

Springsschool 2023

Track 1

Workshop z.T. Onlinestudien, Datenanalyse & Versionierung

Merle Schuckart

E-Mail: merle.schuckart@gmx.de



Springschool 2023

Track 1

Tag 1:
Experimente bauen in lab.js
Daten erheben mit OpenLab & OSF

Tag 2:
Daten analysieren in R
Versionierung mit Github

Ablauf

Das machen wir heute:

1. Erste Schritte in lab.js
2. Wir bauen eine kleine Studie.

Mittagspause

3. Umgang mit Open Lab & OSF
4. Fragerunde / Übungsaufgaben

Das braucht ihr heute:

- Google Chrome auf eurem Laptop
- einen Open Lab Account
- ggf. einen OSF-Account

Wieso benutzen wir heute lab.js?

	PsychoPy	lab.js
Programmiersprache im Builder	Python	JavaScript
Muss man ein Programm installieren?	ja	nein (nur Google Chrome)
Muss man viel selbst programmieren?	eher ja	eher nein
Muss der Code für Onlinestudien noch in eine andere Programmiersprache übersetzt werden?	ja	nein
Zugehöriges Hosting-Tool	Pavlovia	OpenLab

Erste Schritte in lab.js

Builder:
Seite, auf der man die Studien baut

Documentation:
Handbuch mit allen Funktionen und
Tutorials

Support:
Link zum Slack-Channel

[lab.js](#) Overview

[Builder](#) [Documentation](#) [Support](#) [Resources](#) ▾ 

Online
research
made easy

lab.js is a free, open, online study
builder for the behavioral and cognitive
sciences. (it works great in the lab, too)



Erste Schritte in lab.js

The screenshot shows the lab.js application interface. On the left, there's a vertical toolbar with a blue play button, a save icon, a dropdown menu, and a settings icon. Below the toolbar is a large empty white space with a plus sign. On the right, there's a main content area with a header bar containing a heart icon and a gear icon. The main text in the center says "Welcome!" followed by "Thank you for using lab.js! We hope you find it useful, and that you enjoy using it as much as we did building it." At the bottom, there are three cards: "Get started" with a map icon, "Learn more" with a graduation cap icon, and "Find support" with a circular icon.

+

Welcome!

Thank you for using lab.js!
We hope you find it useful, and that
you enjoy using it as much as we did building it.

Get started

Learn more

Find support

Erste Schritte in lab.js

The screenshot shows the lab.js interface. At the top left are four icons: a play button, a save icon, a dropdown arrow (highlighted with an orange box), and a settings gear. Below these are two input fields with a plus sign. In the center, the word "Welcome!" is displayed in large bold letters. Below it, a message reads: "Thank you for using lab.js! We hope you find it useful, and that you enjoy using it as much as we did building it." At the bottom, there are three buttons: "Get started" with a map icon, "Learn more" with a graduation cap icon, and "Find support" with a circular icon.

A large orange arrow points from the top right towards the dropdown menu. The menu itself is also outlined in orange and contains the following items:

- Study
- New
- Open
- Save
- Export for local use
- Offline data collection
- Deploy study online
- Generic web host... (PHP backend)
- Upload to Netlify... (cloud provider)
- Upload to Open Lab...
- Export as integration
- Generic survey tools... (Qualtrics, etc.)
- JATOS (Just Another Tool for Online Studies)
- Pavlovia
- The Experiment Factory... (v3)
- Generate metadata
- Psych-DS sidecar template (JSON-LD)

Erste Schritte in lab.js

The screenshot shows the lab.js application interface. At the top left are four icons: a play button, a save icon, a dropdown arrow, and a settings gear. Below these are two buttons: a plus sign and a minus sign. In the center, there's a 'Welcome!' message with a thank you note: "Thank you for using lab.js! We hope you find it useful, and that you enjoy using it as much as we did building it." At the bottom are three cards: "Get started" with a map icon, "Learn more" with a graduation cap icon, and "Find support" with a circular icon.

Studie als json-File speichern

Studie als Laborstudie exportieren

Studie als Onlinestudie exportieren

Studie als Onlinestudie für Upload auf Pavlovia exportieren

The right side of the image shows a detailed view of the export menu, which is a vertical list of options:

- Study
- New
- Open
- Save**
- Export for local use
- Offline data collection
- Deploy study online
- Generic web host... (PHP backend)**
- Upload to Netlify... (cloud provider)
- Upload to Open Lab...
- Export as integration
- Generic survey tools... (Qualtrics, etc.)
- JATOS (Just Another Tool for Online Studies)
- Pavlovia**
- The Experiment Factory... (v3)
- Generate metadata
- Psych-DS sidecar template (JSON-LD)

Orange arrows point from the text labels above to the corresponding menu items. A red arrow points from the red text label at the bottom to the 'Pavlovia' item in the menu.

Erste Schritte in lab.js

The screenshot shows the lab.js interface. At the top left is a toolbar with a play button, a save icon, a dropdown menu, and a settings icon (highlighted with an orange box). Below the toolbar is a large 'Welcome!' message. Underneath it, a thank you message reads: "Thank you for using lab.js! We hope you find it useful, and that you enjoy using it as much as we did building it." At the bottom are three buttons: "Get started" (map icon), "Learn more" (graduation cap icon), and "Find support" (globe icon).

The 'Study information' panel is shown in a detailed view. It includes fields for "Title" and "Description". A question "What does your study do?" is present. The "Repository" field contains links to OSF and GitHub. The "Contributors" field lists Rita Levi-Montalcini with an email and URL. An orange arrow points from the settings icon in the toolbar to this panel.

Study information
Please tell your fellow scientists a few things about your study.

Title

Description

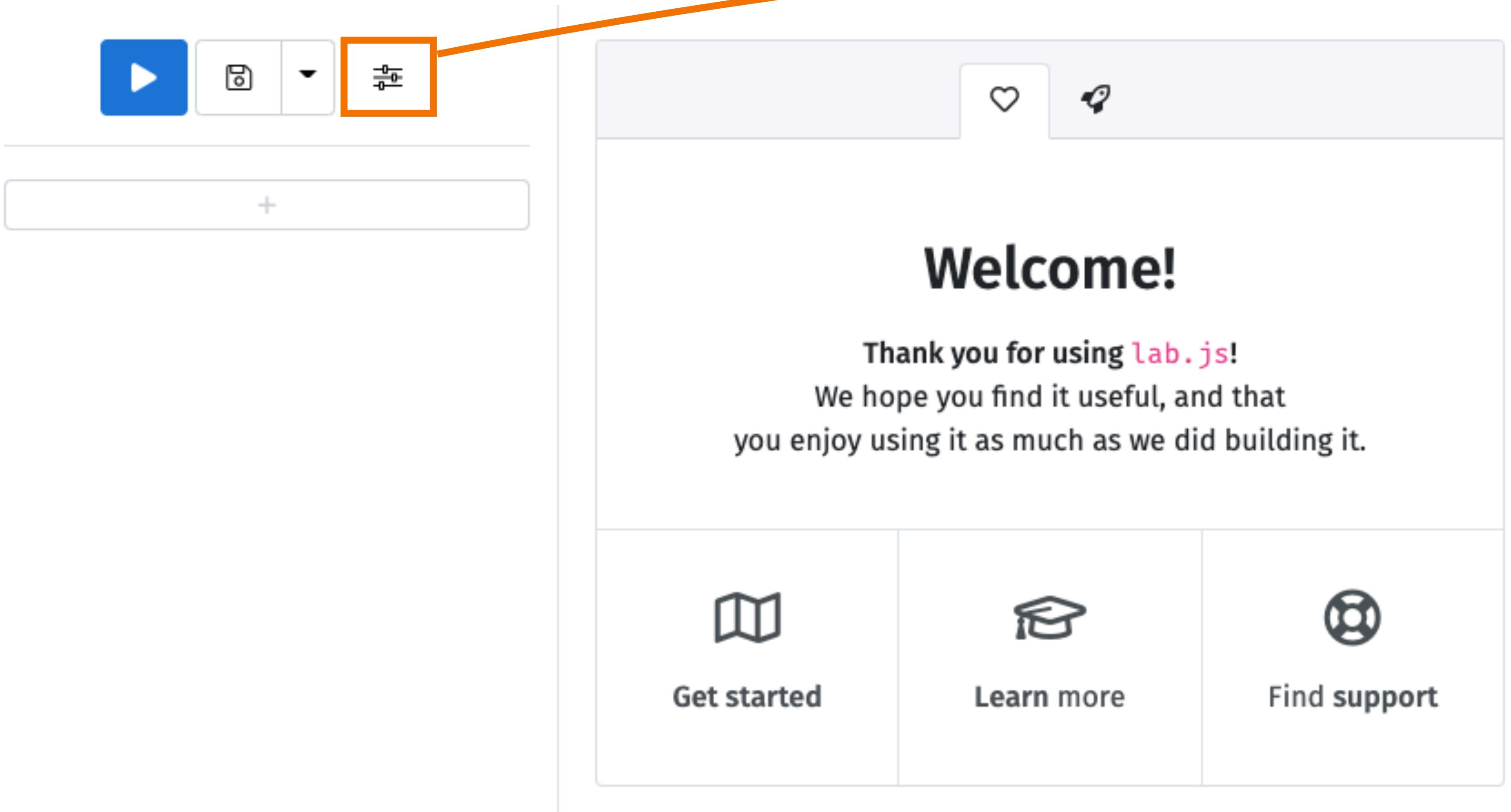
What does your study do?

Repository [https://osf.io/...](https://osf.io/) / [https://github.com/...](https://github.com/)

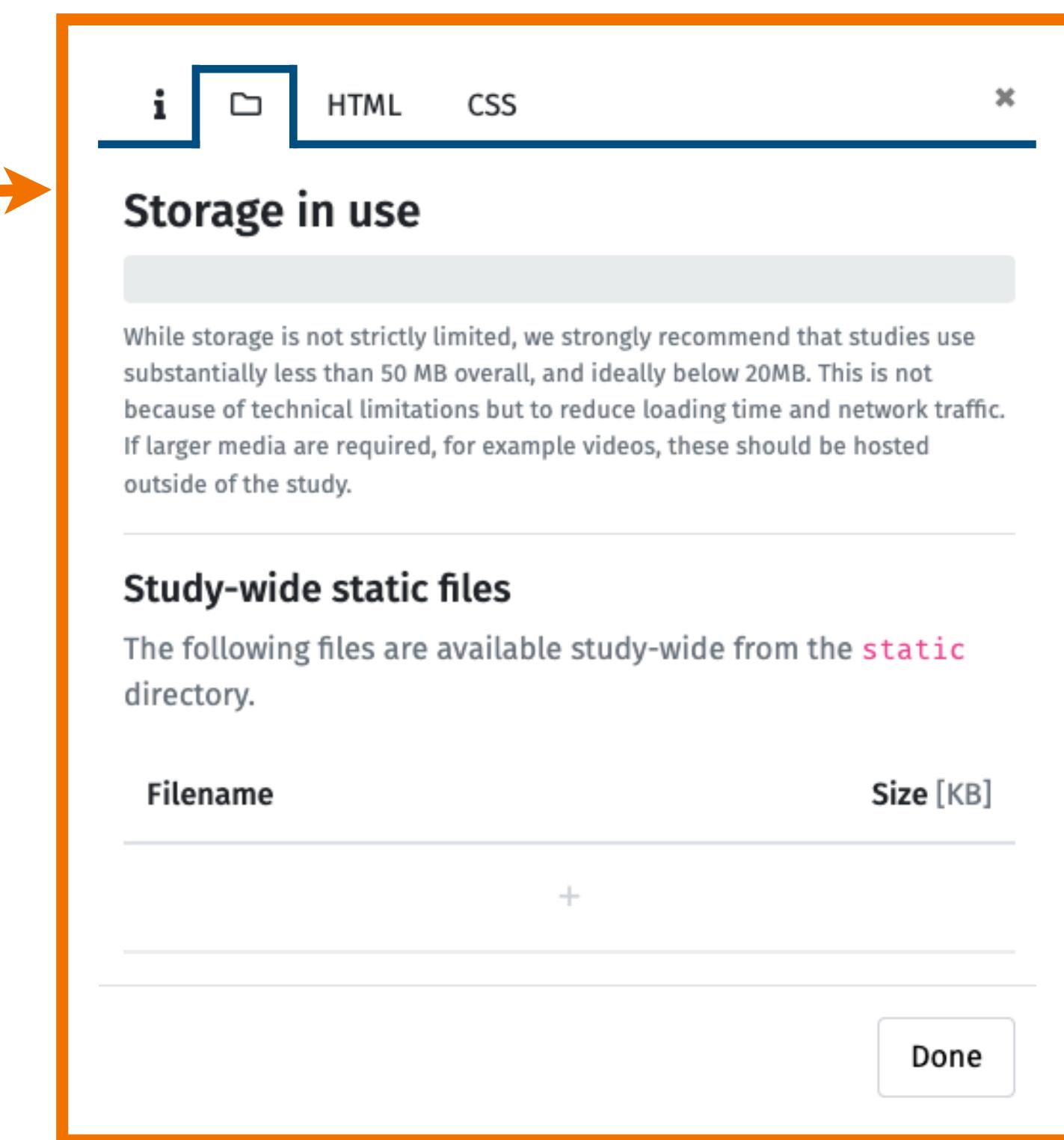
Where can researchers find the canonical or latest version of your study?

Contributors Rita Levi-Montalcini <rlm@nobel.example> (<http://ibcn.cnr.it>)

Erste Schritte in lab.js



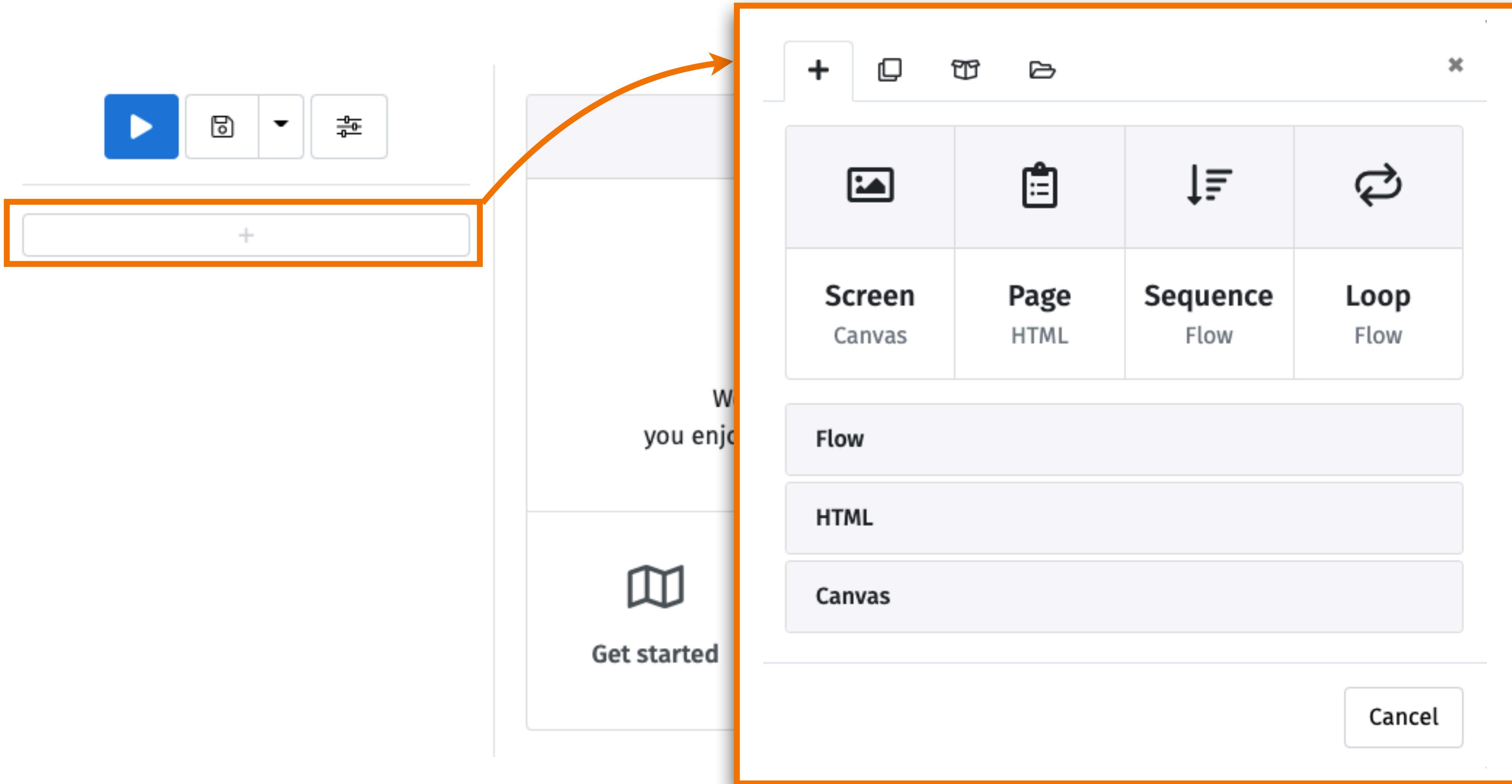
The screenshot shows the main interface of the lab.js application. At the top, there are several icons: a play button, a save icon, a dropdown menu, and a settings icon (highlighted with an orange box). Below these are two small interactive icons: a heart and a rocket. The central part of the screen features a large "Welcome!" heading and a message: "Thank you for using lab.js! We hope you find it useful, and that you enjoy using it as much as we did building it." At the bottom, there are three buttons with icons: "Get started" (map icon), "Learn more" (graduation cap icon), and "Find support" (globe icon).



The screenshot shows a detailed view of the study's storage usage. It includes a header with tabs for "Info" (selected), "File", "HTML", and "CSS". The main content area is titled "Storage in use" and contains a note about recommended file sizes. Below this is a section for "Study-wide static files" with a table header for "Filename" and "Size [KB]". A "Done" button is located at the bottom right.

Filename	Size [KB]
(empty)	(empty)

Erste Schritte in lab.js



Erste Schritte in lab.js

The screenshot shows the lab.js application interface. At the top, there are several icons: a play button, a save icon, a dropdown menu, and a settings gear. Below these is a toolbar with a large orange-bordered 'plus' button. To the right of the toolbar is a header bar with a heart icon and a rocket ship icon. The main content area features a large 'Welcome!' heading, a thank you message, and three calls-to-action: 'Get started' (map icon), 'Learn more' (graduation cap icon), and 'Find supp' (globe icon). The entire interface is framed by a light gray border.

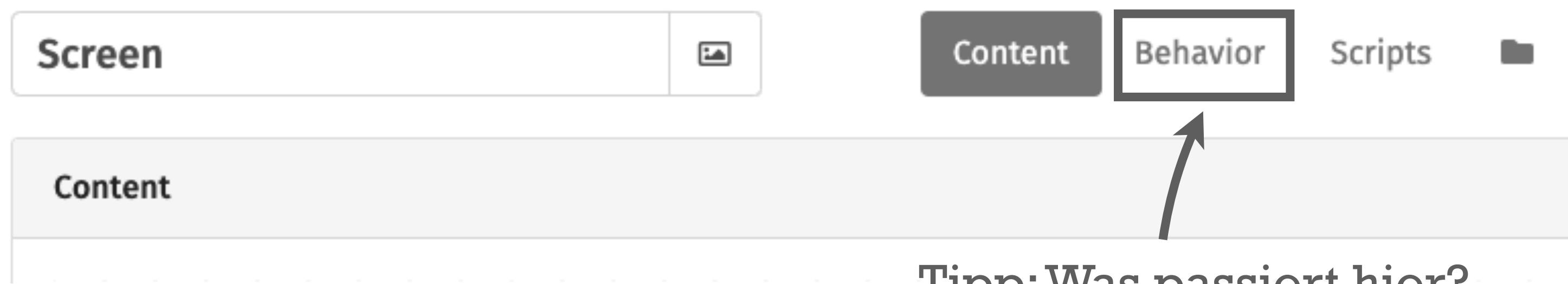
The screenshot shows the component palette of lab.js. It has a grid of four categories: 'Screen' (Canvas), 'Page' (HTML), 'Sequence' (Flow), and 'Loop' (Flow). Below these are sections for 'Flow', 'HTML', 'Form', 'Frame', 'Screen', 'Page', 'Canvas', 'Frame', and 'Screen'. Each item in the list includes a small icon, the component name, a brief description, and a settings gear icon. The entire palette is enclosed in a white box with an orange border.

