

# Wissenschaftliches Arbeiten und Forschungsmethoden

## Einheit 13: Fortgeschrittenes reproduzierbares Arbeiten

25.01.2024 | Dr. Caroline Zygar-Hoffmann

# Heutige Themen

Lehrevaluation

Versionierung

Rmarkdown und Websites

Praxis

# Lehrevaluation

"Die Studierenden finden die jeweils individuell freigeschalteten Evaluationen im studynet"

The graphic features a collage background showing hands writing on paper. Overlaid text includes "GUTE LEHRE FÖRDERN! Evaluation ist JETZT." and "DEINE LEHREVALUATION im Wintersemester 2023/24". A QR code links to hsf.click/Eval-CFH. The Charlotte FreseNIUS logo is present.

**GUTE LEHRE FÖRDERN!**  
Evaluation ist JETZT.

**DEINE LEHREVALUATION**  
im Wintersemester 2023/24

Im studynet: [hsf.click/Eval-CFH](https://hsf.click/Eval-CFH)

Jetzt ausfüllen!

hsf.click/Eval-CFH

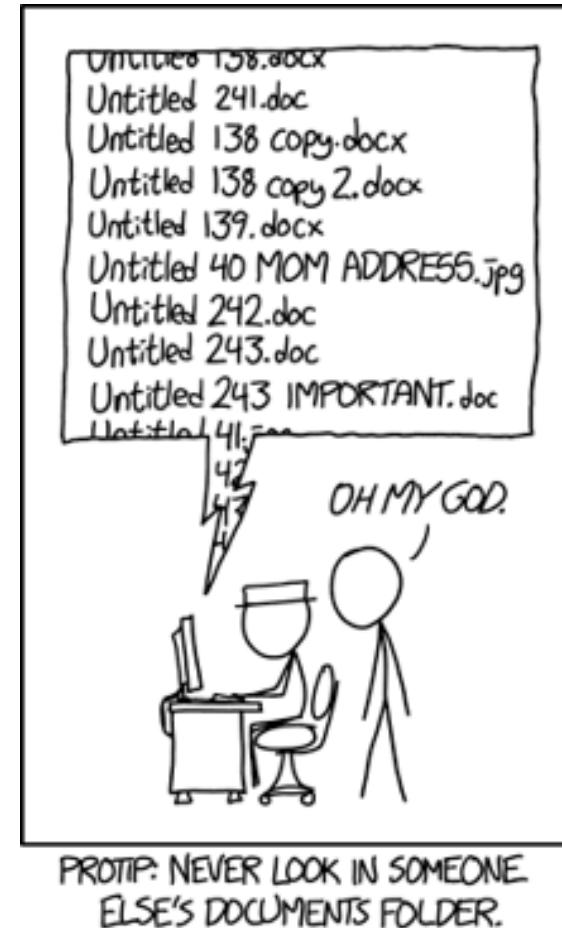
CHARLOTTE FRESENIUS  
HOCHSCHULE  
UNIVERSITY OF PSYCHOLOGY

evasys

# Versionierung

## Ziele:

- Änderungen nachvollziehen
- flexibel zu früheren Versionen zurückkehren



# Versionierung

## Git:

- kostenlose open source software für (lokale) Versionskontrolle: <https://git-scm.com/>
- Git kann "alleinstehend" in einem Terminal / Kommandozeile ausgeführt werden, dann hat man aber kein benutzerfreundliches "front-end" bzw. "user interface"
- Git kann auch in R integriert werden, sofern es vorher installiert wurde (Tools -> Global Options -> Git/SVN -> Enable version control)

## GitHub oder GitLab:

- Bieten Repositorien (bei denen Dateien gespeichert werden können), mit einem front-end, erlauben Kollaboration, beinhalten z.B. auch issue tracker
- GitHub (<https://github.com/>) gehört zu Microsoft, und ist kommerziell (auch wenn es kostenlose Nutzungsmöglichkeiten gibt)
- GitLab (<https://about.gitlab.com/>) ist kostenlos und open source, kann auf eigenen Servern gehostet werden (kostenpflichtige Nutzungsmöglichkeit über fremde Server)

## GitHub Desktop:

- Software die eine Benutzeroberfläche bietet, um Git einfach lokal *oder* mit einem Repository zu nutzen

