

# Kurzworkshop: Organisation und Dokumentation experimenteller Forschung mit Rmarkdown

Edith Scheifele (B7, Z2)

01 März 2019

## Plan für heute

12.15 - 13.30 Teil 1: Organisation und Einführung in Rmarkdown

13.30 - 13.50 Pause

13.50 - 16.00 Teil 2: Dokumentation mit Rmarkdown

## Teil 1 - Organisation

# Teil 1 - Organisation

## **Vorteile einer einheitlichen, standardisierten Ordnerstruktur**

- ▶ effizientes Arbeiten, keine Sucherei mehr
- ▶ jede Datei hat ihren Platz, keine Duplikate
- ▶ ein Projekt ist idealerweise self-contained, d.h. in sich geschlossen und *beweglich* (relative Verweise innerhalb des Projekts)

# Teil 1 - Organisation

- ▶ Wenn wir von Projekten reden, meinen wir im Allgemeinen 1 Experiment oder 1 Analyse
- ▶ R in Kombination mit RStudio: *R projects*
- ▶ Dateiendung auf *Rproj*
- ▶ ein *project* ist, wenn richtig konfiguriert, self-contained, beweglich und von anderen direkt nutzbar

# Anlegen eines R-Projects

1. Wir öffnen RStudio.
2. Wir legen ein sogenanntes *project* an, indem wir auf File > New Project klicken.
3. Dann erscheint ein Pop-up-Fenster, in dem wir New Directory > New Project auswählen.
4. Unter *Directory name* geben wir unserem Projektordner einen einschlägigen Namen und unter *Create project as subdirectory of* wählen wir *Desktop* (oder einen anderen gut wiederauffindbaren Ort) aus. Anschließend klicken wir auf *Create project*.
5. Unsere RStudio-Oberfläche erneuert sich und wir sehen rechts oben in der Ecke einen blauen Quader mit dem Namen des Projekts.

# Anlegen von Unterordnern

In unserem neuen Ordner *my\_new\_experiment*, legen wir folgende Unterordner an: *data*, *scripts* und *fig*.

- ▶ in *data* legen wir unsere Output-Dateien (txt, csv, edat) ab
- ▶ in *scripts* legen wir unsere R-Skripte ab
  - ▶ *01\_prepare.R*
  - ▶ *02\_load.R*
  - ▶ ...
  - ▶ *Dokumentation.Rmd*
- ▶ in *figures* legen wir unsere Plots ab
- ▶ ggf. weitere Ordner

## Relative und absolute Pfade

- ▶ Installation von *here*. *here* ist ein Package, das mittels einer Heuristik immer wieder herausfindet, wo sich das Projekt z.B. nach einem PC-Wechsel befindet.
- ▶ Das klingt trivial, ist aber eine wunderbare Sache!
- ▶ z.B. müssen nicht alle *setwd()*s neu gesetzt werden, wenn man auf einen anderen Laptop wechselt



## *here* in Aktion

- ▶ Wir installieren und laden *here*:

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

## here in Aktion

`here()`

- ▶ Um z.B. das Bild *lingExpGefallen.jpg* zu laden, das sich im Ordner *figures* befindet, übergibt man *here()* alle Ordner in Anführungszeichen, ab dem Ordner, der sich nicht mehr in *here()* befindet
- ▶ als letzten Bestandteil übergibt man den Namen der Datei

```
knitr::include_graphics(here('figures',  
                             'lingExpGefallen.jpg'))
```



## Teil 2 - Dokumentation mit Rmarkdown

## Teil 2 - Dokumentation mit Rmarkdown

- ▶ Rmarkdown eine an Markdown und an HTML angelehnte, leicht zu erlernende Sprache
- ▶ Dateien enden auf *Rmd*
- ▶ konzipiert für Analyse von Daten
- ▶ erleichtert das sogenannte *Literate Programming*: die Kombination von Code und Text in einem Dokument und der automatischen, dynamischen Ausgabe der Resultate
- ▶ Reproduzierbarkeit durch einen selbst und andere
- ▶ kein Copy+Paste mehr von R-Code, Plots und Resultaten von R nach Word/Latex:

<https://www.youtube.com/watch?v=s3JldKoA0zw>

## Teil 2 - Dokumentation mit Rmarkdown

**Reproduzierbarkeit** durch einen selbst und andere

*Reproducibility is defined as the ability to recompute data analytic results, given an observed data set and knowledge of the data analysis pipeline.*

**Replizierbarkeit**

*The replicability of a study is related to the chance that an independent experiment targeting the same scientific question will produce a result consistent with the original study.*

(Peng, 2015 p. 31)

# Wie legt man ein Rmd-File an?

- ▶ In RStudio -> File -> New File -> Rmarkdown
- ▶ Wir vergeben einen Titel und schreiben unseren Namen in das Autoren-Feld.
- ▶ Ferner wählen wir *HTML* als Output.
- ▶ Alle 3 Optionen lassen sich im Nachhinein noch anpassen bzw. variieren.
- ▶ Über den Button mit dem blauen Knäul lässt sich das gewünschte Dokument *knitten*.

## Aufgabe

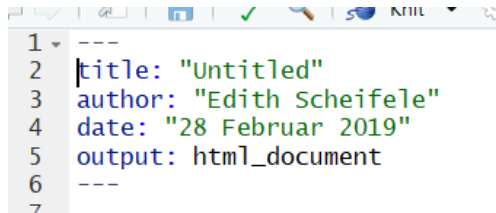
Legt wie oben beschrieben ein Rmd-File an. RStudio gibt Euch ein Dummy-File aus. Knittet dieses!

# Aufbau und Bestandteile eines Rmd-Files

Ein Rmd-File besteht im Wesentlichen aus folgenden Bestandteilen

- ▶ YAML-Header
- ▶ Chunks und Inline-Kode
- ▶ Text

# YAML-Header

A screenshot of a code editor window. The editor has a toolbar at the top with icons for undo, redo, save, and search. The code is as follows:

```
1 ---  
2 title: "Untitled"  
3 author: "Edith Scheifele"  
4 date: "28 Februar 2019"  
5 output: html_document  
6 ---  
7
```

- ▶ der Header beginnt und endet mit drei Dashes (- - -)
- ▶ alle Optionen sind als key-value pairs variierbar
- ▶ **R Markdown:: Cheat Sheet: S. 2 Set render options with YAML**
- ▶ Etwas tricky: Achtung bei der Indentierung; Doppelpunkte zeigen an, dass danach noch ein Wert kommt



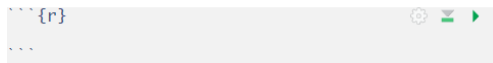
# YAML-Header - Aufgabe

1. Variiert das Output-Format
2. Fügt ein Inhaltsverzeichnis dazu
3. Probiert 1 andere Veränderung aus! Z.B. setzt die Linkfarbe auf blau.

Nutzt Google!

# Chunks

- ▶ Chunks enthalten Euren Code und werden von R ausgeführt, wenn Ihr das Dokument knittet
- ▶ Einfügen von Chunks:
  - ▶ Windows: STRG+ALT+I
  - ▶ Mac: Cmd+Option+I
  - ▶ Über das Icon *Insert* > R
- ▶ ein Chunk beginnt und endet mit 3 Backticks, wobei nach den ersten drei ein `{r}` kommt



The screenshot shows a code editor interface. On the left, there are three backticks followed by `{r}`, indicating the start of an R code chunk. On the right, there is a toolbar with three icons: a gear (settings), a green square (run), and a green right-pointing triangle (run).

# Chunks - Aufgabe

1. Platziert in unserem Dummy-Rmd-File 3 Chunks am Ende des Files:
2. Chunk: Summe aus 2 beliebigen Zahlen:
3. Chunk: eine Variable *sum\_ab*, die die Summe aus zwei beliebigen Zahlen beinhaltet
4. Chunk: einen R-Kommentar

Knittet das Dokument! Was fällt Euch auf?

# Inline-Kode

- ▶ Manchmal möchte man Variablen, die man einem Chunk berechnet hat, im Fließtext verwenden
- ▶ Vorteil: Jedesmal, wenn sich der Wert der Variablen ändert, ändert sich auch der entsprechende Wert im Fließtext
- ▶ Ihr könnt auch innerhalb des Inline-Kodes Berechnungen vornehmen wie in ganz normalen Chunks auch

```
```{r}  
sum_ab <- 2 + 3  
```
```

Die Summe von a und b ist `r sum\_ab`.