Category	Component	Description	Action
Screen	Screen - Canvas	Show content using a canvas	gear
	Screen - HTML	Show content using HTML	gear
Page	Page - HTML	Graphical page builder	gear
	Page - Canvas	Provide a common canvas for nested components	gear
Sequence	Sequence - Flow	Show components sequentially	gear
	Sequence - HTML	Create a common frame around nested content	gear
Loop	Loop - Flow	Repeat a component	gear
	Loop - HTML	Collect HTML form data	gear
Flow	Flow - HTML	-	gear
	Flow - Canvas	-	gear
HTML	HTML - Flow	-	gear
	HTML - Canvas	-	gear
Form	Form - Flow	-	gear
	Form - Canvas	-	gear
Frame	Frame - Flow	-	gear
	Frame - Canvas	-	gear
Screen	Screen - Flow	-	gear
	Screen - Canvas	-	gear
Page	Page - Flow	-	gear
	Page - Canvas	-	gear
Canvas	Canvas - Flow	-	gear
	Canvas - HTML	-	gear
Frame	Frame - Flow	-	gear
	Frame - Canvas	-	gear
Screen	Screen - Flow	-	gear
	Screen - HTML	-	gear

Aufgabe für Euch

Baut eine Studie, in der zuerst eine multiple Choice Frage angezeigt wird und danach eine Seite mit einem roten Kreis in der Mitte.

Für Schnelle: Baut nach der Seite mit dem roten Kreis noch eine weitere Seite ein, auf der ein blauer Kreis angezeigt wird. Der rote Kreis sollte für 1 Sekunde angezeigt werden, bevor der blaue Kreis erscheint.



Wichtig: Merkt euch, wie ihr eure Studie gebaut habt und erzählt mir Schritt für Schritt, wie ich das Ganze nachbauen kann.

Erste Schritte in lab.js

Wir wollen nicht nur einen Trial bauen,
sondern ganz viele.

Wie könnte man das umsetzen?



Erste Schritte in lab.js

The screenshot shows the lab.js application interface. On the left, there's a toolbar with a play button, a save icon, and other controls. Below it is a large input field with a plus sign. The main area features a "Welcome!" heading and a message of thanks. At the bottom, there are three buttons: "Get started" (map icon), "Learn more" (graduation cap icon), and "Find supp" (globe icon). A red arrow points from the input field towards the right sidebar. The top right corner of the main area has a red border.

The screenshot shows the lab.js component library. It has a header with icons for Screen, Page, Sequence, and Loop. The "Loop" icon is highlighted with a blue border. Below is a section titled "Flow" containing a list of components: "Loop · Repeat a component" (selected), "Sequence · Show components sequentially", "HTML", "Form · Collect HTML form data", "Frame · Create a common frame around nested content", "Screen · Show content using HTML", "Page · Graphical page builder", "Canvas", "Frame · Provide a common canvas for nested components", and "Screen · Show content using a canvas". Each item has a small edit icon to its right.

Erste Schritte in lab.js

The screenshot shows the lab.js interface with a 'Loop' block selected. The 'Content' tab is active. Inside the loop, there are two parameters: 'parameter0' (set to 'A') and 'color' (set to 'A'). Below the loop, there is a 'Sample' block with 'Use all' selected and 'In random order' checked.

Hier kommt alles rein, was ihr wiederholen wollt

UVn + ihre Faktorstufen (Bedingungen) + ihr Datentyp

Wie viele Wiederholungen (= Trials) wollt ihr?

Wie sollen eure Trials abgespielt werden? (z.B. random, Ziehen mit/ohne Zurücklegen, als geordnete Sequenz)

Aufgabe für Euch

Baut eine Studie, in der in zufälliger Abfolge ein roter oder ein blauer Kreis angezeigt werden (insg. 10 Trials, 5 x rot, 5x blau).

Tipp 1: Nutzt einen Loop mit einer Variable „color“

Tipp 2: Nutzt statt des Farbnamens \${parameters.color}

Wichtig: Merkt euch, wie ihr eure Studie gebaut habt und erzählt mir Schritt für Schritt, wie ich das Ganze nachbauen kann.

Aufgabe für Euch

Benutzt die gleiche Studie wie eben.

Zwischen den Kreisen sollte aber diesmal jeweils
für 1500 ms eine leere Seite angezeigt werden.

Die Kreise sollten je nur für 30 ms angezeigt werden.

Tipp:

In einen Loop passt nur 1 Seite. Wie könnt ihr trotzdem 2 Seiten unterbringen? Sucht nach einer zusätzlichen Komponente, die das Problem löst!

*Wichtig: Merkt euch, wie ihr eure Studie gebaut habt
und erzählt mir Schritt für Schritt, wie ich das Ganze nachbauen kann.*

Aufgabe für Euch

Benutzt die gleiche Studie wie eben.

Wir möchten jetzt, dass unsere VP auf die Leertaste drückt, wenn ein roter Kreis angezeigt wurde, und nicht reagiert, wenn ein blauer Kreis angezeigt wurde.

Wir möchten die Reaktion als „saw_red_circle“ speichern.

Tipp 1: Die VP sollte in der Pause zwischen den Kreisen reagieren.

Tipp 2: Schaut im Tab „Behavior“ nach.

Tipp 3: Die Leertaste heißt Space.

Wichtig: Merkt euch, wie ihr eure Studie gebaut habt und erzählt mir Schritt für Schritt, wie ich das Ganze nachbauen kann.

Gruppenarbeit

Wir bauen eine kleine Studie zusammen!
(Ich baue & ihr sagt mir, was ich tun soll.)

Experiment: Redundant Signals Effect

Es wird zufällig entweder ein weißer Kreis angezeigt oder kein Kreis.

Zusätzlich wird ein Klickgeräusch abgespielt oder kein Geräusch.

Es gibt also 3 Bedingungen:

V (weißer Kreis), A (Geräusch) und VA (weißer Kreis + Geräusch)

Die VP soll reagieren, wenn sie ein Klicken gehört, einen Kreis gesehen oder beides gleichzeitig wahrgenommen hat.

Springschool 2023

Track 1

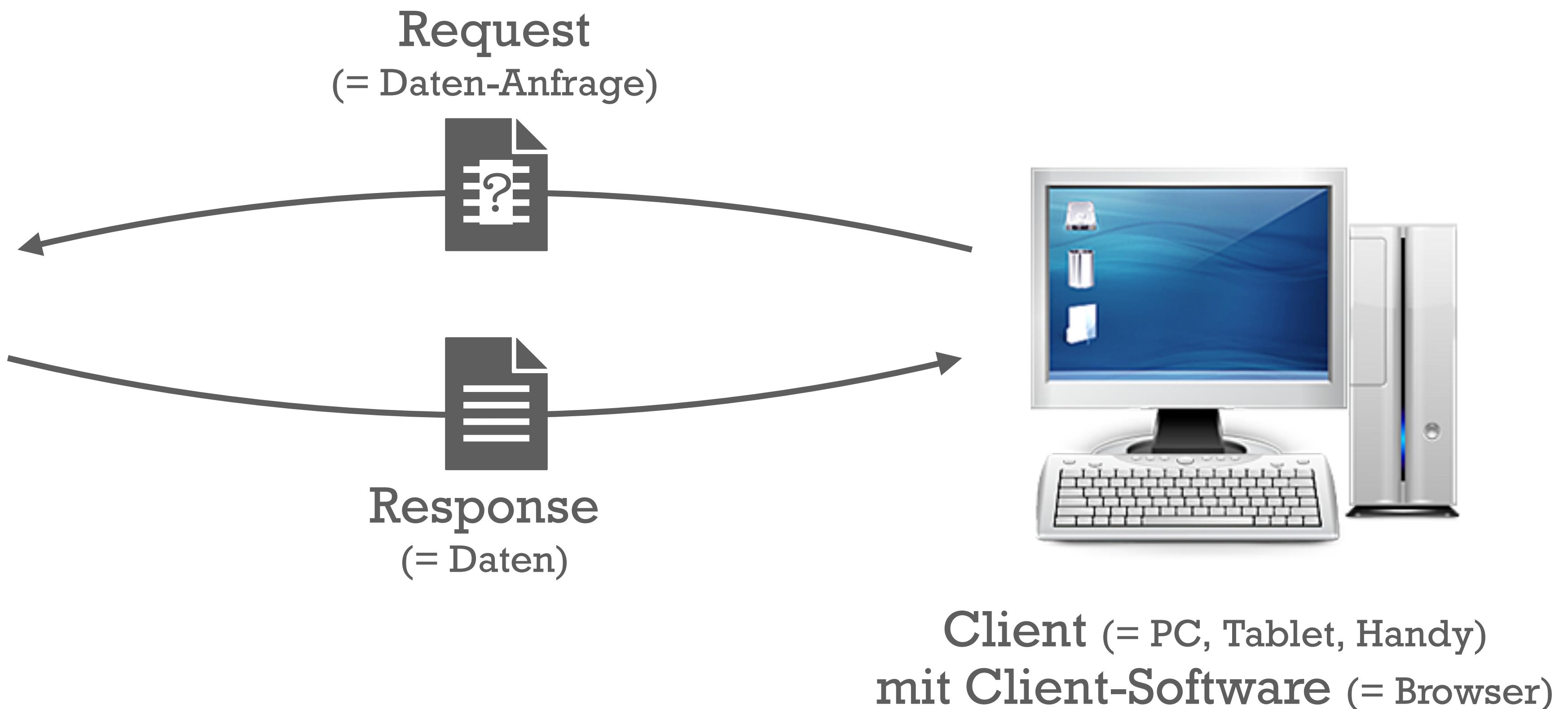
Tag 1 - Nachmittag :

*Online Daten erheben mit
Open Lab und OSF*

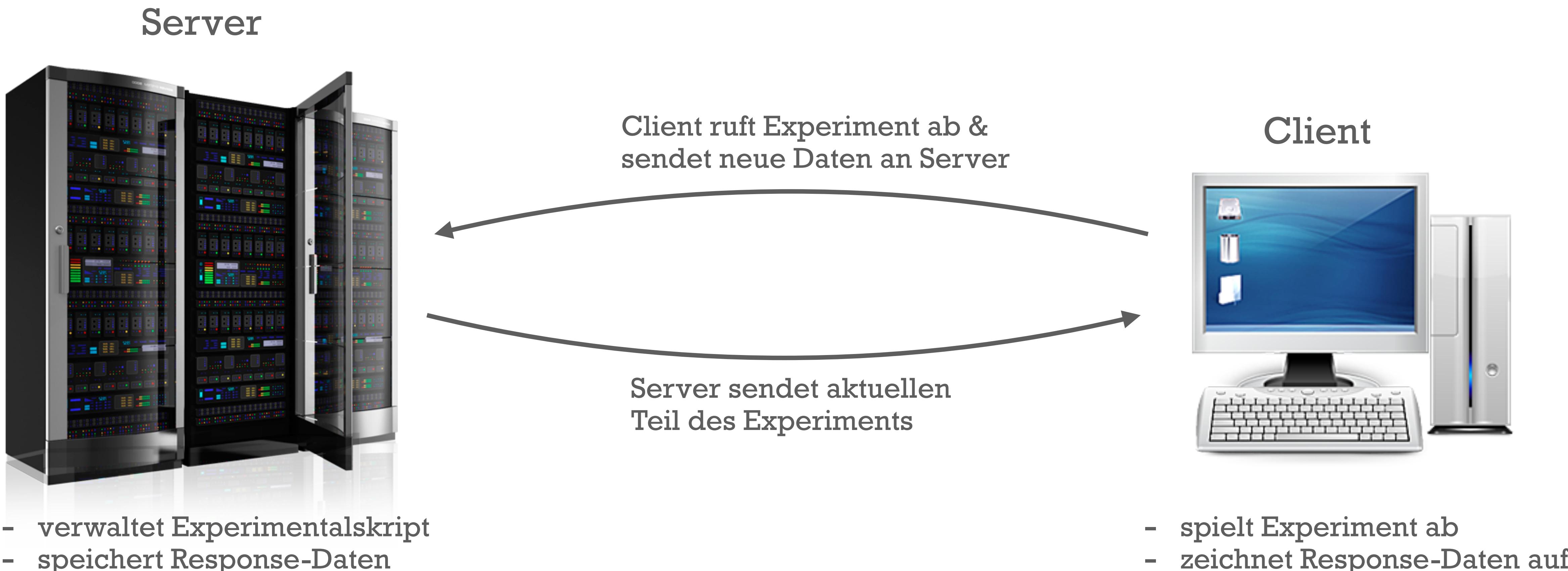
Wie funktionieren Onlinestudien?



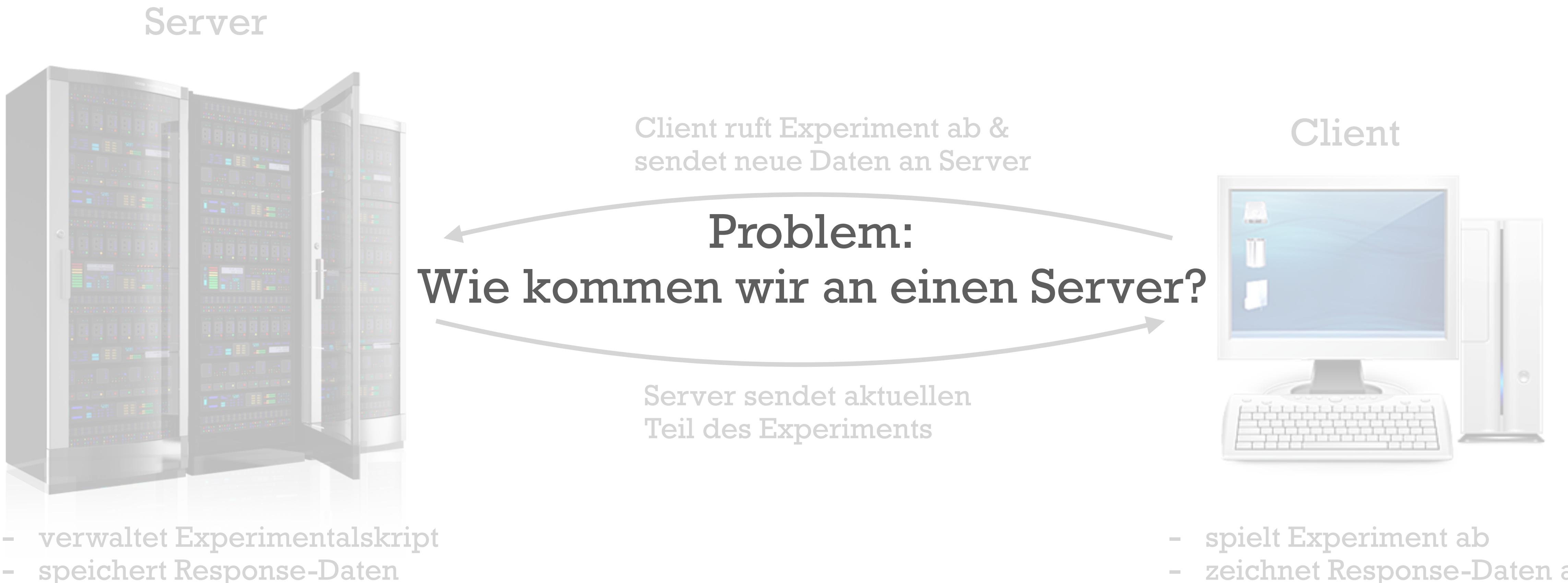
Server
mit Server-Software



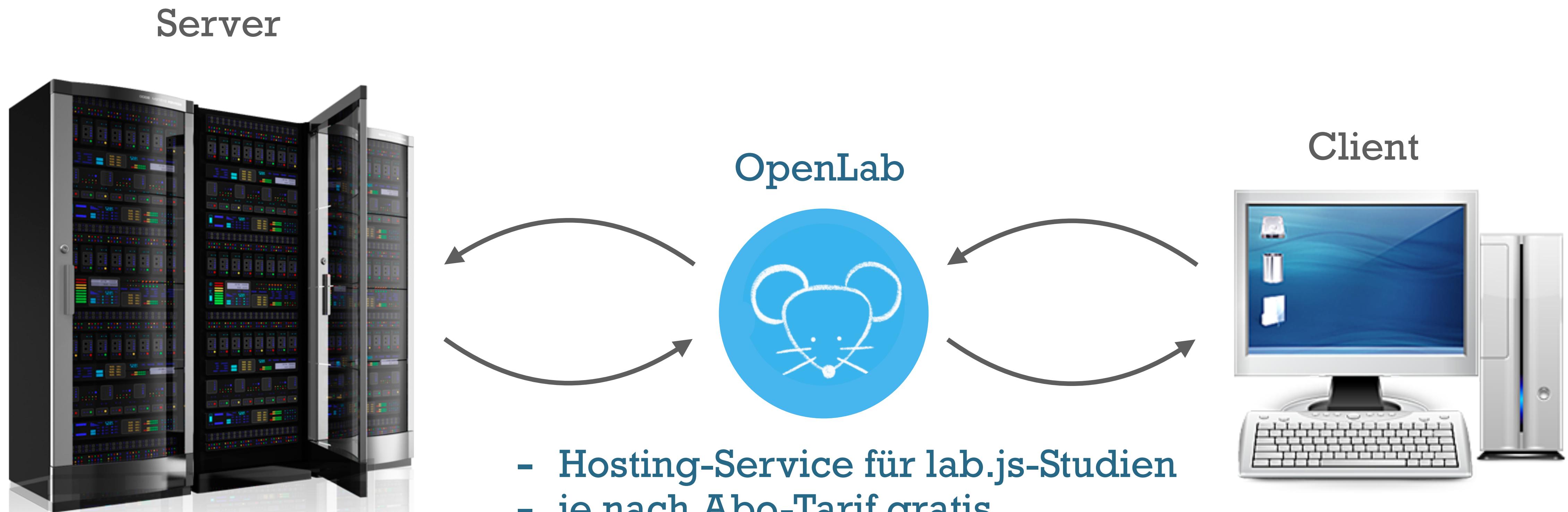
Wie funktionieren Onlinestudien?



Wie funktionieren Onlinestudien?



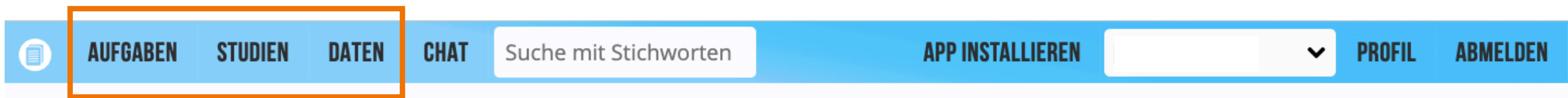
Studien hosten mit Open Lab



- Hosting-Service für lab.js-Studien
- je nach Abo-Tarif gratis oder ab 6€/Monat
- Server wird bei Bedarf gestellt

Aufgabe für Euch

- Ladet euch die lab.js-Studie aus dem Github-Repo herunter.
- Öffnet Open Lab und loggt euch ein: <https://open-lab.online/researcher/login>
- Erstellt eine neue Aufgabe, in der ihr das lab.js-Skript hochladet.
- Erstellt dann eine neue Studie, in der ihr die Aufgabe verwendet.
- Testet eure Studie und schaut, was dabei in den Daten passiert.