# Versionierung

## Grundlegende Befehle in Git

### Commit:

- die aktuelle Version ("snapshot") von ausgewählten ("staged") Dateien speichern
- enthält eine "commit message", die beschreibt, welche Änderungen seit dem letzten commit gemacht wurden
- commit-Funktion ist für einfache lokale Versionskontrolle ausreichend: ersetzt manuelles Kopieren und Umbenennen von Dateien um ältere Versionen zu speichern

### Push:

- Änderungen auf ein Repository hochladen
- Nach einem push kann jeder die Änderungen sehen, der Zugriff auf das Repository hat (d.h. z.B. Sie selbst auf einem anderen Computer, Gruppenkolleg:innen, die Welt wenn es ein öffentliches Repository ist)
- Kann auch als Backup Maßnahme dienen

### Pull:

- die aktuelle Version des Repositorys herunterladen, um Änderungen von einem anderen Computer oder von Gruppenkolleg:innen lokal zu haben
- wenn eigene lokale Änderungen noch nicht gepusht wurden, erhält man von Git eine Aufforderung das zu tun

# Versionierung

## Git Ignore

Git-Projekte beinhalten häufig eine `.gitignore` Datei, welche Dateien und/oder Ordner spezifiziert, die nicht versioniert werden sollen.

Beispiele:

- irrelevante Dateien (z.B. temporäre Dateien in R)
- sensitive Daten (z.B. Zuordnungslisten)
- große Dateien, die sich normalerweise nicht ändern (z.B. Rohdaten)

Dokumentation: <https://git-scm.com/docs/gitignore>

R gitignore file

```
Raw .gitignore
1 # History files
2 .Rhistory
3 .Rapp.history
4
5 # Session Data files
6 .RData
7
8 # Example code in package build process
9 *-Ex.R
10
11 # Output files from R CMD build
12 /*.tar.gz
13
14 # Output files from R CMD check
15 /*.Rcheck/
16
17 # RStudio files
18 .Rproj.user/
19
20 # produced vignettes
21 vignettes/*.html
22 vignettes/*.pdf
23
24 # OAuth2 token, see https://github.com/hadley/httr/releases/tag/v0.3
25 .httr-oauth
26
27 # knitr and R markdown default cache directories
28 /*_cache/
29 /cache/
30
31 # Temporary files created by R markdown
32 *.utf8.md
33 *.knit.md
34 .Rproj.user
```

# Versionierung

## Sich bei Git identifizieren

- 1) Git installieren (<https://git-scm.com/>)
- 2) RStudio und Git verbinden über Tools -> Global Options -> Git/SVN
- 3) Terminal von Rstudio mit Git verbinden über Tools -> Global Options -> Terminal
- 4) im Terminal von R eintippen:
  - git config --global user.name "WUNSCHNAME"
  - git config --global user.email "EMAILADRESSE"

# Versionierung

## Git und GitHub verbinden

- GitHub Konto anlegen
- In RStudio ein "SSH RSA Key" (= ein Passwort) anlegen über Tools -> Global Options -> Git/SVN -> create SSH RSA Key
- In GitHub (im Browser) unter Settings -> SSH Key den Key aus RStudio hinterlegen (Kopieren des Keys über Tools -> Global Options -> Git/SVN -> copy SSH RSA Key)

## GitHub und GitHub Desktop verbinden

- Einloggen/Sign in (ggf. mit SSH Key)
- Anlegen eines neuen (lokalen) Repositoriums über File -> New repository, dann Option "publish repository" auswählen
- Oder: Ein Repotorium aus dem Internet kopieren ("clone": File -> Clone repository)

# Versionierung

## Demo

- Git in R
- GitHub Desktop

# Quarto, RMarkdown und Websites

- **Quarto:** aktuelle Weiterentwicklung von RMarkdown, sehr ähnliche Syntax
- Dateiendung .qmd statt .rmd
- Ermöglicht die Erstellung von Websites, auch im Kontext von R
  - Beispiel: <https://psycaroly.github.io/prereg1/>
  - Generelle Dokumentation: <https://quarto.org/docs/websites/>
- Wichtige Bestandteile einer Quarto-Website
  - \_quarto.yml: Übersicht über den Aufbau der Website ("config file")
  - einzelne qmd- (oder auch rmd-) Dateien mit Inhalten der Website; index.qmd als Startseite
- Schritt 1: Quarto installieren: <https://quarto.org/docs/get-started/>
- Schritt 2: In RStudio Projekt mit Quarto Website aufrufen über File -> New Project -> Quarto Website

