aes(color = species, shape = species)) +
  geom_smooth(method = "lm") +
  labs(title = "Body mass and flipper length",
       subtitle = "Dimensions for Adelie, Chinstrap, and Gentoo Penguins",
       x = "Flipper length (mm)",
       y = "Body mass (g)",
       color = "Species",
       shape = "Species")
```

Ändern Sie nun folgende Punkte an der Abbildung:

- Unterscheiden Sie die Arten von Pinguinen nur durch Shape und nicht durch Color.
- Färben Sie die Datenpunkte einheitlich in #2A4D69 ein.
- Ändern Sie die Farbe der Regressionslinie zu #F29E4C.
- Weisen Sie der Abbildung das neue Theme zu.

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

Speichern der Abbildungen

R Code | Start Over | Run Code

```r
ggsave(
  filename,
  plot = last_plot(),
  device = NULL,
  path = NULL,
  scale = 1,
  width = NA,
  height = NA,
  units = c("in", "cm", "mm", "px"),
  dpi = 300,
  limitsize = TRUE,
  bg = NULL,
  create.dir = FALSE,
  ...
)
```

`ggsave()` speichert in viele Formate, z.B. “png”, “eps”, “ps”, “tex” (pictex), “pdf”, “jpeg”, “tiff”, “png”, “bmp”, “svg”, …

Neben der Größe (Breite und Höhe) kann auch die Auflösung (dpi) oder das Farbmodel angegeben werden.

Anmerkung: Die `base`-R Lösung über `pdf()` oder `png()` ist auch mit `ggplot-Objekten möglich.

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

3. Einlesen und Bereinigen von Daten

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_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 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_4_Part_1/

CSV-Datensätze (1)

R Code ⟳ Start Over ▷ Run Code

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

alternativ

R Code ⟳ Start Over ▷ Run Code

```r
students_web <- read_csv("https://pos.it/r4ds-students-csv")
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_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", ""))
```

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/

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_4_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_4_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()`.
- Anmerkung: R kann auch mit Stata, SPSS, SAS o.ä. Daten umgehen z.B. via des Pakets `haven`.

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_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_4_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_4_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_4_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_4_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 (nächste Woche).

GitHub-Repo: https://github.com/dschnitzlein/GDSR_4_Part_1/