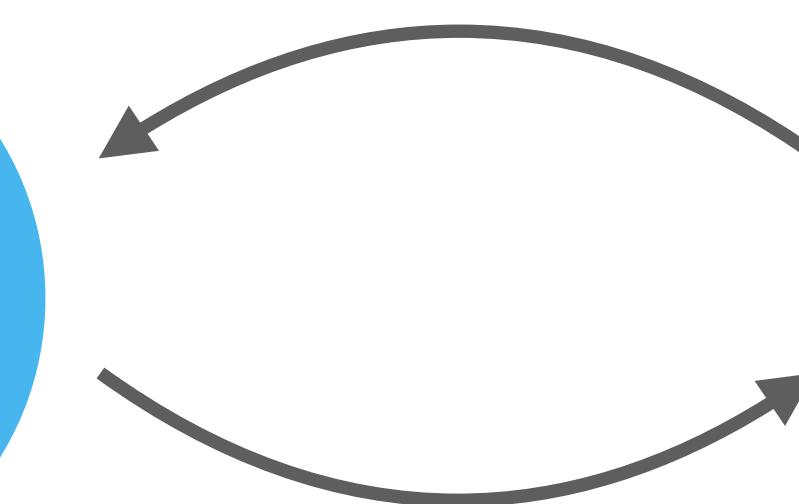
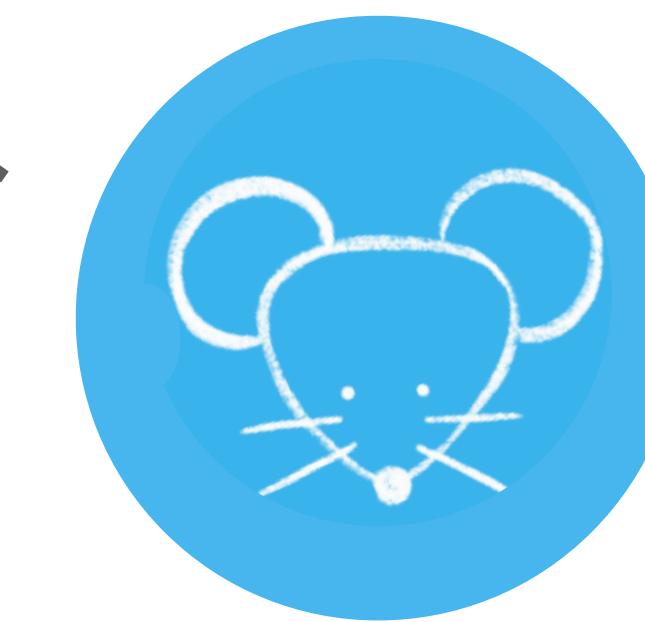


Wo landen unsere Daten?

gemieteter Server
von OpenLab



OpenLab



Client

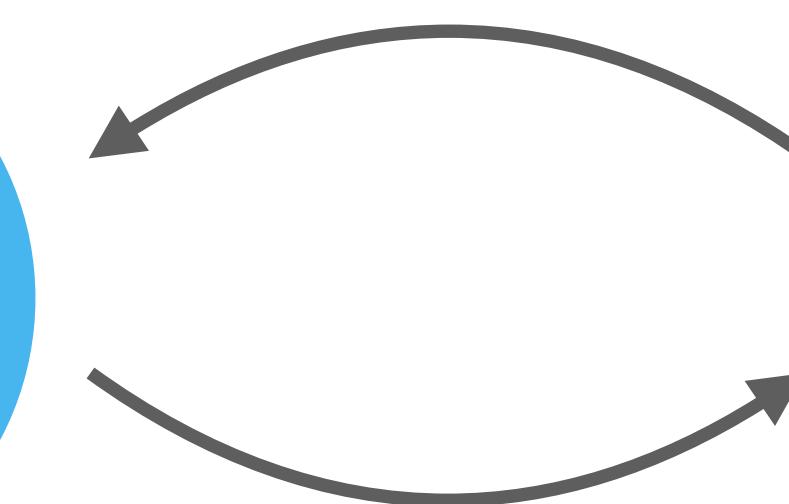
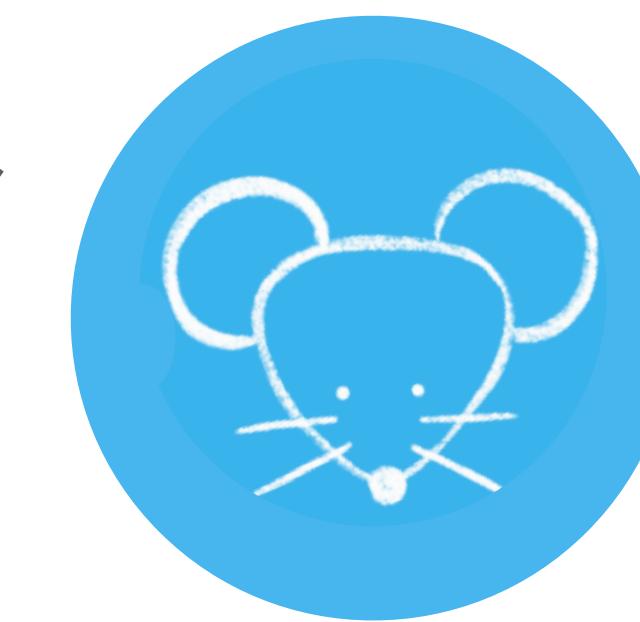


Wo landen unsere Daten?

Euer eigener Server
(z.B. ein Uni-Server)



OpenLab



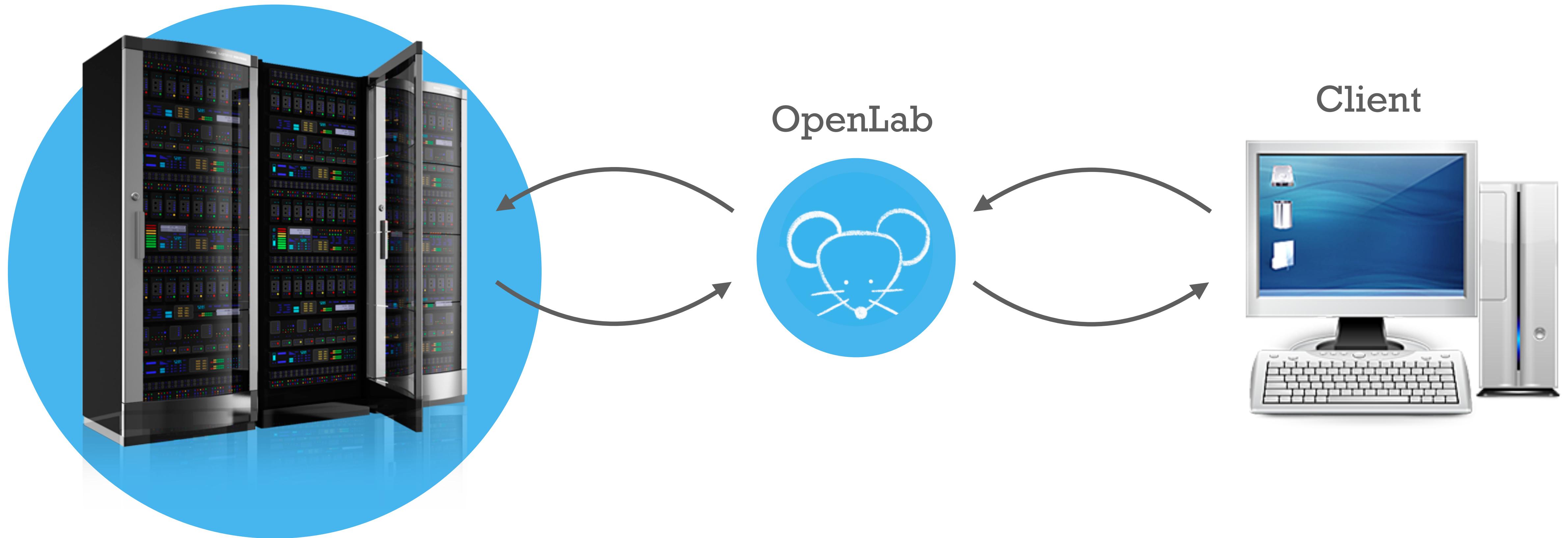
Client



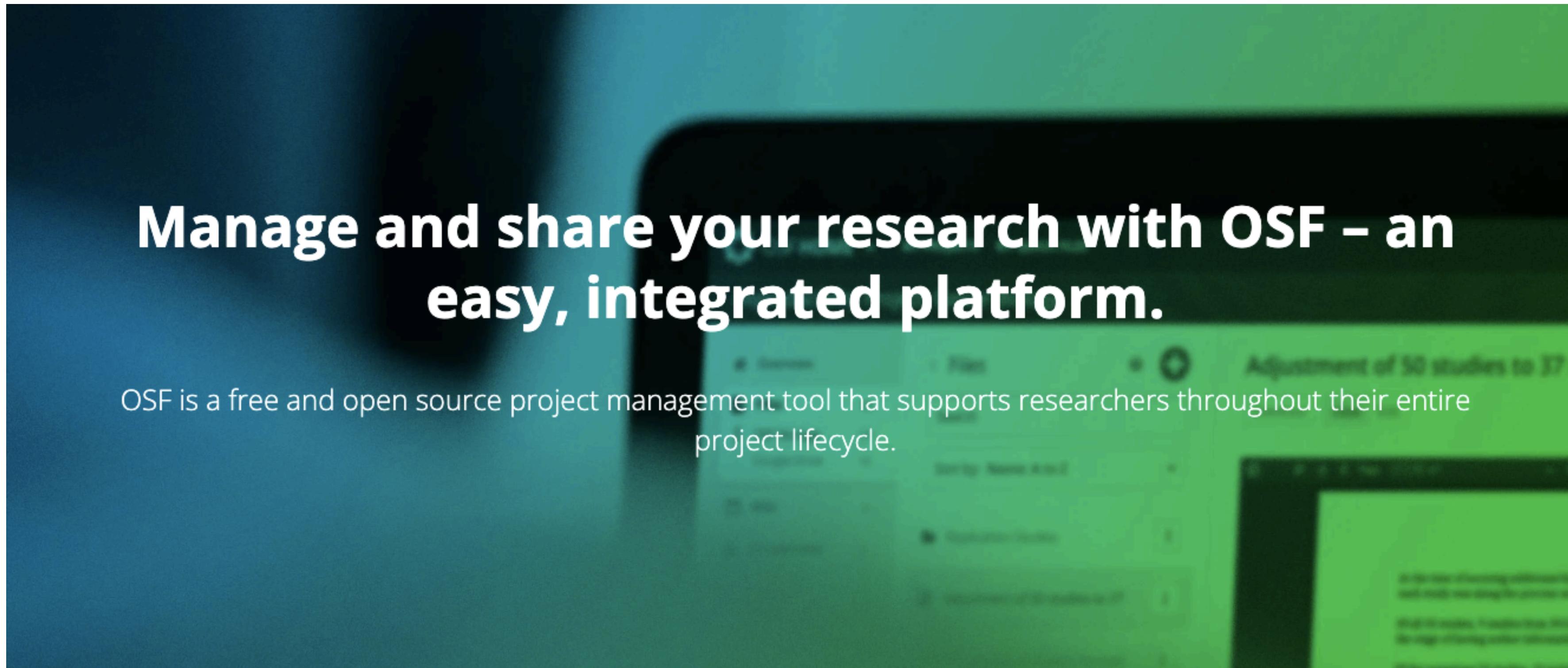
Wo landen unsere Daten?

Open Science Framework (OSF) Server

- > eigener Account notwendig
- > Verknüpfung über Personal Access Token



Daten speichern auf OSF



- Daten sicher auf OSF-Servern speichern (& ggf. mit anderen teilen)
- mit anderen an einem gemeinsamen Projekt arbeiten
- Preregistrations anlegen
- Preprints teilen
- Mit Github, Dropbox oder anderen Tools verknüpfen

Daten speichern auf OSF

Open Science Framework (OSF) Server

—> eigener Account notwendig

—> Verknüpfung über Personal Access Token

Daten speichern auf OSF

Open Science Framework (OSF) Server

—> eigener Account notwendig

—> Verknüpfung über Personal Access Token

- Personal Access Token = lange Abfolge von Zahlen und Buchstaben
- Website A bekommt PAT für Website B und darf damit auf Website B Daten hochladen oder abrufen

Daten speichern auf OSF

Open Science Framework (OSF) Server

—> eigener Account notwendig

—> Verknüpfung über Personal Access Token

- Personal Access Token = lange Abfolge von Zahlen und Buchstaben
- Website A bekommt PAT für Website B und darf damit auf Website B Daten hochladen oder abrufen
- Was Website A auf Website B tun darf, legt ihr im scope des Personal Access Tokens fest:
 - Schreiben = Daten hochladen
 - Lesen = Daten abrufen

Daten speichern auf OSF

Open Science Framework (OSF) Server

—> eigener Account notwendig

—> Verknüpfung über Personal Access Token

- Personal Access Token = lange Abfolge von Zahlen und Buchstaben
- Website A bekommt PAT für Website B und darf damit auf Website B Daten hochladen oder abrufen
- Was Website A auf Website B tun darf, legt ihr im scope des Personal Access Tokens fest:
 - Schreiben = Daten hochladen
 - Lesen = Daten abrufen
- Vorteil: Ihr teilt euer Passwort für Website B nicht mit Website A
- Vorteil: Ihr könnt das PAT jederzeit löschen und damit alle Zugriffsrechte für Website B löschen.

Daten speichern auf OSF

Open Science Framework (OSF) Server

—> eigener Account notwendig

—> Verknüpfung über Personal Access Token

The screenshot shows the OSF Settings page. On the left is a sidebar with the following options:

- My Profile
- OSF Support
- Settings** (highlighted with an orange border)
- Donate
- Help
- Log Out

The main content area has a title "Settings" and a sidebar with the following links:

- Profile information** (highlighted with a blue bar)
- Account settings
- Configure add-on accounts
- Notifications
- Developer apps
- Personal access tokens** (highlighted with an orange border)

The screenshot shows the "Personal access tokens" page with a "Create token" button at the top. Below it is a "Token name" field containing "Demo Token Name". Under "Scopes", there are several options with checkboxes:

- osf.full_read**: View all information associated with this account, including for private projects.
- osf.full_write**: View and edit all information associated with this account, including for private projects.
- osf.users.profile_read**: Read your profile data
- osf.users.email_read**: Read your primary email address.

A "Create token" button is located at the bottom right.

Daten speichern auf OSF

OSF / Settings / Personal Access Token

Settings

Personal access tokens

[← Back to list of tokens](#)

Token **Demo Token Name** created:



gh7hTTThl3zb782g2aswVBN3222Nbgbffffliuzzt555

This is the only time your token will be displayed.

This token will never expire. This token should never be shared with others. If it is accidentally revealed publicly, it should be deactivated immediately.

[Edit scopes](#)

Open Lab / Daten / OSF

OSF project title

my first study

OSF project description

Hi, this is a description of my study.

OSF project status



Private
Only for you



Public
For everybody

We need your OSF token to be able to write data in the new project. We will keep this token safe and only use it to store the data of the participants of the study my first study. You can delete the token later if you decide to remove the link between the Open Lab and the OSF project.

Read about tokens and create your own token with the scope osf.full_write

Paste your OSF token here



Daten speichern auf OSF

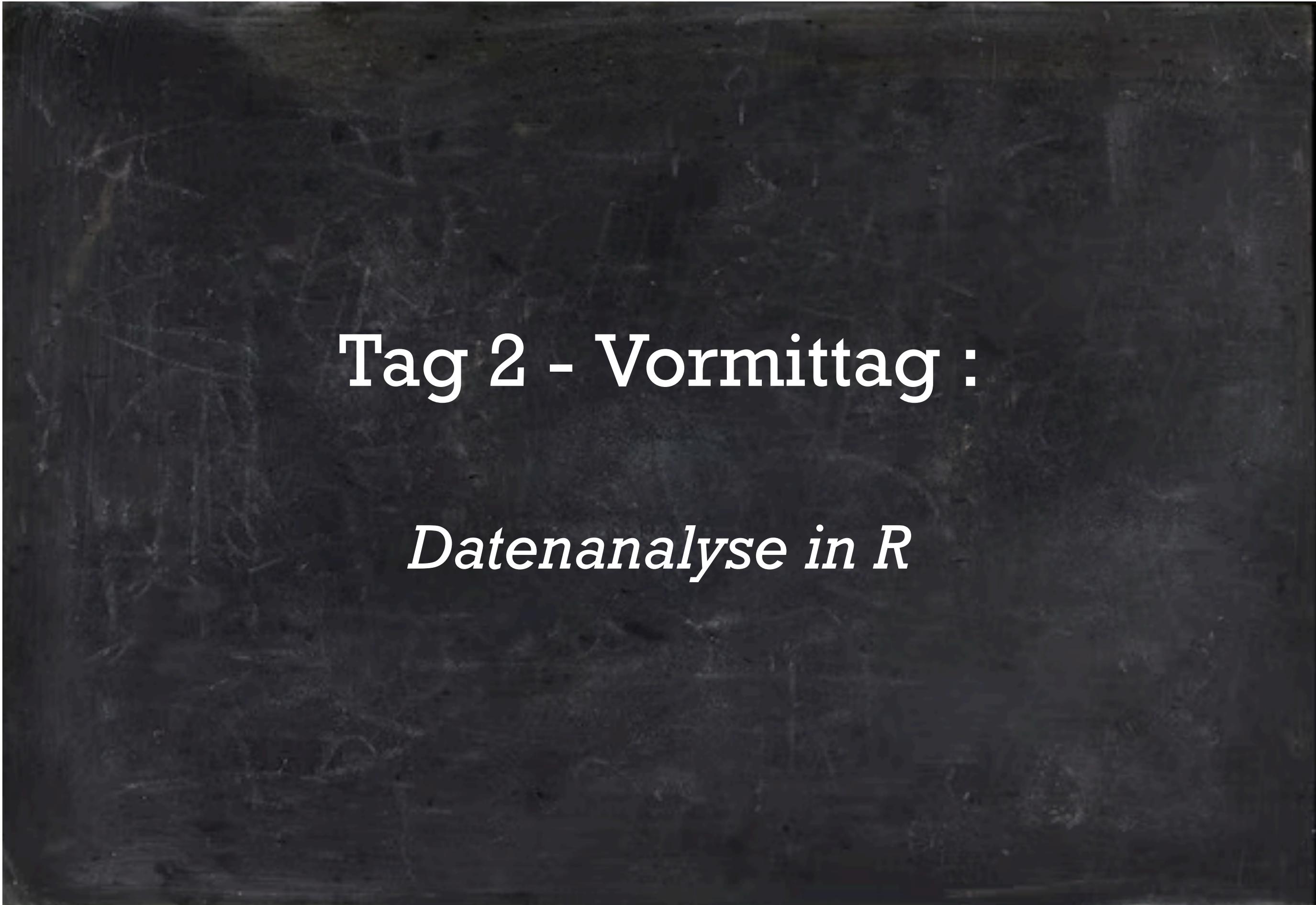
The screenshot shows the OpenLab interface. At the top, there are navigation tabs: AUFGABEN, STUDIEN, **DATEN**, CHAT, and a search bar. Below the tabs, there are links for APP INSTALLIEREN, EXNAT 1, PROFIL, and ABMELDEN. The main content area is titled "TEILNEHMER". It displays a table of participants with columns: №, Open Lab ID, Code, Bestätigungscode, Chat, Datum (Europe/Berlin), Aufgaben, Daten, Anfragen, Name, Sprache, Details, Meta, Params, and Benutzer löschen. The "Daten" column contains CSV and Excel download links. A red box highlights the "Alle Daten herunterladen" (All data download) link at the top right of the table. Another red box highlights the Open Lab ID "CBQ3W1iD5W" for participant 3.

Nº	Open Lab ID	Code	Bestätigungscode	Chat	Datum (Europe/Berlin)	Aufgaben	Daten	Anfragen	Name	Sprache	Details	Meta	Params	Benutzer löschen
1	BH4uxRi7HH	hcpibp6kldwz3ybm	911253	Öffnen	02-09-23, 11:45	1	CSV Excel	0		Ger	Öffnen	CSV Excel		(X)
2	rXQymmm0y-7	hcpibp6kldwzz9yb	273285	Öffnen	02-09-23, 12:09	1	CSV Excel	0		Ger	Öffnen	CSV Excel		(X)
3	CBQ3W1iD5W	hcpibp6kldx1air1	563791	Öffnen	02-09-23, 12:46	1	CSV Excel	0		Ger	Öffnen	CSV Excel		(X)

The screenshot shows the OSF project page for "Executive Networks in Natural Language Processing – EXNAT 1 (online study)". The page includes links for Project Navigation, OSFHOME, and a dropdown menu. It shows project details: 66.0MB, Private, Make Public, and a three-dot menu. Contributors are listed as Merle Marie Schuckart and Sandra Martin. The project was created on 2023-01-11 and last updated on 2023-02-10. Category is Project. Description states it's a repository for all data collected in the EXNAT-1 online study. License is "Add a license". The page features sections for Wiki and Files, and a file list table.