# Quarto, RMarkdown und Websites

## Config file

- Ausgabeordner "out-dir" festlegen
- Navigation auf der Website festlegen
- Anzuzeigende Seiten festlegen
  - wenn man eine RMarkdown-Seite statt einer Quarto-Seite anzeigen will, muss man html angeben (siehe test.html im Screenshot) und die geknittete html in den Ausgabeordner kopieren (klassischerweise \_docs oder \_site)
- "Themes" zur Gestaltung der Website festlegen
- Man kann auch Tools einbinden, z.B. Verlinkungen zu anderen Plattformen wie Twitter/X
- Dokumentation:  
<https://quarto.org/docs/websites/website-navigation.html>

```
project:
  type: website
  output-dir: docs

website:
  title: "ESM1 preregistration website"
  navbar:
    search: true
    left:
      - text: "Home"
        file: index.qmd
      - text: "Descriptives"
        file: desc.qmd
      - text: "test"
        file: test.html
  sidebar:
    style: "docked"
    search: true
    contents:
      - section: "Research goal 1"
        contents:
          - rg1-summary.qmd
          - rg1-prereg.qmd
          - rg1-exp.qmd

format:
  html:
    theme: cosmo
    css: styles.css
    toc: true

editor: source
```

# Quarto, RMarkdown und Websites

## Tabs

::: panel-tabset :::: in einer Quarto-Datei um Überschriften herum, erstellt "klickbare" Tabs

```
## Implicit Motives
::: panel-tabset
### Implicit Agency motive

```{r, echo=FALSE}
hist(preQ.pact.dat$Agency, col = "darkblue", xlab = "Implicit Agency motive", main = "")
describe(preQ.pact.dat$Agency)
```

### Implicit Power Agency motive

```{r, echo=FALSE}
hist(preQ.pact.dat$AgencyPow, col = "darkblue", xlab = "Implicit Power Agency motive", main = "")
describe(preQ.pact.dat$AgencyPow)
```

### Implicit Independence Agency motive

```{r, echo=FALSE}
hist(preQ.pact.dat$AgencyInd, col = "darkblue", xlab = "Implicit Independence Agency motive", main = "")
describe(preQ.pact.dat$AgencyInd)
```

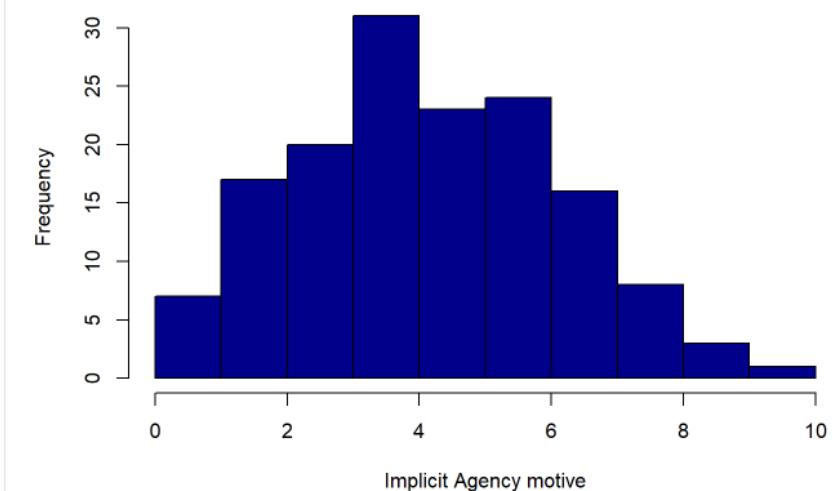
### Implicit Communion motive

```{r, echo=FALSE}
hist(preQ.pact.dat$Communion, col = "darkblue", xlab = "Implicit Communion motive", main = "")
describe(preQ.pact.dat$Communion)
```
:::
```

## Implicit Motives

Implicit Agency motive    Implicit Power Agency motive    Implicit Independence Agency motive