- ▶ Inline-Kode beginnt mit einem Backtick, gefolgt von einem kleinen R und einem schließenden Backtick: ``r und endet mit ``

## Inline-Kode - Aufgabe

Probiert es aus!

## Local options

- ▶ Das lokale Verhalten von Chunks steuert man über Argumente, die man direkt in den Chunk schreibt

Auswahl aus **R Markdown:: Cheat Sheet**: S. 1 *Important Chunk Options* mit gängigen Werten

- ▶ `comment = NA`
- ▶ `echo = TRUE / FALSE`
- ▶ `eval = TRUE / FALSE`
- ▶ `fig.align`, `fig.cap`, `fig.height`, `fig.width`
- ▶ `include = TRUE / FALSE`
- ▶ `message = TRUE / FALSE`
- ▶ `results = 'asis' / 'hide'`

## Local options - Aufgabe

Wir wechseln wieder in unser Dummy-File und verändern nach und nach jedes Argument.

- ▶ `comment = NA` (default: `'###'`)
- ▶ `echo = TRUE / FALSE`
- ▶ `eval = TRUE / FALSE`
- ▶ `fig.align, fig.cap, fig.height, fig.width`
- ▶ `include = TRUE / FALSE`
- ▶ `message = TRUE / FALSE`
- ▶ `results = 'asis' / 'hide'`

# Global options

- ▶ Um die Chunk-Optionen für das ganze Dokument zu setzen, verwendet man folgenden Chunk zu Beginn des Dokuments (unterhalb des YAML-Headers)
- ▶ **opts\_chunk\$set()**

```
---  
title: "Untitled"  
author: "Edith Scheifele"  
date: "28 Februar 2019"  
output: html_document  
---
```

```
```{r setup, include=FALSE}  
knitr::opts_chunk$set(echo = TRUE, comment = NA)  
```
```

- ▶ lokale Options überschreiben globale Options



# Text (und Formatierung)

- ▶ normalen Text könnt Ihr einfach in das Dokument tippen
- ▶ wichtig: Identierung macht Unterschiede

Formatierungsauswahl: **R Markdown:: Cheat Sheet: S. 2**

*Pandoc's Markdown*

- ▶ *\*kursiv\**
- ▶ **\*\*fett\*\***
- ▶ Überschriften mit verschiedenen Einbettungen: #, ## usw.
- ▶ Listen mit Bullet-Points: -

# Text (und Formatierung) - Aufgabe

Probiert die Formatierungsoptionen aus:

- ▶ kursiv, fett, Überschriften verschiedener Einbettungstiefen, Listen und noch 2 weitere aus der Liste

# Tabellen

## **R Markdown:: Cheat Sheet: S. 2 Table Suggestions**

- ▶ je nach Output (html, pdf, Word) sind andere Pakete sinnvoll
- ▶ für html eignet sich das *kable*- bzw. das *kableExtra*-Package, die wie alle anderen Packages installiert und geladen werden müssen

## **Aufgabe**

- ▶ Nehmt das Dataset *cars*, das in R pre-installiert ist, und macht mit daraus in unserem Dokument eine schön formatierte Tabelle.
- ▶ nehmt das *kable*-Package und den entsprechenden *kable*-Befehl
- ▶ Beachtet die *Chunk*-Option

## Andere Output-Formate

- ▶ andere Formate: pdf (Latex muss installiert sein), Word (MS Word muss installiert sein), shinyapps, dashboard und weitere
- ▶ Format-Vorlagen

Im **APA**-Format

viele weitere:

```
devtools::install_github("rstudio/rarticles")
```

# Aufgabe

- ▶ Legt ein Rmd-File für Euren mitgebrachten Datensatz an
- ▶ Beginnt mit der Analyse

# Resources

Roger Peng (2015): [The reproducibility crisis in science. A statistical counterattack](#)

Yihui Xie, J.J. Allaire, and Garrett Golemund (2019): [R Markdown. The Definitive Guide](#)

Christopher Gandrud (2015): [Reproducible Research with R and RStudio](#)

Garrett Golemund and Hadley Wickham (2017): R for Data Science. [Chapter 27 R Markdown](#)