Name	Modified
Executive Networks in Natural Language Processing – EXNAT 1 (o...	
- OSF Storage (Germany - Frankfurt)	
+ experiment	
- pilot_datasets	
0ZqL9PPj3K-c73fa6f7-4552-44e5-839e-5e9cb5165547.csv	2023-02-09 01:54 PM
756XCijawd-f2b205a3-0a91-46ae-b81a-784a7f00423e.csv	2023-02-09 03:59 PM
7xkYvyLI3y-3020387a-a490-4885-8c87-1e3cb97be28d.csv	2023-02-09 06:18 PM
9kiyDUTjX5-38329c81-7f0c-45e7-b7a7-907b953f09cc.csv	2023-02-09 06:14 PM
CBQ3W1iD5W-a18a9da0-319b-4d60-ade9-5946312784e...	2023-02-09 01:52 PM
D9_zX83Ofv-ed37a3bc-59e4-4204-882b-e772c23b6809....	2023-02-09 04:19 PM
DGmsp9cBWO-63b958d5-c4f2-4818-87a2-001d262f9e2...	2023-02-09 05:44 PM

- Open Lab speichert vollständige UND inkrementelle (= unvollständige) Datensätze
- Daten abrufbar als CSV mit den Daten aller VPn oder als Einzel-Datensätze für jede VP
- komplette Datensätze werden auf OSF gespeichert, inkrementelle nicht



Tag 2 - Vormittag :

Datenanalyse in R

Ablauf

Das machen wir heute:

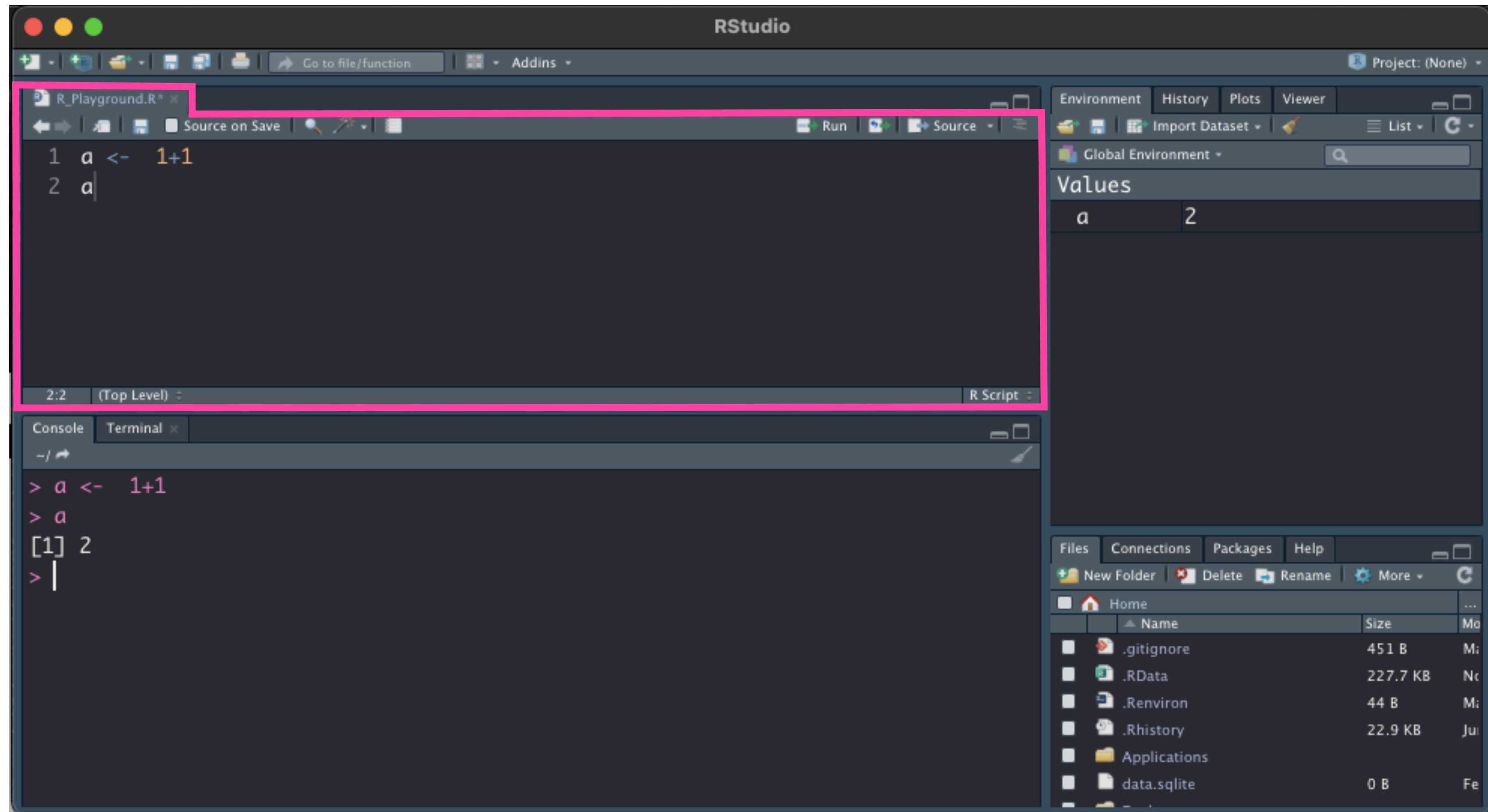
1. Basics in R + Aufgaben
 2. Wir bauen ein R-Skript
- Mittagspause*
3. Was ist Versionierung?
 4. Übungen zur Versionierung

Das braucht ihr heute:

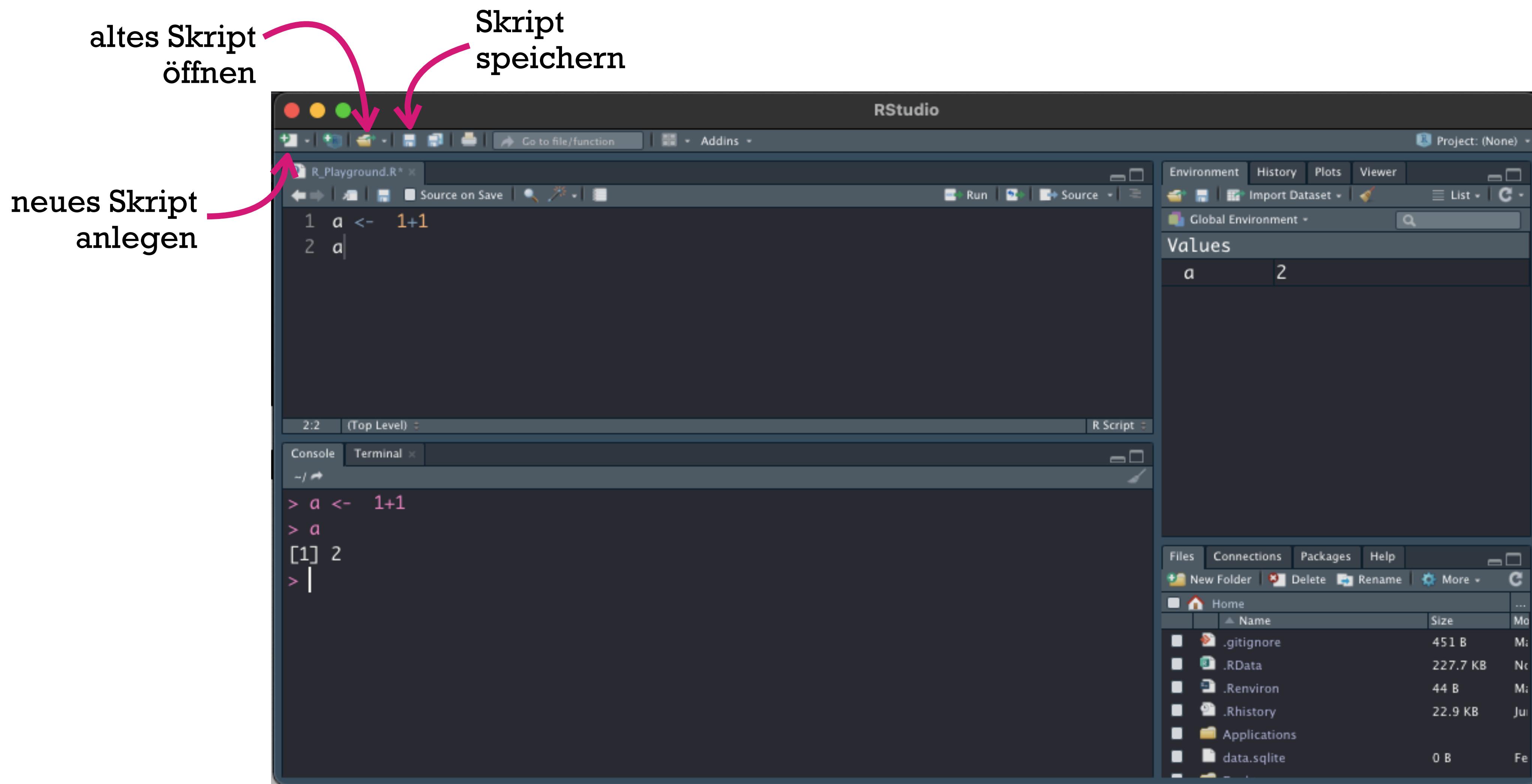
- R und RStudio auf eurem Laptop
- einen Github-Account
- Github-Desktop auf eurem Laptop
- 12 Dateien mit .csv-Endung
- zwei .R-Dateien:
`R_Tutorial_live.R`
`z_sqrt_POMS_func.R`

R-Studio

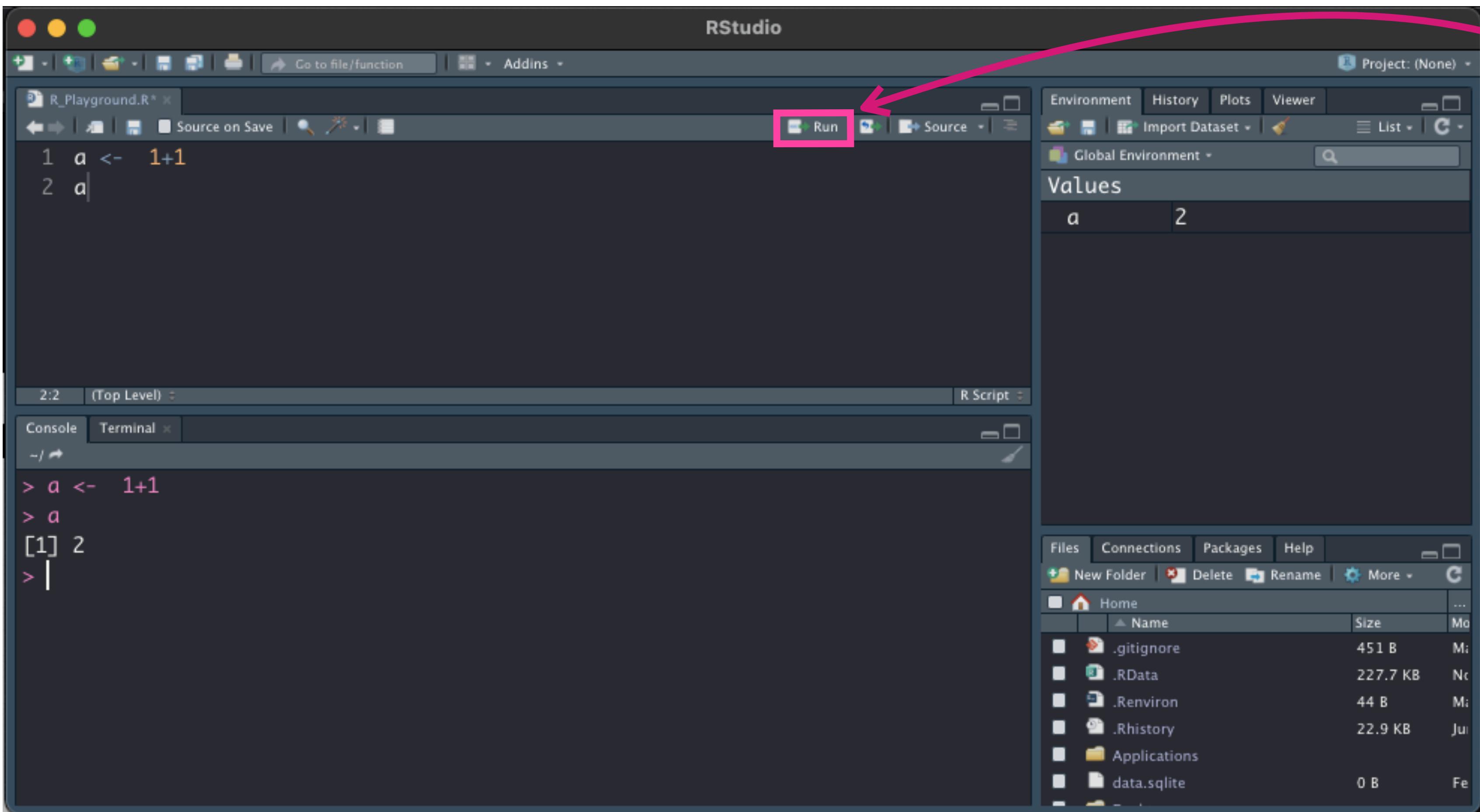
Skript:
Das Dokument, in dem
euer fertiger Code steht



R-Studio



R-Studio

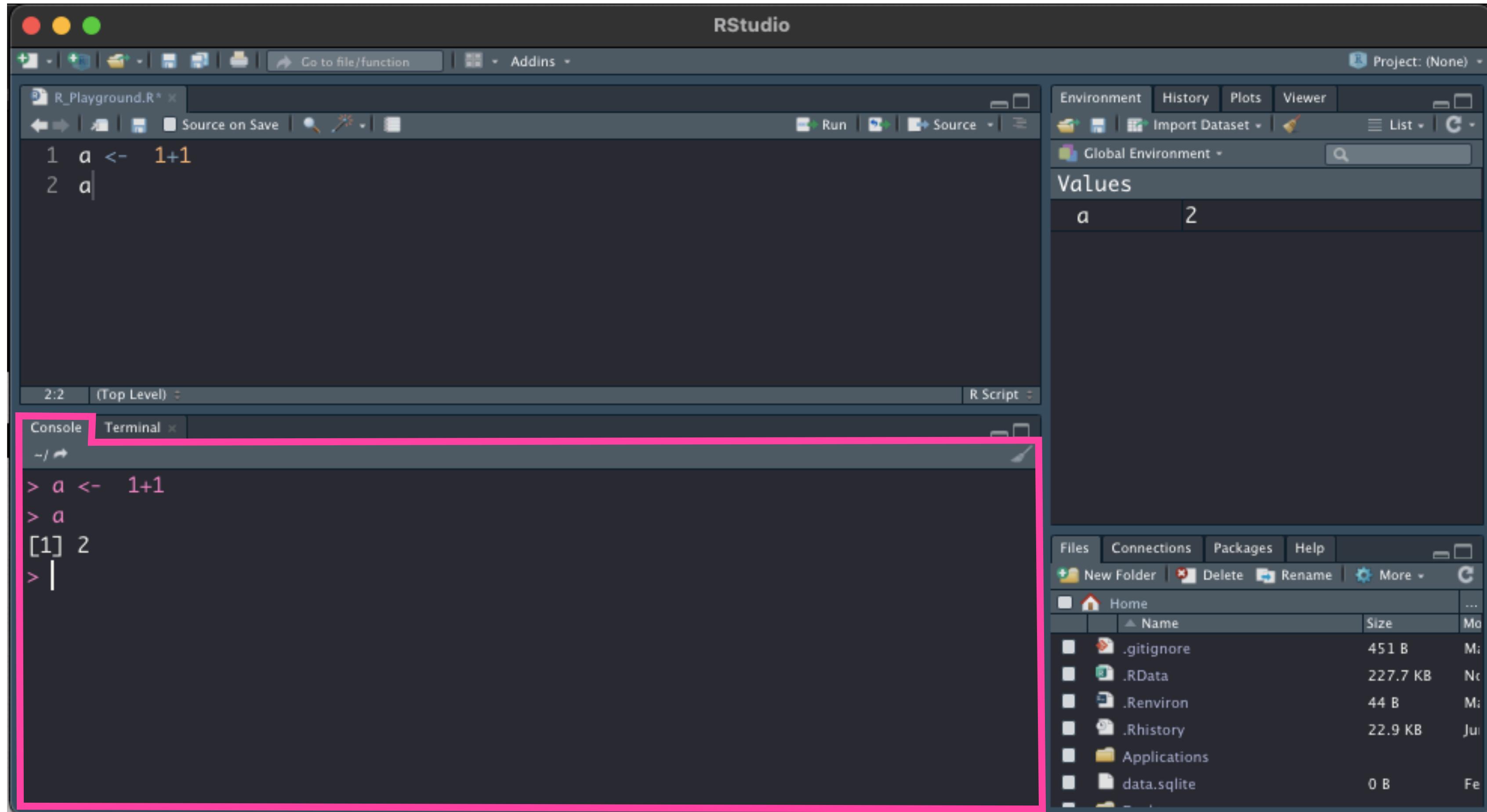


Run:
führt markierte Zeile(n) im Skript aus und gibt Ergebnis in der Konsole aus

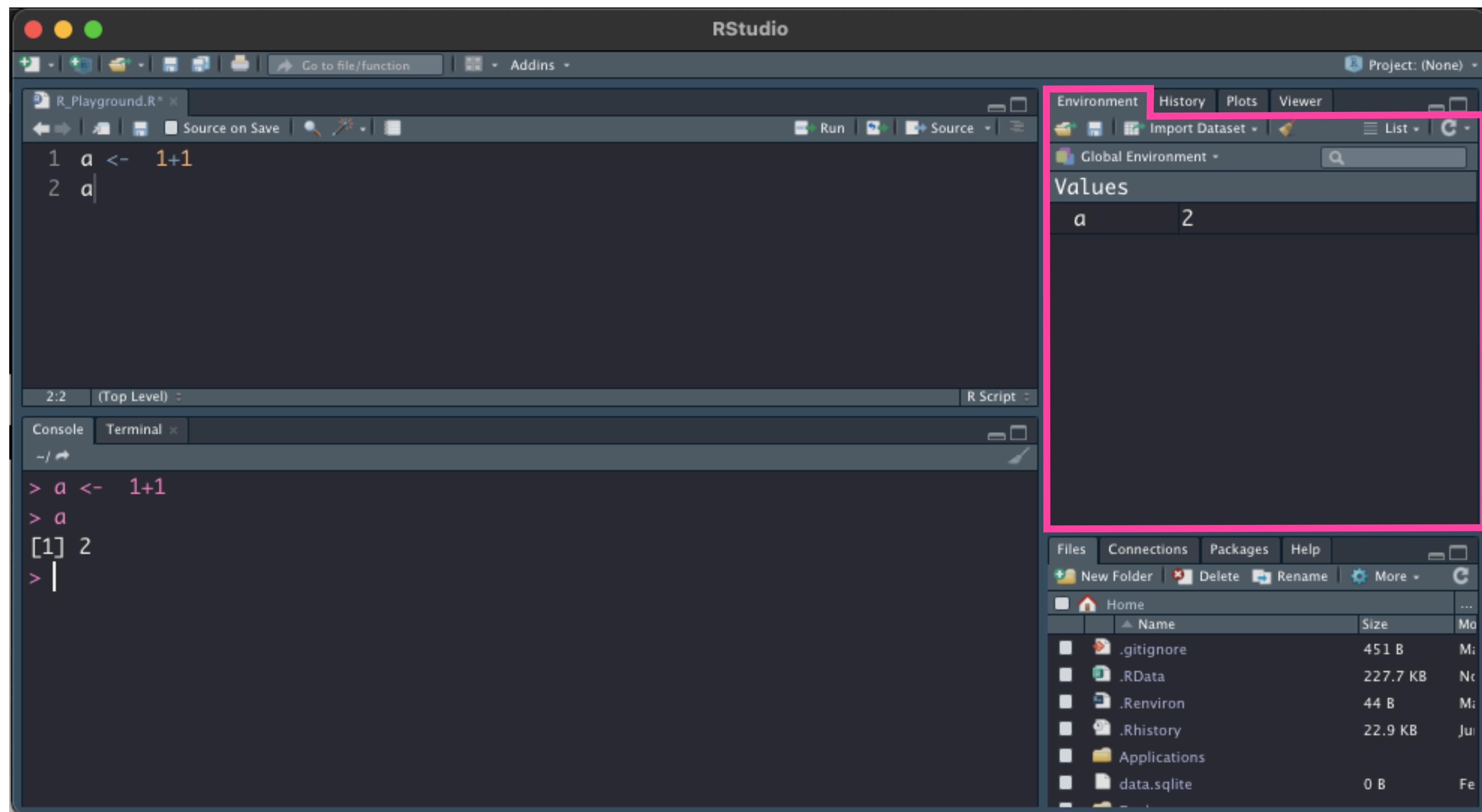
Alternative:
alt + Enter

R-Studio

Konsole:
gibt Ergebnisse und
Fehlermeldungen aus,
man kann auch darin
direkt schreiben