Implicit Communion motive



# Quarto, RMarkdown und Websites

## Website betrachten

- Lokal: In R im "Terminal" folgendes eintippen: `quarto preview` (innerhalb von R ansehen) oder `quarto render` (außerhalb von R ansehen -> im Ausgabeordner die "index.html" öffnen)
- Im Internet:
  - Verschiedene Möglichkeiten: <https://quarto.org/docs/publishing/>
  - Beispiel GitHub:
    - 1) Ausgabeordner im config-file auf "docs" festlegen
    - 2) Im RStudio Terminal vom Quarto-Projekt "copy NUL .nojekyll" eintippen
    - 3) Git installieren, sich bei Git über R identifizieren, Git im Projekt aktivieren über Tools -> Project Options -> Version Control, GitHub Konto anlegen
    - 4) Im RStudio Terminal vom Quarto-Projekt erst `quarto render` dann `git push` eintippen
    - 5) Im Browser in GitHub das Repository aufrufen, auf "public" stellen und in den Settings -> Pages -> Branch den Ordner "master -> /docs" auswählen

Die ersten Schritte durchführen, um diese Tools nutzen zu können

**1) Schritt 1:** Git installieren (<https://git-scm.com/>), mit R verknüpfen und sich identifizieren (siehe Screenshots auf studynet und Folie 8)

**2) Schritt 2:** Auswahl:

**a) GitHub Desktop** herunterladen, lokalen Ordner versionieren, eine Datei ändern, committen, und history anschauen

**b) R-Projekt mit Git** anlegen/verknüpfen, eine R-Datei anlegen, committen, etwas ändern, nochmal committen und history anschauen

**c) Quarto-Tutorial-Projekt** anlegen und entweder die Tutorial-Website gestalten, oder eine eigene Quarto-Seite (ggf. mit Studiendaten?) anlegen

# Einladung zur Podiumsdiskussion an der Bayerischen Akademie der Wissenschaften

## Podiumsdiskussion: "Wie robust kann/muss Wissenschaft sein?"

Wann: 06.02.2024 um 18:30 Wo: Bayerische Akademie der Wissenschaften, Alfons-Goppel-Str. 11 (Residenz), 80539 München

Informationen/Flyer: <https://badw.de/veranstaltungen.html?>

[tx\\_badwdb\\_events%5Baction%5D=show&tx\\_badwdb\\_events%5Bcontroller%5D=Events&tx\\_badwdb\\_events%5Bevent\\_id%5](#)

Anmeldung: <https://meta-rep.de/badw/> Live-Stream: <https://badw.de>

# Einladung zur Podiumsdiskussion an der Bayerischen Akademie der Wissenschaften

Podiumsdiskussion: "Wie robust kann/muss Wissenschaft sein?"

Die Sozial-, Verhaltens-, Lebens- und Kognitionswissenschaften sind seit einiger Zeit mit dem Problem konfrontiert, dass wissenschaftliche Ergebnisse häufig wenig robust und wiederholbar sind. Die Auswirkungen von Unsicherheit und Wiederholbarkeit von Forschungsergebnissen auf Fragen der Wissenschaft, Gesellschaft und Öffentlichkeit werden Hauptgegenstand der Diskussion sein: Wie robust können wissenschaftliche Befunde unter den aktuellen Voraussetzungen sein? Wie verlässlich müssen oder sollen Ergebnisse sein, um sie öffentlich zu kommunizieren und gegebenenfalls zum Anlass für gesellschaftliche oder politische Maßnahmen zu machen? Diskutieren werden Vertreterinnen und Vertreter der Wissenschaft, der Forschungsförderung (Deutsche Forschungsgemeinschaft) und der Medien (Süddeutsche Zeitung).

Die Veranstaltung ist öffentlich und richtet sich an Interessierte mit und ohne wissenschaftlichen Hintergrund. Veranstalter sind das junge Kolleg der Bayerischen Akademie der Wissenschaften und das von der Deutschen Forschungsgemeinschaft geförderte Schwerpunktprogramm „META-REP“. Veranstaltungsort ist die Bibliothek der Bayerischen Akademie der Wissenschaften in München (Anmeldung über: <https://meta-rep.de/badw>). Falls Sie nicht vor Ort sein können, bieten wir einen Live-Stream an, Informationen hierzu finden Sie unter <https://badw.de> (zum Live-Stream ist keine Anmeldung erforderlich).