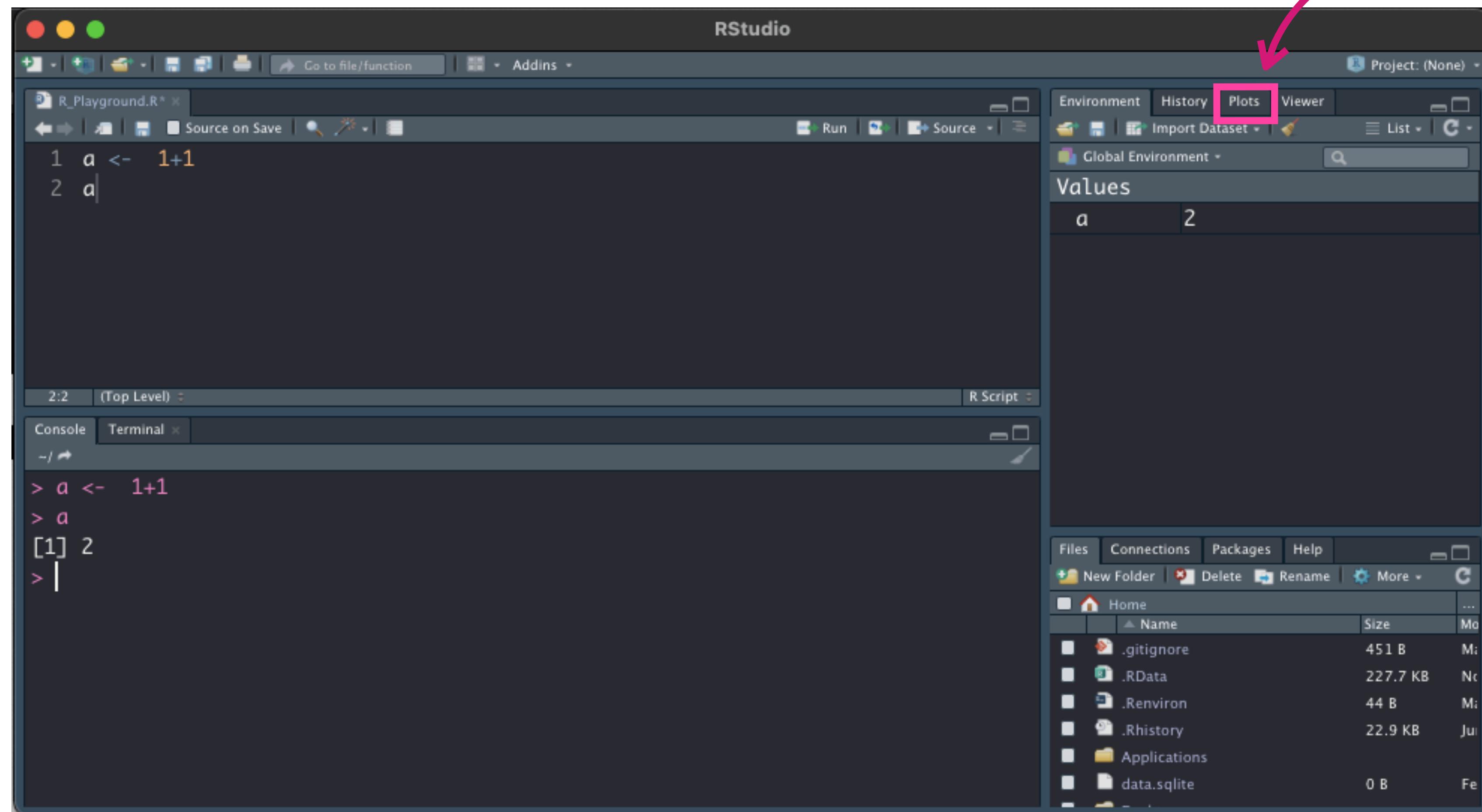


R-Studio



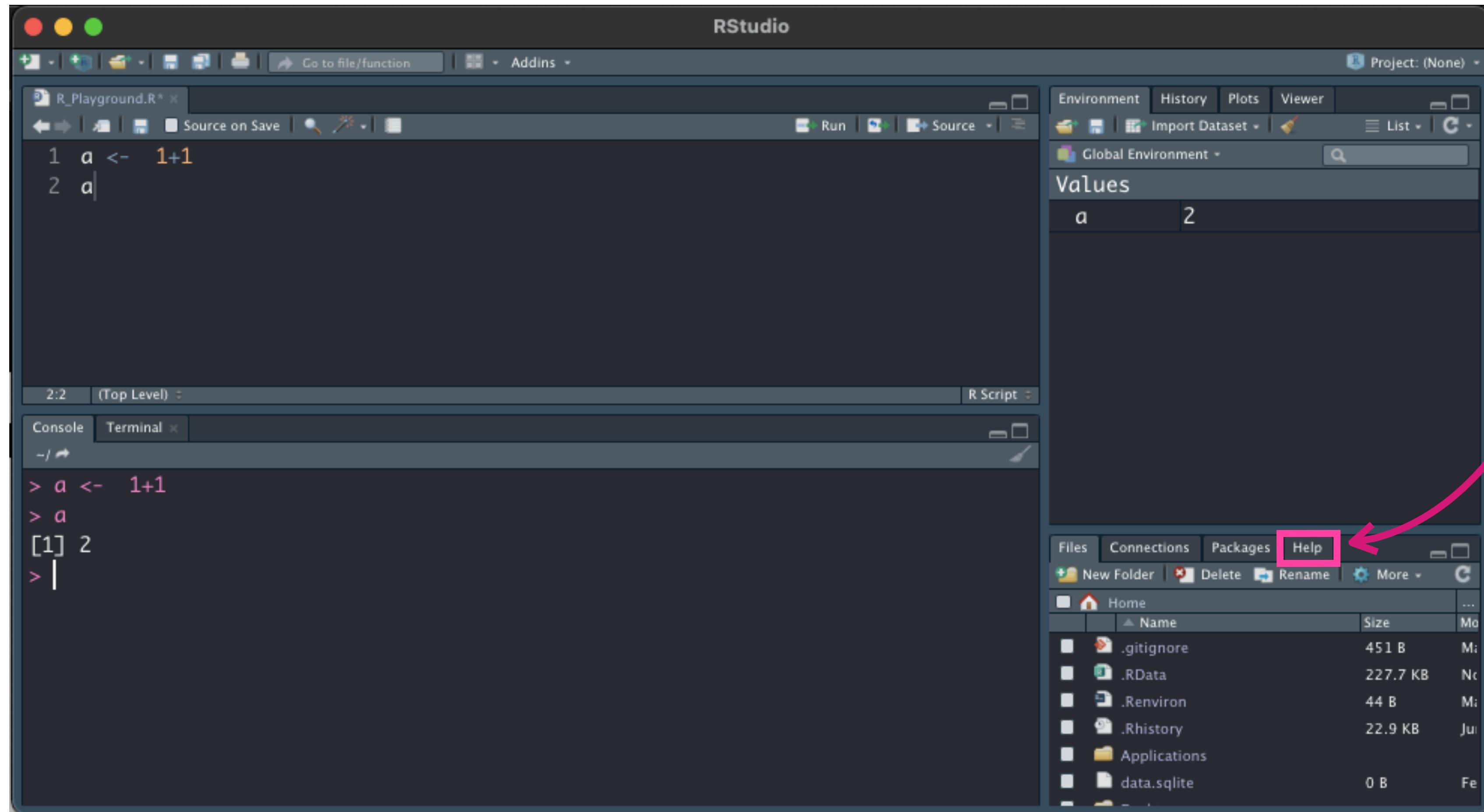
Environment:
Übersicht über
alle Variablen im
Speicher

R-Studio



Plots:
Zeigt euch eure
Abbildungen an

R-Studio



Help:
z.B. Package
Dokumentation
anschauen

Basics in R

Arten von Daten:

Character Strings (Verkettungen von Zeichen): "abc", "Das ist ein Satz", "ab39gHb", "2783", ":-)"
Integers / Numerics (diskrete Werte = Zahlen): 42, 82736.6666
Booleans / Logicals (boolsche/binäre Ausdrücke): TRUE, FALSE, T, F
Fehlende Werte: NA
Factors: kategoriale Werte, die eine bestimmte Ordnung haben; Level werden als Character gespeichert

Datentypen verändern (= typecasten):

Beispiel: Numeric in Character umwandeln

```
as.character(1234)
> [1] "1234"
as.numeric("1234")
> [1] 1234
```

Achtung! Beim Typecasten von factors zu numerics muss man aufpassen, weil sonst ggf. die Werte verändert werden.

Nutzt am besten einen der folgenden Befehle:
as.numeric(levels(f))[f]
as.numeric(as.character(f))

Basics in R

Daten organisieren:

Einzelwerte als Objekte anlegen:

a <- 17

Wenn man dann a ausführt, wird der Wert 17 ausgegeben:

> [1] 17

Aufgaben für Euch

Legt eine Variable namens *meine_erste_Variable* an, die den Wert 2 hat.

Hinweis: Ihr könnt die Zeile mit dem Button „Run“ ausführen.

Aufgaben für Euch

Legt eine Variable namens *meine_erste_Variable* an, die den Wert 2 hat.

Hinweis: Ihr könnt die Zeile mit dem Button „Run“ ausführen.

Lösung:

```
meine_erste_Variable <- 2
```

Aufgaben für Euch

Legt eine Variable namens *meine_erste_Variable* an, die den Wert 2 hat.

Hinweis: Ihr könnt die Zeile mit dem Button „Run“ ausführen.

Lösung:

```
meine_erste_Variable <- 2
```

Multipliziert die 2 in eurer Variable mit 5. Eure Variable sollte nun den Wert 10 haben.

Typecastet eure Variable zu einem *character*.

Hinweise:

Schreibt den neuen Code unter die Zeile aus Aufgabe 1.

Der Befehl `as.character()` hilft euch beim Typecasten.

Aufgaben für Euch

Legt eine Variable namens *meine_erste_Variable* an, die den Wert 2 hat.

Hinweis: Ihr könnt die Zeile mit dem Button „Run“ ausführen.

Lösung:

```
meine_erste_Variable <- 2
```

Multipliziert die 2 in eurer Variable mit 5. Eure Variable sollte nun den Wert 10 haben.

Typecastet eure Variable zu einem *character*.

Hinweise:

Schreibt den neuen Code unter die Zeile aus Aufgabe 1.

Der Befehl *as.character()* hilft euch beim Typecasten.

mögliche Lösung 1:

```
meine_erste_Variable <- meine_erste_Variable * 5  
meine_erste_Variable <- as.character(meine_erste_Variable)
```

mögliche Lösung 2:

```
meine_erste_Variable <- as.character(meine_erste_Variable * 5)
```

Aufgaben für Euch

Legt eine weitere Variable namens *meine_zweite_Variable* an und weist ihr den Wert 100 zu.
Schreibt ein # vor euren Variablennamen.
Was passiert mit dem Code? Was passiert, wenn ihr die Zeile ausführt?

Aufgaben für Euch

Legt eine weitere Variable namens *meine_zweite_Variable* an und weist ihr den Wert 100 zu.
Schreibt ein # vor euren Variablenamen.
Was passiert mit dem Code? Was passiert, wenn ihr die Zeile ausführt?

```
# meine_zweite_Variable <- 10
```

Lösung: Zeilen mit einem # davor werden nicht ausgeführt.

Basics in R

Daten organisieren:

Einzelwerte als Objekte anlegen:

```
a <- 17
```

Wenn man dann a ausführt, wird der Wert 17 ausgegeben:

```
> [1] 17
```

Vectoren anlegen:

```
a <- c(1,2,3)
```

Wenn man dann a ausführt, wird der Vector ausgegeben:

```
> [1] 1 2 3
```

Vectoren aneinander hängen:

```
A <- c(1,2,3)
```

```
B <- c(4,5,6,7)
```

```
C <- c(a, b)
```

```
> [1] 1 2 3 4 5 6 7
```

Aufgaben für Euch

Legt einen Vector namens `vec` an und weist ihm die folgenden Werte zu: 1, 3, 4, 6, 3, 2000

Aufgaben für Euch

Legt einen Vector namens `vec` an und weist ihm die folgenden Werte zu: 1, 3, 4, 6, 3, 2000

Lösung:

```
vec <- c(1, 3, 4, 6, 3, 2000)
```

Aufgaben für Euch

Legt einen Vector namens `vec` an und weist ihm die folgenden Werte zu: 1, 3, 4, 6, 3, 2000

Lösung:

```
vec <- c(1, 3, 4, 6, 3, 2000)
```

Legt einen zweiten Vector an und nennt ihn `vec2`.

Weist ihm die folgenden Werte zu: 10, 12, 90, 5, 13, 32

Hängt `vec2` hinten an euren ersten Vector `vec` an. Der neue Vector soll trotzdem noch `vec` heißen. Was passiert nach dem Ausführen mit `vec2`?

Aufgaben für Euch

Legt einen Vector namens `vec` an und weist ihm die folgenden Werte zu: 1, 3, 4, 6, 3, 2000

Lösung:

```
vec <- c(1, 3, 4, 6, 3, 2000)
```

Legt einen zweiten Vector an und nennt ihn `vec2`.

Weist ihm die folgenden Werte zu: 10, 12, 90, 5, 13, 32

Hängt `vec2` hinten an euren ersten Vector `vec` an. Der neue Vector soll trotzdem noch `vec` heißen. Was passiert nach dem Ausführen mit `vec2`?

Lösung:

```
vec2 <- c(10, 12, 90, 5, 13, 32)
```

```
vec <- c(vec, vec2)
```

```
# vec2 bleibt bestehen
```

Aufgaben für Euch

Legt einen Vector namens `vec` an und weist ihm die folgenden Werte zu: 1, 3, 4, 6, 3, 2000

Lösung:

```
vec <- c(1, 3, 4, 6, 3, 2000)
```

Legt einen zweiten Vector an und nennt ihn `vec2`.

Weist ihm die folgenden Werte zu: 10, 12, 90, 5, 13, 32

Hängt `vec2` hinten an euren ersten Vector `vec` an. Der neue Vector soll trotzdem noch `vec` heißen. Was passiert nach dem Ausführen mit `vec2`?

Lösung:

```
vec2 <- c(10, 12, 90, 5, 13, 32)
```

```
vec <- c(vec, vec2)
```

```
# vec2 bleibt bestehen
```

Legt eine neue Variable namens `vec3` an und addiert darin `vec2` und die Zahl 5.

Was passiert mit den Zahlen in `vec2`?

Aufgaben für Euch

Legt einen Vector namens `vec` an und weist ihm die folgenden Werte zu: 1, 3, 4, 6, 3, 2000

Lösung:

```
vec <- c(1, 3, 4, 6, 3, 2000)
```

Legt einen zweiten Vector an und nennt ihn `vec2`.

Weist ihm die folgenden Werte zu: 10, 12, 90, 5, 13, 32

Hängt `vec2` hinten an euren ersten Vector `vec` an. Der neue Vector soll trotzdem noch `vec` heißen. Was passiert nach dem Ausführen mit `vec2`?

Lösung:

```
vec2 <- c(10, 12, 90, 5, 13, 32)
```

```
vec <- c(vec, vec2)
```

```
# vec2 bleibt bestehen
```

Legt eine neue Variable namens `vec3` an und addiert darin `vec2` und die Zahl 5.

Was passiert mit den Zahlen in `vec2`?

Lösung:

```
vec3 <- vec2 + 5
```

```
> 15 17 95 10 18 37
```

Basics in R

Daten organisieren:

Einzelwerte als Objekte anlegen:

```
a <- 17
```

Wenn man dann a ausführt, wird der Wert 17 ausgegeben:

```
> [1] 17
```

Vector anlegen:

```
a <- c(1,2,3)
```

Wenn man dann a ausführt, wird der Vector ausgegeben:

```
> [1] 1 2 3
```

Matrix erstellen:

mehrere Vectoren als Zeilen aneinander binden:

```
z <- rbind(vector1, vector2)
```

```
View(z)
```



	vector1	1	2	3	4
vector2	10	16	8	9	

Matrix erstellen:

mehrere Vectoren als Spalten aneinander binden:

```
z <- cbind(vector1, vector2)
```

```
View(z)
```



	vector1	vector2
1	1	10
2	2	16
3	3	8

Matrix in einen Dataframe umwandeln:

```
my_df <- as.data.frame(z)
```

Aufgaben für Euch

Legt 2 Vectoren an:

- 1 Vector namens *name* mit den folgenden Werten: „Bertie“, „Fiete“, „Carlos“, „Agnes“
- 1 Vector namens *cuteness* mit den folgenden Werten: 100, 2, 100, 37

Bindet die Vectoren als Spalten (= columns) aneinander und nennt eure neue Matrix *cat_ranking*.

Aufgaben für Euch

Legt 2 Vectoren an:

- 1 Vector namens *name* mit den folgenden Werten: “Bertie“, “Fiete“, “Carlos“, “Agnes“
- 1 Vector namens *cuteness* mit den folgenden Werten: 100, 2, 100, 37

Bindet die Vectoren als Spalten (= columns) aneinander und nennt eure neue Matrix *cat_ranking*.

Lösung:

```
name      <- c("Bertie", "Fiete", "Carlos", "Agnes")
cuteness <- c(100, 2, 100, 37)
cat_ranking <- cbind(name, cuteness)
```

Aufgaben für Euch

Legt 2 Vectoren an:

- 1 Vector namens *name* mit den folgenden Werten: “Bertie“, “Fiete“, “Carlos“, “Agnes“
- 1 Vector namens *cuteness* mit den folgenden Werten: 100, 2, 100, 37

Bindet die Vectoren als Spalten (= columns) aneinander und nennt eure neue Matrix *cat_ranking*.

Lösung:

```
name      <- c("Bertie", "Fiete", "Carlos", "Agnes")
cuteness <- c(100, 2, 100, 37)
cat_ranking <- cbind(name, cuteness)
```

Wandelt nun die Matrix *cat_ranking* in einen *data.frame* um.

Hinweis: Der Befehl, den ihr braucht, lautet *as.data.frame()*

Aufgaben für Euch

Legt 2 Vectoren an:

- 1 Vector namens *name* mit den folgenden Werten: "Bertie", "Fiete", "Carlos", "Agnes"
- 1 Vector namens *cuteness* mit den folgenden Werten: 100, 2, 100, 37

Bindet die Vectoren als Spalten (= columns) aneinander und nennt eure neue Matrix *cat_ranking*.

Lösung:

```
name      <- c("Bertie", "Fiete", "Carlos", "Agnes")
cuteness <- c(100, 2, 100, 37)
cat_ranking <- cbind(name, cuteness)
```

Wandelt nun die Matrix *cat_ranking* in einen data.frame um.

Hinweis: Der Befehl, den ihr braucht, lautet *as.data.frame(matrix)*

Lösung:

```
cat_ranking <- as.data.frame(cat_ranking)
```

Aufgaben für Euch

Typecastet die Spalte cuteness in der Matrix `cat_ranking` zu numeric.

Hinweis 1: Mit dem Operator `$` könnt ihr Spalten auswählen (Beispiel: `some_matrix$colname` gibt euch die Spalte `colname` wie einen Vector aus).

Hinweis 2: der Befehl `as.numeric(vector)` hilft euch.

Aufgaben für Euch

Typecastet die Spalte cuteness in der Matrix cat_ranking zu numeric.

Hinweis 1: Mit dem Operator \$ könnt ihr Spalten auswählen (Beispiel: *some_matrix\$colname* gibt euch die Spalte *colname* wie einen Vector aus).

Hinweis 2: der Befehl `as.numeric(vector)` hilft euch.

Lösung:

```
cat_ranking$cuteness <- as.numeric(cat_ranking$cuteness)
```

Basics in R

Rechenoperationen:

Addieren: $a + b$

Subtrahieren: $a - b$

Multiplizieren: $a * b$

Dividieren: a / b

Potenz: a^b

Wurzel ziehen: `sqrt(a)`

Mittelwert: `mean(vector)`

Median: `median(vector)`

Standardabweichung: `sd(vector)`

Minimum und Maximum in einem Vektor finden: `range(vector)`

Minimum oder maximum in einem Vektor finden: `min(vector)` bzw. `max(vector)`

Anzahl von Elementen in einem Vektor zählen: `length(vector)`

Verteilung des Vectors zusammenfassen: `summary(vector)`

Aufgaben für Euch

Wir möchten wissen, wie niedlich die Katzen in meinem Umfeld im Schnitt sind.

Berechnet den Mittelwert und den Median der Werte in der Spalte *cuteness*.

Aufgaben für Euch

Wir möchten wissen, wie niedlich die Katzen in meinem Umfeld im Schnitt sind.

Berechnet den Mittelwert und den Median der Werte in der Spalte *cuteness*.

Lösung:

```
mean(cat_ranking$cuteness)
```

```
> 59.75
```

```
median(cat_ranking$cuteness)
```

```
> 68.5
```

Aufgaben für Euch

Wir möchten wissen, wie niedlich die Katzen in meinem Umfeld im Schnitt sind.

Berechnet den Mittelwert und den Median der Werte in der Spalte *cuteness*.

Lösung:

```
mean(cat_ranking$cuteness)
```

```
> 59.75
```

```
median(cat_ranking$cuteness)
```

```
> 68.5
```

Wir möchten außerdem wissen, wie niedlich die übelste Katze in meinem Umfeld ist.

Lasst euch den niedrigsten Wert in der Spalte *cuteness* ausgeben.

Aufgaben für Euch

Wir möchten wissen, wie niedlich die Katzen in meinem Umfeld im Schnitt sind.

Berechnet den Mittelwert und den Median der Werte in der Spalte *cuteness*.

Lösung:

```
mean(cat_ranking$cuteness)
```

```
> 59.75
```

```
median(cat_ranking$cuteness)
```

```
> 68.5
```

Wir möchten außerdem wissen, wie niedlich die übelste Katze in meinem Umfeld ist.

Lasst euch den niedrigsten Wert in der Spalte *cuteness* ausgeben.

mögliche Lösungen:

```
range(cat_ranking$cuteness)
```

```
> 2 100
```

```
min(cat_ranking$cuteness)
```

```
> 2
```

```
summary(cat_ranking$cuteness)
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
2.00	28.25	68.50	59.75	100.00	100.00	

Basics in R

Rechenoperationen:

Addieren: $a + b$

Subtrahieren: $a - b$

Multiplizieren: $a * b$

Dividieren: a / b

Potenz: a^b

Was ist das eigentlich
für ein komisches Format?

Wurzel ziehen: `sqrt(a)`

Mittelwert: `mean(vector)`

Median: `median(vector)`

Standardabweichung: `sd(vector)`

Minimum und Maximum in einem Vektor finden: `range(vector)`

Minimum oder maximum in einem Vektor finden: `min(vector)` bzw. `max(vector)`

Anzahl von Elementen in einem Vektor zählen: `length(vector)`

Verteilung des Vectors zusammenfassen: `summary(vector)`

Basics in R: Funktionen

Eine Gärtnerin pflanzt Mango- und Limettenbäume. Sie weiß, dass jeder Mangobaum im Schnitt 21 Früchte tragen wird und jeder Limettenbaum 100. Sie möchte wissen, wie viele Früchte sie insgesamt ernten wird.

```
# eigene Funktion schreiben  
comp_harvest <- function(mangotrees, limetrees){ # Argumente definieren  
}  
# hier kommt der Rest der Funktion
```

Basics in R: Funktionen

Eine Gärtnerin pflanzt Mango- und Limettenbäume. Sie weiß, dass jeder Mangobaum im Schnitt 21 Früchte tragen wird und jeder Limettenbaum 100. Sie möchte wissen, wie viele Früchte sie insgesamt ernten wird.

```
# eigene Funktion schreiben  
comp_harvest <- function(mangotrees, limetrees){ # Argumente definieren  
  # mit Argumenten rechnen:  
  mangos <- mangotrees * 21  
  limes <- limetrees * 100  
  harvest <- mangos + limes  
}
```

Basics in R: Funktionen

Eine Gärtnerin pflanzt Mango- und Limettenbäume. Sie weiß, dass jeder Mangobaum im Schnitt 21 Früchte tragen wird und jeder Limettenbaum 100. Sie möchte wissen, wie viele Früchte sie insgesamt ernten wird.

```
# eigene Funktion schreiben
comp_harvest <- function(mangotrees, limetrees){ # Argumente definieren
  # mit Argumenten rechnen:
  mangos <- mangotrees * 21
  limes <- limetrees * 100
  harvest <- mangos + limes
  # neuen Wert ausgeben lassen:
  return(harvest)
}
```

Basics in R: Funktionen

Eine Gärtnerin pflanzt Mango- und Limettenbäume. Sie weiß, dass jeder Mangobaum im Schnitt 21 Früchte tragen wird und jeder Limettenbaum 100. Sie möchte wissen, wie viele Früchte sie insgesamt ernten wird.

```
# eigene Funktion schreiben
comp_harvest <- function(mangotrees, limetrees){ # Argumente definieren
  # mit Argumenten rechnen:
  mangos <- mangotrees * 21
  limes <- limetrees * 100
  harvest <- mangos + limes
  # neuen Wert ausgeben lassen:
  return(harvest)
}

# eigene Funktion benutzen:
comp_harvest(mangotrees = 2, limetrees = 3) # 2 * 21 Mangos und 3 * 100 Limetten = 342 Früchte insg.
> [1] 342

# Argumentnamen müssen nicht genannt werden, nur die richtige Reihenfolge ist wichtig:
comp_harvest(2, 3)
> [1] 342
```

Basics in R: Funktionen

Optionale Argumente in Funktionen

Die Gärtnerin weiß, dass nur gesunde Limettenbäume Früchte tragen. Sie geht aber grundsätzlich davon aus, dass alle ihre Bäume gesund sind, sofern sie keinen kranken Baum findet.

```
comp_harvest <- function(mangotrees, limetrees, sick_limetrees = 0){ # Default-Wert für Zusatzargument
  # normal mit Argumenten rechnen:
  mangos <- mangotrees * 21
  limes <- (limetrees - sick_limetrees) * 100
  harvest <- mangos + limes
  # neuen Wert ausgeben lassen:
  return(harvest)
}
```

Basics in R: Funktionen

Optionale Argumente in Funktionen

Die Gärtnerin weiß, dass nur gesunde Limettenbäume Früchte tragen. Sie geht aber grundsätzlich davon aus, dass alle ihre Bäume gesund sind, sofern sie keinen kranken Baum findet.

```
comp_harvest <- function(mangotrees, limetrees, sick_limetrees = 0){ # Default-Wert für Zusatzargument
  # normal mit Argumenten rechnen:
  mangos <- mangotrees * 21
  limes <- (limetrees - sick_limetrees) * 100
  harvest <- mangos + limes
  # neuen Wert ausgeben lassen:
  return(harvest)
}
```

Funktion ohne Zusatzargument benutzen:

```
comp_harvest(mangotrees = 2, limetrees = 3) # kein Baum krank, nach wie vor 342 Früchte insg.
> [1] 342
```

Funktion mit Zusatzargument benutzen:

```
comp_harvest(mangotrees = 2, limetrees = 3, sick_limetrees = 1) # 1 Baum krank = 100 Früchte weniger
> [1] 242
```

Basics in R: Funktionen

Funktionen muss man nicht immer selbst schreiben, man kann auch packages mit fertigen Funktionen importieren:

```
install.packages(„yarrr“) # package installieren  
library(yarrr) # package laden
```

```
yarrr.guide() # optional bei einigen Packages: package guide anzeigen lassen  
help(package = yarrr)
```

The screenshot shows a dark-themed web page titled "yarrr package guide". It includes author information ("Nathaniel Phillips (yarrr.book@gmail.com)" and date "2017-04-18"), a main title "YaRrr! The pirate's guide to R", and a brief description of the package. It features sections for "Guides" (with links to "pirateplot()", "piratepal()", and "transparent()") and "Examples" (with a link to "pirateplot()"). A code snippet for "pirateplot()" is shown at the bottom.

The screenshot shows the R documentation for the "yarrr" package. It features a header with the title "A Companion to the e-Book \"YaRrr!: The Pirate's Guide to R\"". Below the header, there are links for "DESCRIPTION file" and "User guides, package vignettes and other documentation". A "Help Pages" section lists various functions: apa, auction, BeardLengths, capture, diamonds, examscores, movies, piratepal, pirateplot, pirates, pirateserrors, pircharter, poopdeck, recodev, transparent, and yarrr.guide. The "transparent" function is described as an "Open the HTML manual for the yarrr package".

Aufgaben für Euch

Wir möchten wissen, welche Katzen einen Niedlichkeit-Wert von 100 haben und ihre Namen aus dem Datensatz ziehen. Wie macht man sowas?

Indizieren

= Wie zieht man einzelne Werte oder Vektoren aus einem Objekt?

Basics in R

Indizieren

= Wie zieht man einzelne Werte oder Vektoren aus einem Objekt?

Option 1: eckige Klammern & Index des Objekts

`my_vector[7]` => Element an Stelle 7 in *my_vector*

1 2 3 4 5 6

`my_vector <- c(9, 2, 78, 61, 1, 32)`

`my_vector[3]`

> 78

`my_vector[3:5]`

> 78 61 1

Basics in R

Indizieren

= Wie zieht man einzelne Werte oder Vektoren aus einem Objekt?

Option 1: eckige Klammern & Index des Objekts

`my_vector[7]` => Element an Stelle 7 in `my_vector`

`my_df[1,]` => Zeile Nr 1 in `my_df`

`my_df[, 5]` => Spalte Nr 5 in `my_df`

```
my_df[2, ]  
  fruit  acidity  
> 2  apple  15
```

```
my_df[ , 1]  
> [1] "pear" "apple" "peach" "lemon"
```

```
my_df[2, 1]  
> [1] "apple"
```

my_df			
		fruit	acidity
	1	pear	5
	2	apple	15
	3	peach	2
	4	lemon	54

Basics in R

Indizieren

= Wie zieht man einzelne Werte oder Vektoren aus einem Objekt?

Option 2: eckige Klammern & Name des Objekts

`my_df[,, fruit"]` => gibt Spalte *fruit* aus

`my_df[fruit == "apple",]` => gibt alle Zeilen aus *my_df* aus, wo *fruit* == „apple“

```
my_df[,fruit"]
> [1] "pear" "apple" "peach" "lemon"

my_df[fruit == "apple"]
> [1] "apple"
```

	my_df	
	fruit	acidity
1	pear	5
2	apple	15
3	peach	2
4	lemon	54

Basics in R

Indizieren

= Wie zieht man einzelne Werte oder Vektoren aus einem Objekt?

Option 3: Dollar-Zeichen und Name der Spalte

`my_df$fruit` => Spalte *fruit* in data.frame `my_df`

```
my_df$fruit  
> [1] "pear" "apple" "peach" "lemon"
```

	my_df	
	fruit	acidity
1	pear	5
2	apple	15
3	peach	2
4	lemon	54

Basics in R

Indizieren

= Wie zieht man einzelne Werte oder Vektoren aus einem Objekt?

Option 4: subset-Funktion

`subset(my_df, fruit == „apple“)` => gibt alle Zeilen aus *my_df* aus, wo *fruit* == „apple“

```
subset( my_df, fruit == "apple")
      fruit    acidity
> 2   apple      15
```

```
subset( my_df, fruit == „apple“ | fruit == „lemon“)
      fruit    acidity
> 2   apple      15
> 4   lemon     54
```

		my_df
	fruit	acidity
1	pear	5
2	apple	15
3	peach	2
4	lemon	54

Basics in R

Indizieren

= Wie zieht man einzelne Werte oder Vektoren aus einem Objekt?

Sonderfall:

Option 5: which-Funktion

`which(my_df$fruit == „apple“)` => gibt den Index aller Zeilen aus, wo *fruit* == „apple“

```
which( my_df$fruit == "apple")
```

```
> 2
```

kann dann z.B. zum Indizieren mit eckigen Klammern
genutzt werden:

```
my_df[which( my_df$fruit == "apple"), ]
```

	fruit	acidity
> 2	apple	15

	my_df		
	fruit	acidity	
1	pear	5	
2	apple	15	
3	peach	2	
4	lemon	54	

Übersicht zum Indizieren

= Wie zieht man einzelne Werte oder Vektoren aus einem Objekt?

Option 1: eckige Klammern & Index des Objekts

`my_vector[7]` => Element an Stelle 7 in *my_vector*

`my_df[1,]` => Zeile Nr 1 in *my_df*

`my_df[, 5]` => Spalte Nr 5 in *my_df*

Option 2: eckige Klammern & Name des Objekts

`my_df[,, fruit"]` => gibt Spalte *fruit* aus

`my_df[fruit == "apple",]` => gibt alle Zeilen aus *my_df* aus, wo *fruit* == “apple”

Option 3: Dollar-Zeichen und Name der Spalte

`my_df$fruit` => Spalte *fruit* aus dem data.frame *my_df*

Option 4: subset-Funktion

`subset(my_df, fruit == „apple“)` => gibt alle Zeilen aus *my_df* aus, wo *fruit* == “apple”

Sonderfall:

Option 5: which-Funktion

`which(my_df$fruit == „apple“)` => gibt den Index aller Zeilen aus, wo *fruit* == “apple”

Aufgaben für Euch

Lasst euch die Namen der Katzen ausgeben, die in der Spalte „cuteness“ einen Wert von 100 haben.

Aufgaben für Euch

Lasst euch die Namen der Katzen ausgeben, die in der Spalte „cuteness“ einen Wert von 100 haben.

Mögliche Lösungen:

`subset(cat_ranking, cuteness == 100)$name`

oder

`cat_ranking[cuteness == 100,]$name`

oder

`cat_ranking[which(cat_ranking$cuteness == 100), "name"]`

Basics in R: if-„Loops“

Nutzt man, wenn man Teile seines Codes nur ausführen will, wenn eine bestimmte Bedingung gegeben ist (oder mehrere Bedingungen).

```
a <- 1  
b <- 2
```

```
if (a == b) {  
  c <- 1  
  
} else if (a < b) {  
  c <- 2  
  
} else { # if a > b  
  c <- 3  
  
}
```

Was ist c?

Basics in R: if-„Loops“

Nutzt man, wenn man Teile seines Codes nur ausführen will, wenn eine bestimmte Bedingung gegeben ist (oder mehrere Bedingungen).

```
a <- 1  
b <- 2
```

```
if (a == b) {  
  c <- 1  
  
} else if (a < b) {  
  c <- 2  
  
} else { # if a > b  
  c <- 3  
  
}
```

Was ist c? Richtig Antwort: 2

Logische Operatoren:

a ist gleich b: `a == b`

a ist ungleich b: `a != b`

a ist größer als b: `a > b`

a ist größer gleich b: `a >= b`

a ist kleiner als b: `a < b`

a ist kleiner gleich b: `a <= b`

a ist gleich b ODER c:

`a == b | a == c`

a ist gleich b UND c:

`a == b & a == c`

a ist ein NA: `is.na(a)`

A ist kein NA: `!is.na(a)`

Aufgaben für Euch

Legt eine Variable `a` an, die den Wert “`b` is smaller than 6“ hat. Legt eine Variable `b` an und weist ihr den folgenden Wert zu: `sample(1:10, 1)`

Schreibt einen if-„Loop“, der prüft, ob `b` größer als 5 ist. Falls `b` größer als 5 ist, soll der Text in `a` geändert werden zu “`b` is bigger than 5“

```
a <- 1  
b <- 2  
  
if (a == b) {  
  c <- 1  
} else if (a < b) {  
  c <- 2  
} else { # if a > b  
  c <- 3  
}
```

Hilfestellung:

`a` ist gleich `b`: `a == b`

`a` ist ungleich `b`: `a != b`

`a` ist größer als `b`: `a > b`

`a` ist größer gleich `b`: `a >= b`

`a` ist kleiner als `b`: `a < b`

`a` ist kleiner gleich `b`: `a <= b`

`a` ist gleich `b` ODER `c`:

`a == b | a == c`

`a` ist gleich `b` UND `c`:

`a == b & a == c`

`a` ist ein NA: `is.na(a)`

`A` ist kein NA: `!is.na(a)`

Aufgaben für Euch

Legt eine Variable *a* an, die den Wert “*b* is smaller than 6“ hat. Legt eine Variable *b* an und weist ihr den folgenden Wert zu: `sample(1:10, 1)`

Schreibt einen if-„Loop“, der prüft, ob *b* größer als 5 ist. Falls *b* größer als 5 ist, soll der Text in *a* geändert werden zu “*b* is bigger than 5“

Lösung:

```
a <- "b is smaller than 6"  
b <- sample(1:10, 1)  
if (b > 5){  
  a <- "b is bigger than 5"  
}
```

Basics in R: if-„Loops“

Nutzt man, wenn man Teile seines Codes nur ausführen will, wenn eine bestimmte Bedingung gegeben ist (oder mehrere Bedingungen).

Beispiel: Ich möchte alle VPn ausschließen, die angegeben haben, dass sie immer überall weiße Mäuse sehen, die anderen sollen nicht ausgeschlossen werden.

Beispieldatensatz *df*:

	code	seeing_mice	gender	age	mean_RT	exclude
1	VP1	FALSE	female	23	300.56	NA
2	VP2	TRUE	male	21	800.45	NA
3	VP3	FALSE	female	20	376.45	NA
4	VP4	FALSE	nonbinary	22	342.82	NA

Basics in R: if-„Loops“

Nutzt man, wenn man Teile seines Codes nur ausführen will, wenn eine bestimmte Bedingung gegeben ist (oder mehrere Bedingungen).

Beispiel: Ich möchte alle VPn ausschließen, die angegeben haben, dass sie immer überall weiße Mäuse sehen, die anderen sollen nicht ausgeschlossen werden.

Beispieldatensatz „df“:

	code	seeing_mice	gender	age	mean RT	exclude
1	VP1	FALSE	female	23	300.56	FALSE
2	VP2	TRUE	male	21	800.45	NA
3	VP3	FALSE	female	20	376.45	NA
4	VP4	FALSE	nonbinary	22	342.82	NA

```
if (df$seeing_mice[1] == TRUE){  
  df$exclude[1] <- TRUE  
} else {  
  df$exclude[1] <- FALSE  
}
```

Basics in R: if-„Loops“

Nutzt man, wenn man Teile seines Codes nur ausführen will, wenn eine bestimmte Bedingung gegeben ist (oder mehrere Bedingungen).

Beispiel: Ich möchte alle VPn ausschließen, die angegeben haben, dass sie immer überall weiße Mäuse sehen, die anderen sollen nicht ausgeschlossen werden.

Beispieldatensatz „df“:

	code	seeing_mice	gender	age	mean_RT	exclude
1	VP1	FALSE	female	23	300.56	FALSE
2	VP2	TRUE	male	21	800.45	TRUE
3	VP3	FALSE	female	20	376.45	NA
4	VP4	FALSE	nonbinary	22	342.82	NA

```
if (df$seeing_mice[2] == TRUE){  
  df$exclude[2] <- TRUE  
} else {  
  df$exclude[2] <- FALSE  
}
```

Basics in R: if-„Loops“

Nutzt man, wenn man Teile seines Codes nur ausführen will, wenn eine bestimmte Bedingung gegeben ist (oder mehrere Bedingungen).

Beispiel: Ich möchte alle VPn ausschließen, die angegeben haben, dass sie immer überall weiße Mäuse sehen, die anderen sollen nicht ausgeschlossen werden.

Beispieldatensatz „df“:

	code	seeing_mice	gender	age	mean_RT	exclude
1	VP1	FALSE	female	23	300.56	FALSE
2	VP2	TRUE	male	21	800.45	TRUE
3	VP3	FALSE	female	20	376.45	FALSE
4	VP4	FALSE	nonbinary	22	342.82	NA

```
if (df$seeing_mice[3] == TRUE){  
  df$exclude[3] <- TRUE  
} else {  
  df$exclude[3] <- FALSE  
}
```

Basics in R: if-„Loops“

Nutzt man, wenn man Teile seines Codes nur ausführen will, wenn eine bestimmte Bedingung gegeben ist (oder mehrere Bedingungen).

Beispiel: Ich möchte alle VPn ausschließen, die angegeben haben, dass sie immer überall weiße Mäuse sehen, die anderen sollen nicht ausgeschlossen werden.

Beispieldatensatz „df“:

	code	seeing_mice	gender	age	mean_RT	exclude
1	VP1	FALSE	female	23	300.56	FALSE
2	VP2	TRUE	male	21	800.45	TRUE
3	VP3	FALSE	female	20	376.45	FALSE
4	VP4	FALSE	nonbinary	22	342.82	FALSE

```
if (df$seeing_mice[4] == TRUE){  
  df$exclude[4] <- TRUE  
} else {  
  df$exclude[4] <- FALSE  
}
```

Basics in R: for-Loops

Nutzt man, wenn man Teile seines Codes eine bestimmte Anzahl von Malen wiederholen möchte (quasi Alternative zu Copy & Pasten von Code).

Beispiel: Ich möchte 10 Zufallszahlen „würfeln“ und die einzeln mit der print-Funktion ausgeben lassen.

Option 1:
Code 10x wiederholen

```
print( sample(1:6, 1) )  
print( sample(1:6, 1) )
```

Option 2:
for-Loop mit 10 Durchgängen

```
for (i in 1:10){  
  print( sample(1:6, 1) )  
}
```

Dieser Codeschnipsel
generiert eine
ganzzahlige Zufallszahl
zwischen 1 und 6

Option 3:
10 Zufallszahlen würfeln und mit for-Loop
einzelne ausgeben lassen

```
random_num <- sample(1:6, 10)  
  
for (i in random_num){  
  print(i)  
}
```

Basics in R: for-Loops

Anwendungsbeispiel: Ich möchte alle VPn ausschließen, die angegeben haben, dass sie immer überall weiße Mäuse sehen, die anderen sollen nicht ausgeschlossen werden.

Zusatz: Ich möchte aber nicht per Hand die Indizes in den Code einsetzen und ihn immer wieder ausführen.

Beispieldatensatz „df“:

	code	seeing_mice	gender	age	mean_RT	exclude
1	VP1	FALSE	female	23	300.56	NA
2	VP2	TRUE	male	21	800.45	NA
3	VP3	FALSE	female	20	376.45	NA
4	VP4	FALSE	nonbinary	22	342.82	NA

```
if (df$seeing_mice[1] == TRUE){  
  df$exclude[1] <- TRUE  
} else {  
  df$exclude[1] <- FALSE  
}
```

Basics in R: for-Loops

Anwendungsbeispiel: Ich möchte alle VPn ausschließen, die angegeben haben, dass sie immer überall weiße Mäuse sehen, die anderen sollen nicht ausgeschlossen werden.

Zusatz: Ich möchte aber nicht per Hand die Indizes in den Code einsetzen und ihn immer wieder ausführen.

```
for ( i in 1:length(df$seeing_mice) ){ # für jede Zeile im Datensatz df...
```

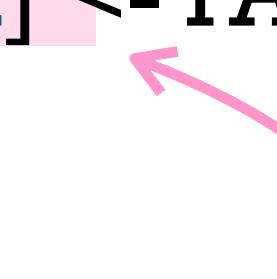
führe unseren Codeschnipsel aus:

```
if (df$seeing_mice[i] == TRUE){  
  df$exclude[i] <- TRUE  
} else {
```

```
  df$exclude[i] <- FALSE  
}
```

```
}
```

mit i als „Platzhalter“ für
den Index, den wir
sonst immer per Hand
ändern müssten



Basics in R: while-Loops

Anwendungsbeispiel: Ich möchte alle VPn ausschließen, die angegeben haben, dass sie immer überall weiße Mäuse sehen, die anderen sollen nicht ausgeschlossen werden.

Ich möchte aber nicht per Hand die Indices in den Code einsetzen und ihn immer wieder ausführen.

Zusatz: Ich möchte den Code nur so lange ausführen, bis ich die eine VP gefunden habe, die rausgeschmissen werden soll, weil ich weiß, dass es nur eine gibt.

Beispieldatensatz „df“ vor dem Loop:

	code	seeing_mice	gender	age	mean_RT	exclude
1	VP1	FALSE	female	23	300.56	NA
2	VP2	TRUE	male	21	800.45	NA
3	VP3	FALSE	female	20	376.45	NA
4	VP4	FALSE	nonbinary	22	342.82	NA

Beispieldatensatz „df“ nach dem while-Loop:

	code	seeing_mice	gender	age	mean_RT	exclude
1	VP1	FALSE	female	23	300.56	FALSE
2	VP2	TRUE	male	21	800.45	TRUE
3	VP3	FALSE	female	20	376.45	NA
4	VP4	FALSE	nonbinary	22	342.82	NA

Basics in R: while-Loops

Anwendungsbeispiel: Ich möchte alle VPn ausschließen, die angegeben haben, dass sie immer überall weiße Mäuse sehen, die anderen sollen nicht ausgeschlossen werden.

Ich möchte aber nicht per Hand die Indices in den Code einsetzen und ihn immer wieder ausführen.

Zusatz: Ich möchte den Code nur so lange ausführen, bis ich die eine VP gefunden habe, die rausgeschmissen werden soll.

```
i <- 1  
found_subject <- FALSE
```

```
while ( found_subject == FALSE){ # solange die VP nicht gefunden wurde...
```

```
# führe unseren Codeschnipsel aus:  
if (df$seeing_mice[i] == TRUE){  
  df$exclude[i] <- TRUE  
  found_subject <- TRUE # VP gefunden, beende Loop!  
} else {  
  df$exclude[i] <- FALSE  
  i = i + 1 # gehe zur nächsten Zeile  
}
```

```
}
```

Basics in R: while-Loops

Anwendungsbeispiel: Ich möchte alle VPn ausschließen, die angegeben haben, dass sie immer überall weiße Mäuse sehen, die anderen sollen nicht ausgeschlossen werden.

Ich möchte aber nicht per Hand die Indices in den Code einsetzen und ihn immer wieder ausführen.

Zusatz: Ich möchte den Code nur so lange ausführen, bis ich die eine VP gefunden habe, die rausgeschmissen werden soll.

```
i <- 1
found_subject <- FALSE

while ( found_subject == FALSE){ # solange die VP nicht gefunden wurde...

  # führe unseren Codeschnipsel aus:
  if (df$seeing_mice[i] == TRUE){
    df$exclude[i] <- TRUE
    found_subject <- TRUE # VP gefunden, beende Loop!
  } else {
    df$exclude[i] <- FALSE
    i = i + 1 # gehe zur nächsten Zeile
  }

}
```

Wichtig:
Bei while-Loops
immer an die
Abbruchbedingung
denken!

Basics in R: for-Loops

```
for ( i in 1:length(df$seeing_mice) ){ # für jede Zeile im Datensatz df...
```

führe unseren Codeschnipsel aus:

```
if (df$seeing_mice[i] == TRUE){  
  df$exclude[i] <- TRUE  
  break  
} else {  
  df$exclude[i] <- FALSE  
}  
}
```

Ungefährlicher
als while-Loops:
Ihr könnt einen Loop
auch abbrechen, indem
ihr statt einer
Abbruchbedingung
einfach *break* schreibt.

Analyseskript

Was machen wir jetzt?

Wir haben 12 Datensätze aus einem RSE-Experiment, die wir analysieren möchten.

- Daten einlesen
- wichtige Daten aus den Datensätzen ziehen und bereinigen
- Deskriptive Statistik
- Daten transformieren
- Inferenzstatistik
- Plot bauen

Das fertige Skript & die Datensätze findet ihr in unserem Springschool-Github-Repository!

<https://github.com/BioPsychKiel/SpringSchool2023/tree/main/Track1/day2>

Ressourcen zum R-Üben

{swirl} Learn R, in R.

- R-Package
- Tutorials nach Lernstand frei wählbar
- Aufgaben direkt in der Konsole lösen
- Erklärungen zu allem, was ihr tut

{swirl}

Learn R, in R.

swirl teaches you R programming and data science
interactively, at your own pace, and right in the R
console!

<https://swirlstats.com/>

YaRrr! The Pirate's Guide to R

- Online-Handbuch
- verständliche Statistik-Erklärungen
- lebensnahe Piraten-Beispiele
- Code-Schnipsel zum Copy & Pasten

YaRrr!



The Pirate's Guide to R

<https://bookdown.org/ndphillips/YaRrr/>

Springschool 2023

Track 1

Tag 2 - Nachmittag :

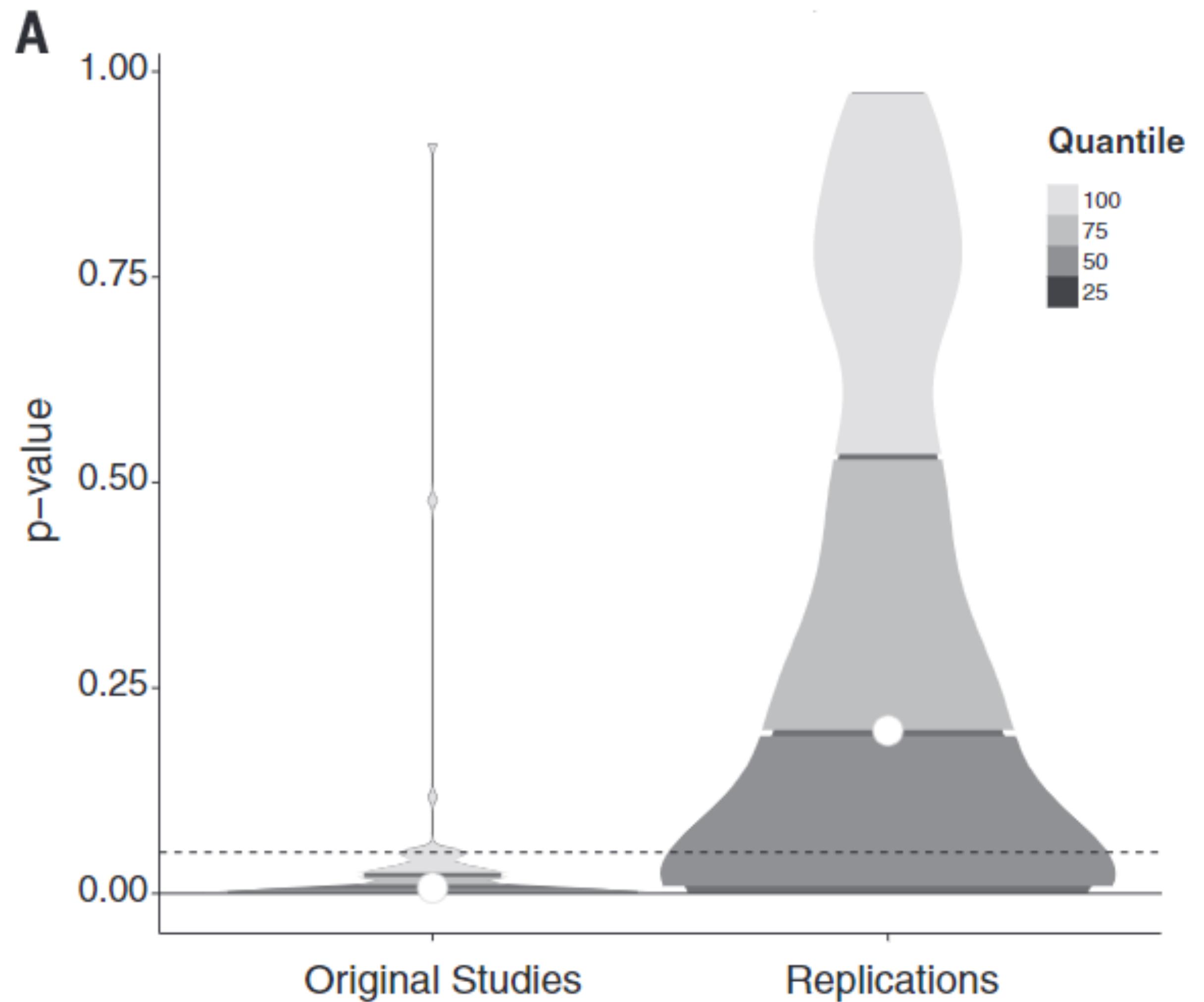
Versionierung mit Github

Replikationskrise

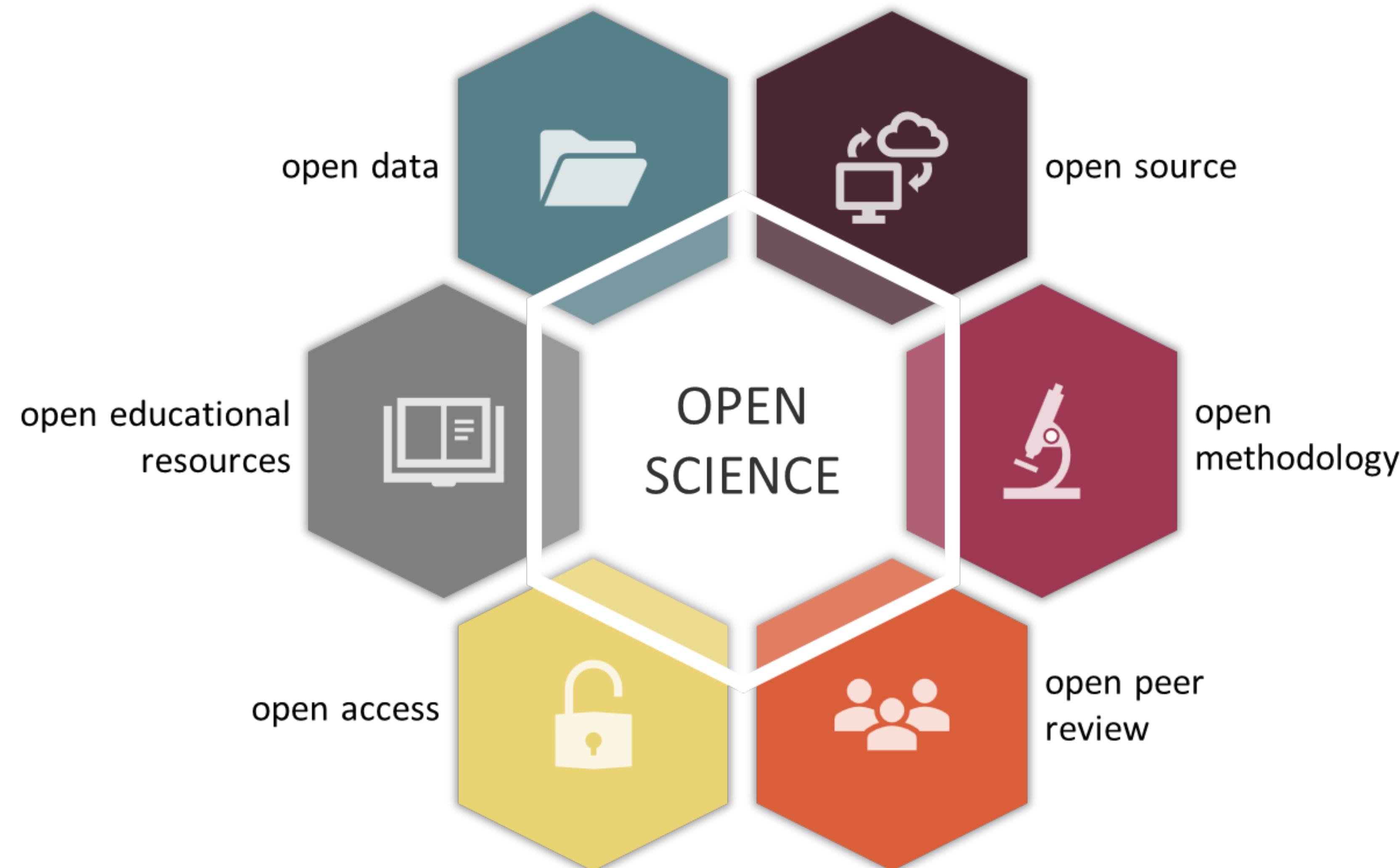
Replikation von 100 Studien mit signifikanten Ergebnissen

Fragestellung: Bei wie vielen Studien kann der p-Wert, die Effektstärke und Signifikanz erfolgreich repliziert werden?

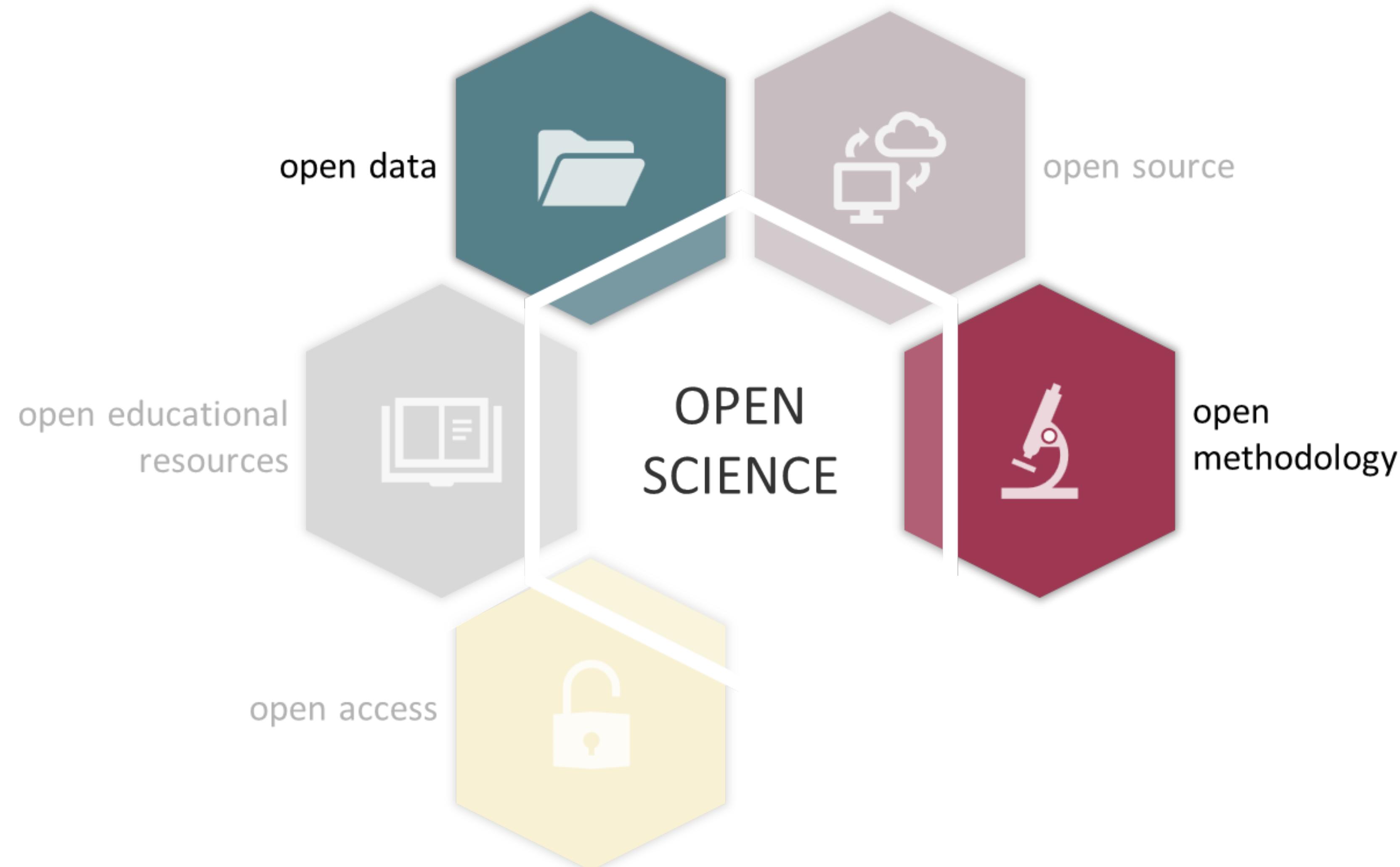
Ergebnis: nur 36% der Replikationen hatten signifikante Ergebnisse



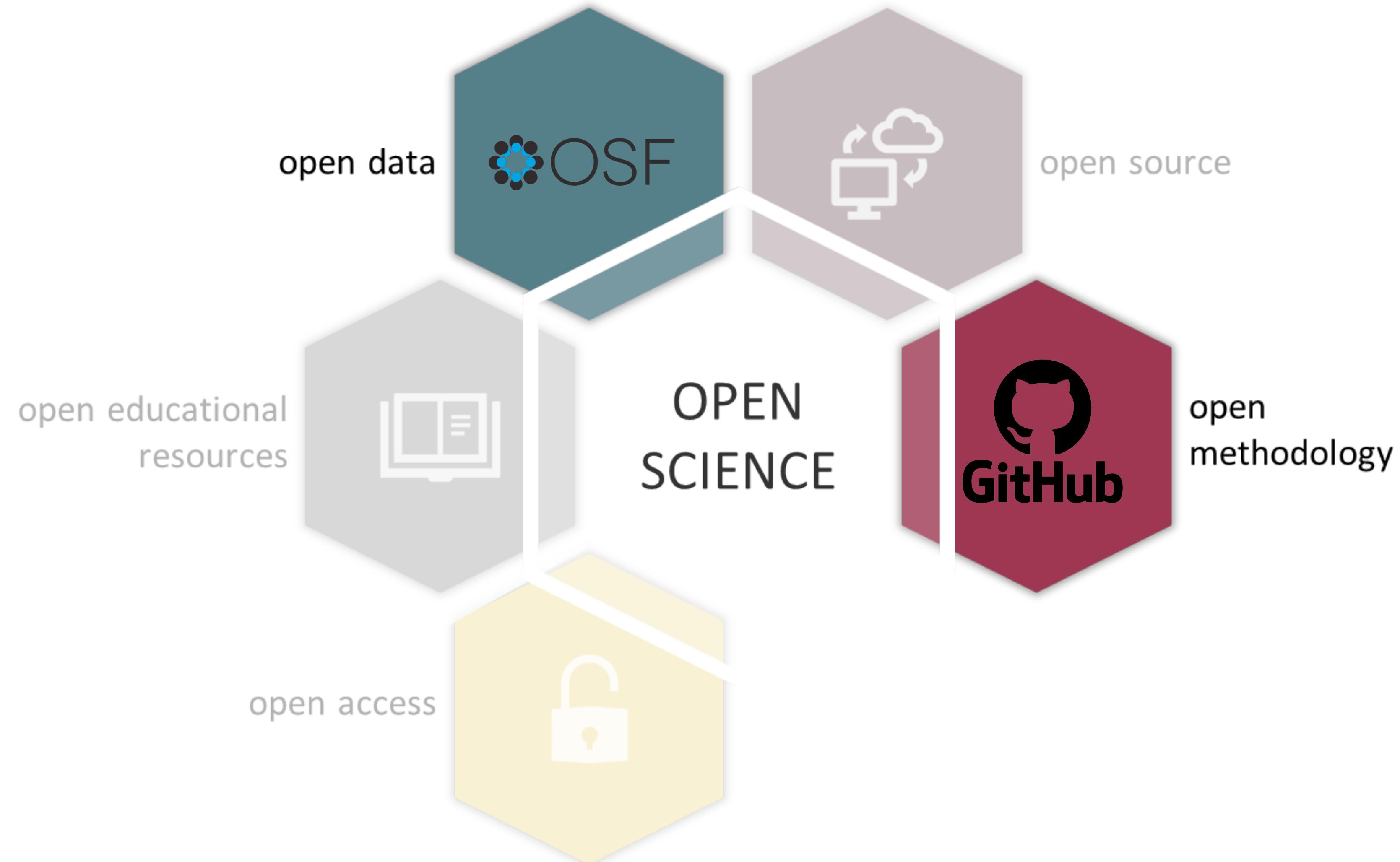
Open Science

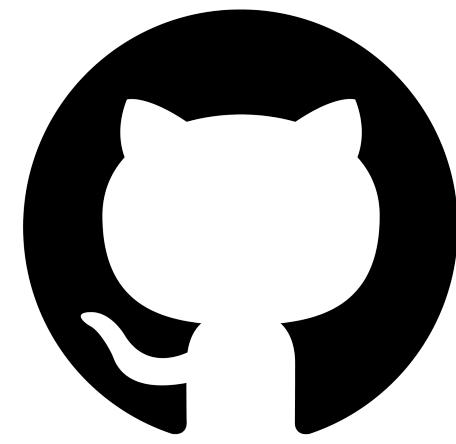


Open Science



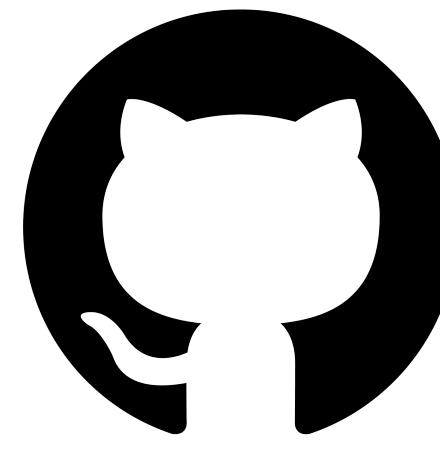
Open Science





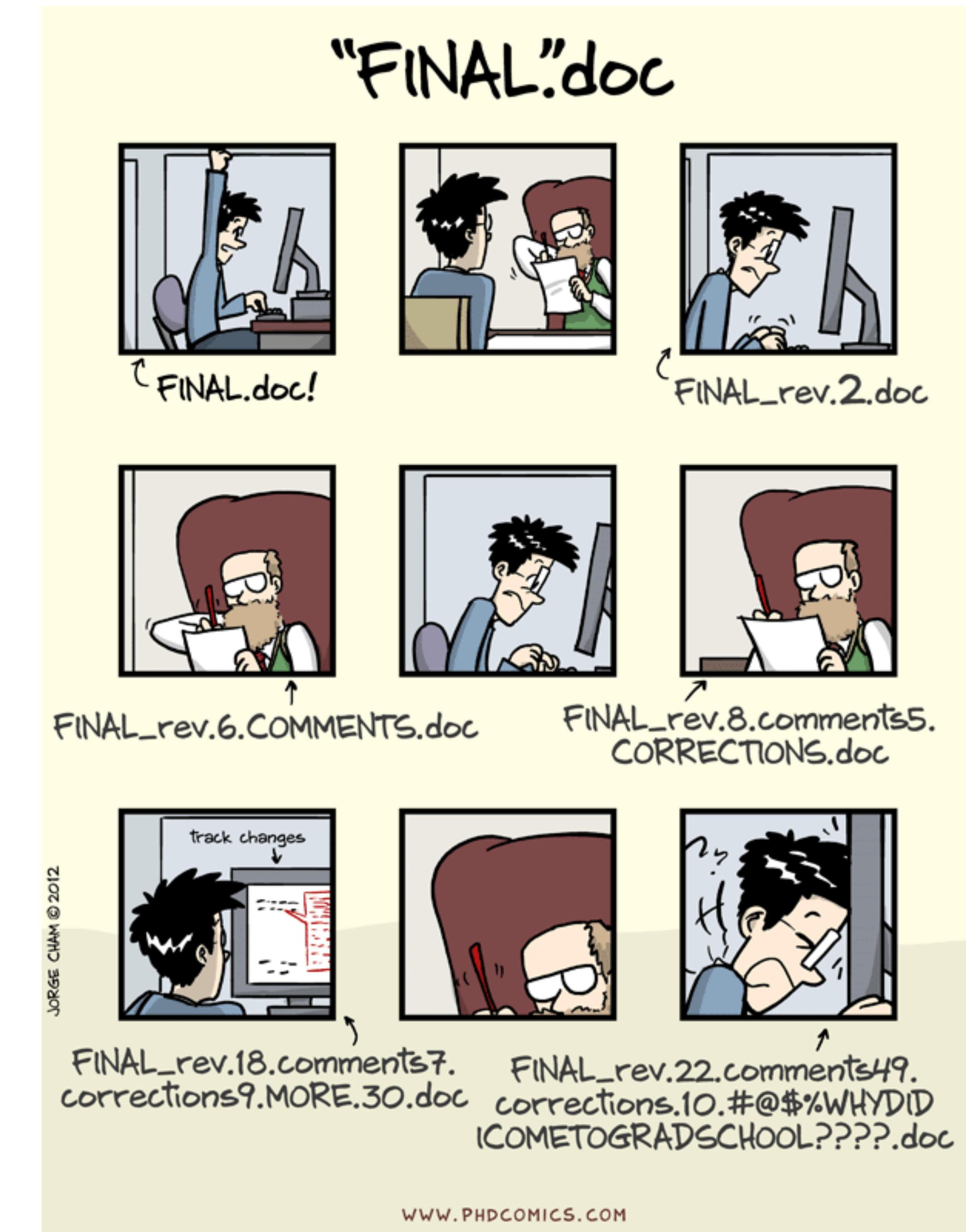
Github

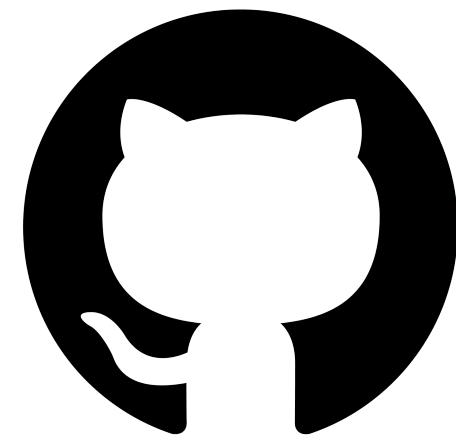
- Skripte & andere Ressourcen geordnet speichern
- Skripte & andere Ressourcen öffentlich teilen
- Versionskontrolle



Github

- Skripte & andere Ressourcen geordnet speichern
- Skripte & andere Ressourcen öffentlich teilen
- Versionskontrolle

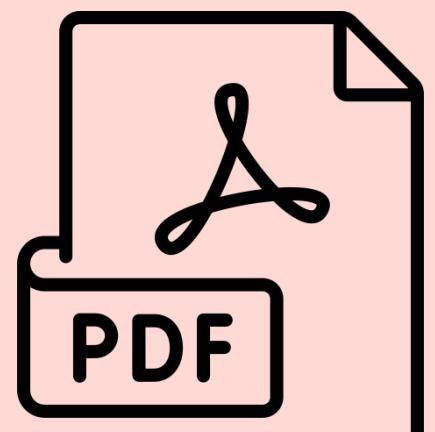




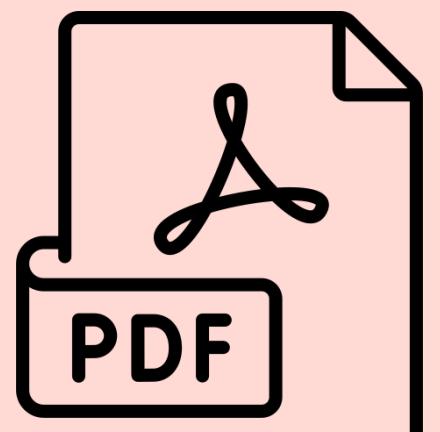
Github

„Versionskontrolle“
ohne Github:

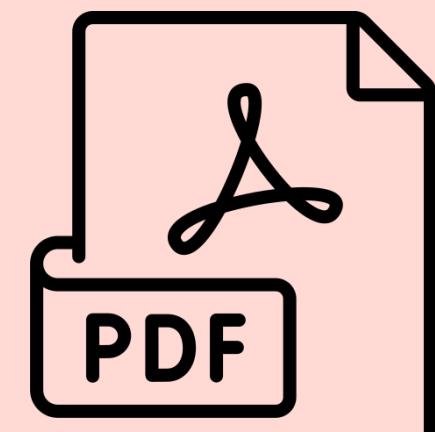
euer PC:



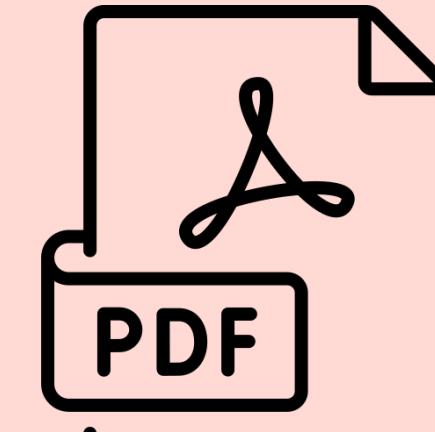
final.pdf



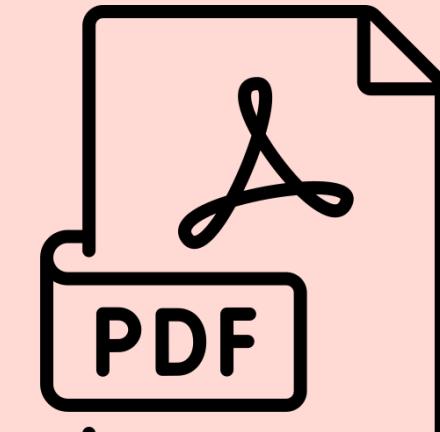
final_new.pdf



final_new2.pdf



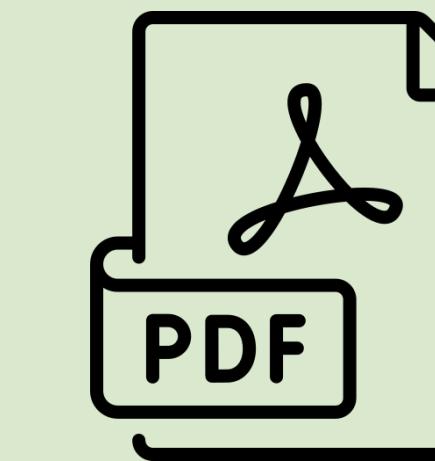
final_new2_
corrections.pdf



final_new2_
corrections_
_new_DONE.pdf

Versionskontrolle
mit Github:

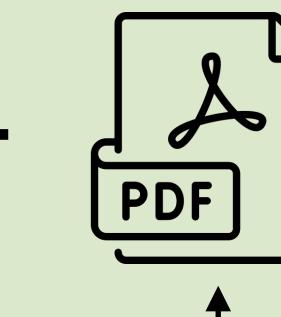
euer PC:



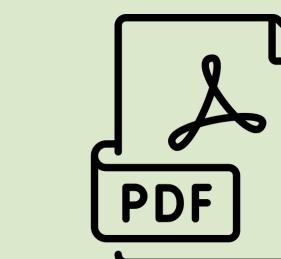
final.pdf



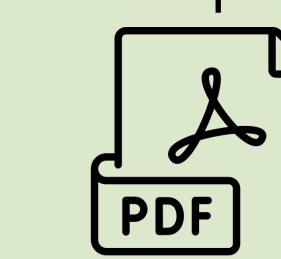
Github:



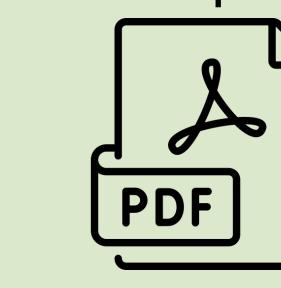
final.pdf
(Version 4)



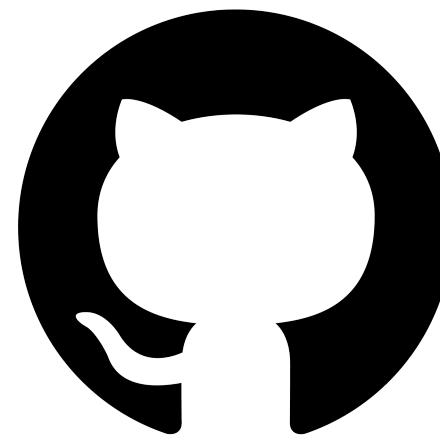
final.pdf
(Version 3)



final.pdf
(Version 2)



(Version 1)



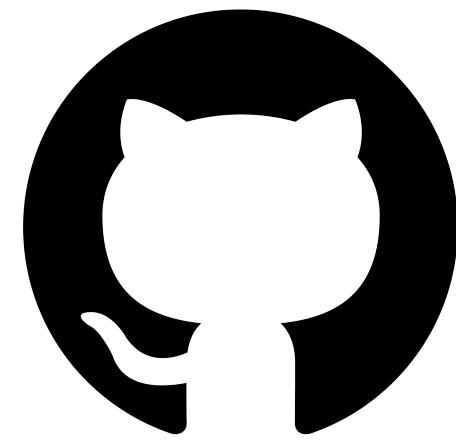
Github

The screenshot shows a GitHub profile page for a user named Merle Marie Schuckart. The top navigation bar includes links for Pull requests, Issues, Codespaces, Marketplace, Explore, Overview, and a highlighted Repositories (17) link. A blue bracket points from the text "Repositories = Ordner mit euren Dateien" to the Repositories link. The main area displays a circular profile picture of a brain scan, the user's name, and their title as a PhD student at the University of Lübeck. Below this is an "Edit profile" button and social links for Twitter and LinkedIn. A pink bracket points from the text "Überblick über eure Uploads" to a chart showing contributions per day and month over the last year. The chart indicates 486 contributions. A green bracket points from the text "Menü: Profil, Repositories, Projects, Organizations, Settings, Sign Out" to the user menu in the top right corner, which lists options like Signed in as, Set status, Your profile, Your repositories, and Sign out.

Repositories = Ordner mit euren Dateien

Menü: Profil, Repositories, Projects, Organizations, Settings, Sign Out

Überblick über eure Uploads



Github

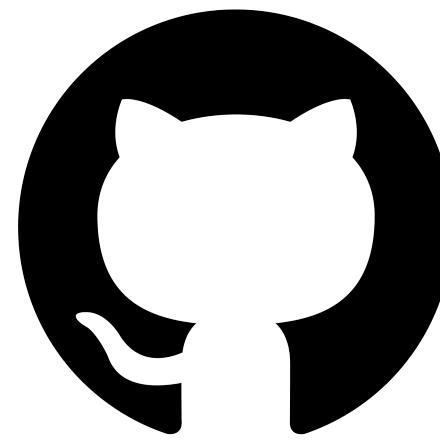
Name des
Repositories

Programmiersprache,
die in den hochgeladenen
Dateien hauptsächlich
genutzt wird

The screenshot shows the GitHub repository overview page with the following details:

- PhD** (Private): Python, Updated yesterday
- Masters_Thesis** (Public): Python, Updated on Apr 12, 2022
- int_trmr_eeg** (Private): Forked from JuliusWelzel/int_trmr_eeg. Pilot project to analyze EMG & behavioural data from tremor patients. C++, Updated on Sep 15, 2020.

Übersicht eurer
Aktivitäten im
Repository



Github

Public Repository
= öffentliches
Repository, das alle
Menschen im
Internet einsehen
können (auch ohne
Github-Account)

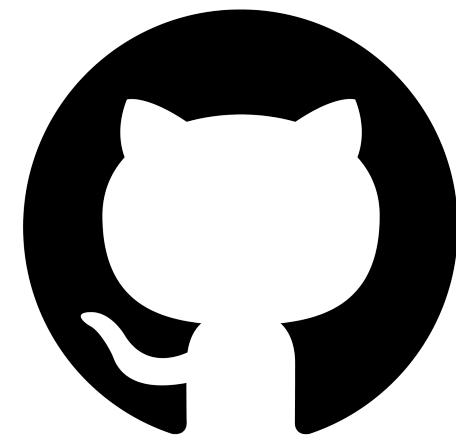
Private Repository
= privates Repository,
das nur die Leute
sehen können, die ich
als Contributors
hinzufüge

The screenshot shows the Github interface with the 'Repositories' tab selected. It displays three repositories:

- PhD** [Private] - Python, Updated yesterday. A pink box highlights the 'Private' status.
- Masters_Thesis** [Public] - Python, Updated on Apr 12, 2022. A blue box highlights the 'Public' status.
- int_trmr_eeg** [Private] - Forked from JuliusWelzel/int_trmr_eeg. Pilot project to analyze EMG & behavioural data from tremor patients. C++, Updated on Sep 15, 2020. A green box highlights the fork information.

Forked Repository
= Repository von einem anderen Github-Nutzer,
das ich kopiert habe, um daran zu arbeiten

Neues Repository
anlegen



Github

Wer soll euer Repo
sehen können?

Leere README-Datei für
das neue Repo erstellen?

.gitignore file
= Liste von Namen von
Dateien, die bei der
Versionierung ignoriert
werden sollen

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner *



MMarieSchuckart ▾

Repository name *



Great repository names are short and memorable. Need inspiration? How about [redesigned-octo-memory](#)?

Description (optional)

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file

This is where you can write a long description for your project. [Learn more](#).

Add .gitignore

Choose which files not to track from a list of templates. [Learn more](#).

.gitignore template: **None ▾**

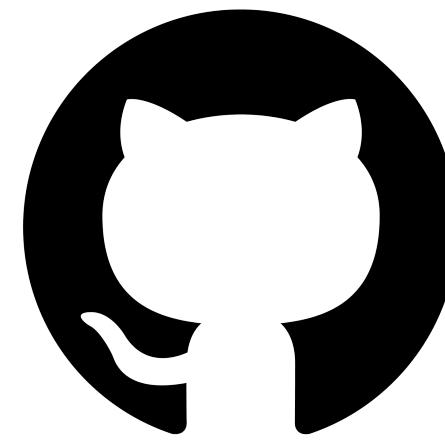
Choose a license

A license tells others what they can and can't do with your code. [Learn more](#).

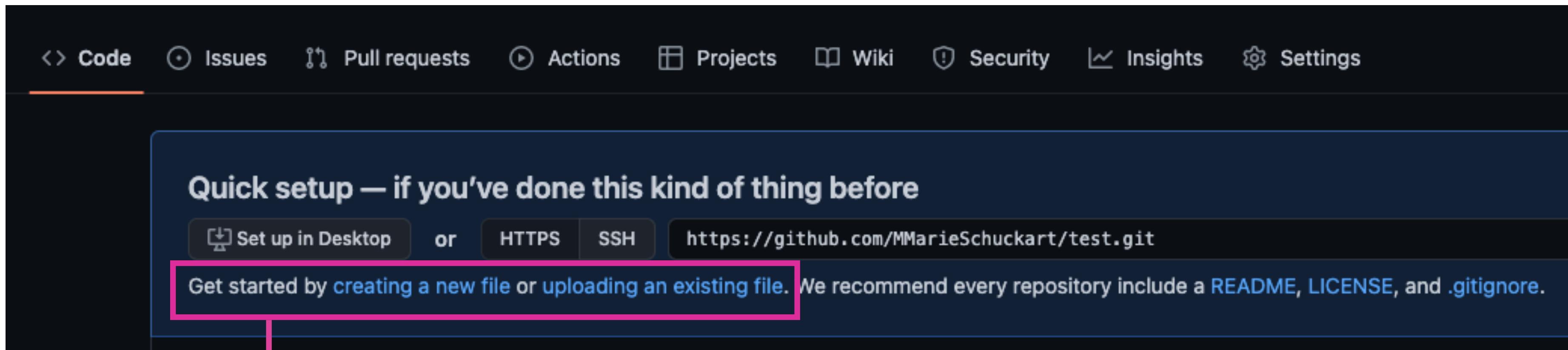
License: **None ▾**

ⓘ You are creating a public repository in your personal account.

Create repository

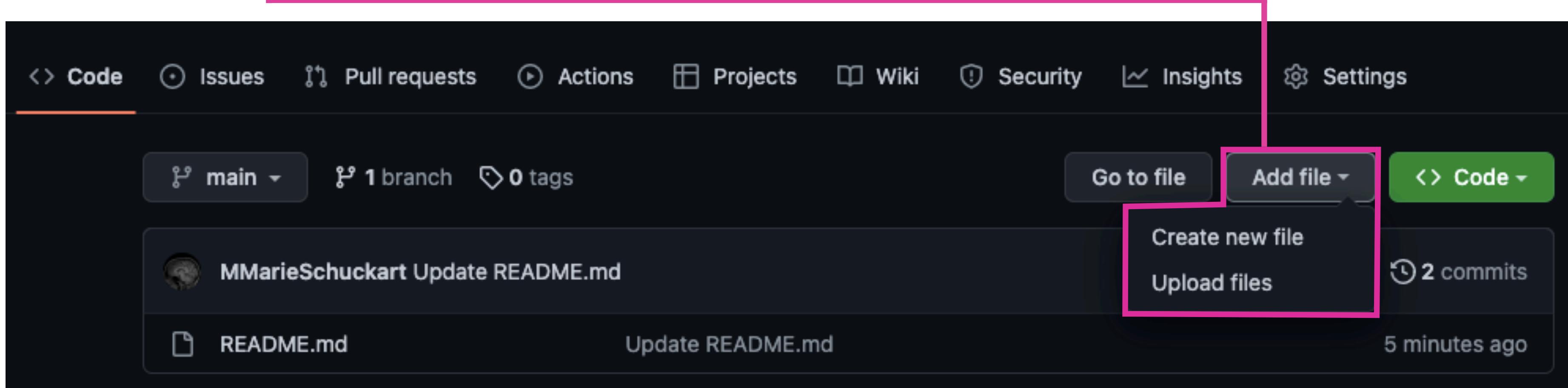


Github



The screenshot shows the GitHub interface for creating a new repository. At the top, there's a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation bar, a dark blue header box contains the text "Quick setup — if you've done this kind of thing before". It includes options to "Set up in Desktop" or "HTTPS" and an SSH link, along with the URL "https://github.com/MMarieSchuckart/test.git". A pink box highlights the instruction "Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.".

– Datei erstellen oder Datei hochladen



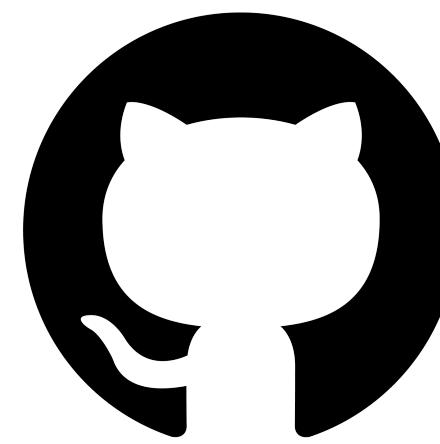
The screenshot shows the GitHub repository dashboard for "MMarieSchuckart/test". The top navigation bar is identical to the previous screenshot. The dashboard displays basic repository statistics: "main", "1 branch", and "0 tags". On the right, there are buttons for "Go to file", "Add file", and "Code". A pink box highlights the "Add file" button, which has a dropdown menu showing "Create new file" and "Upload files". Below the stats, a commit history is shown with one entry: "MMarieSchuckart Update README.md" (5 minutes ago). At the bottom, there are links for "README.md" and "Update README.md".

Aufgaben für Euch

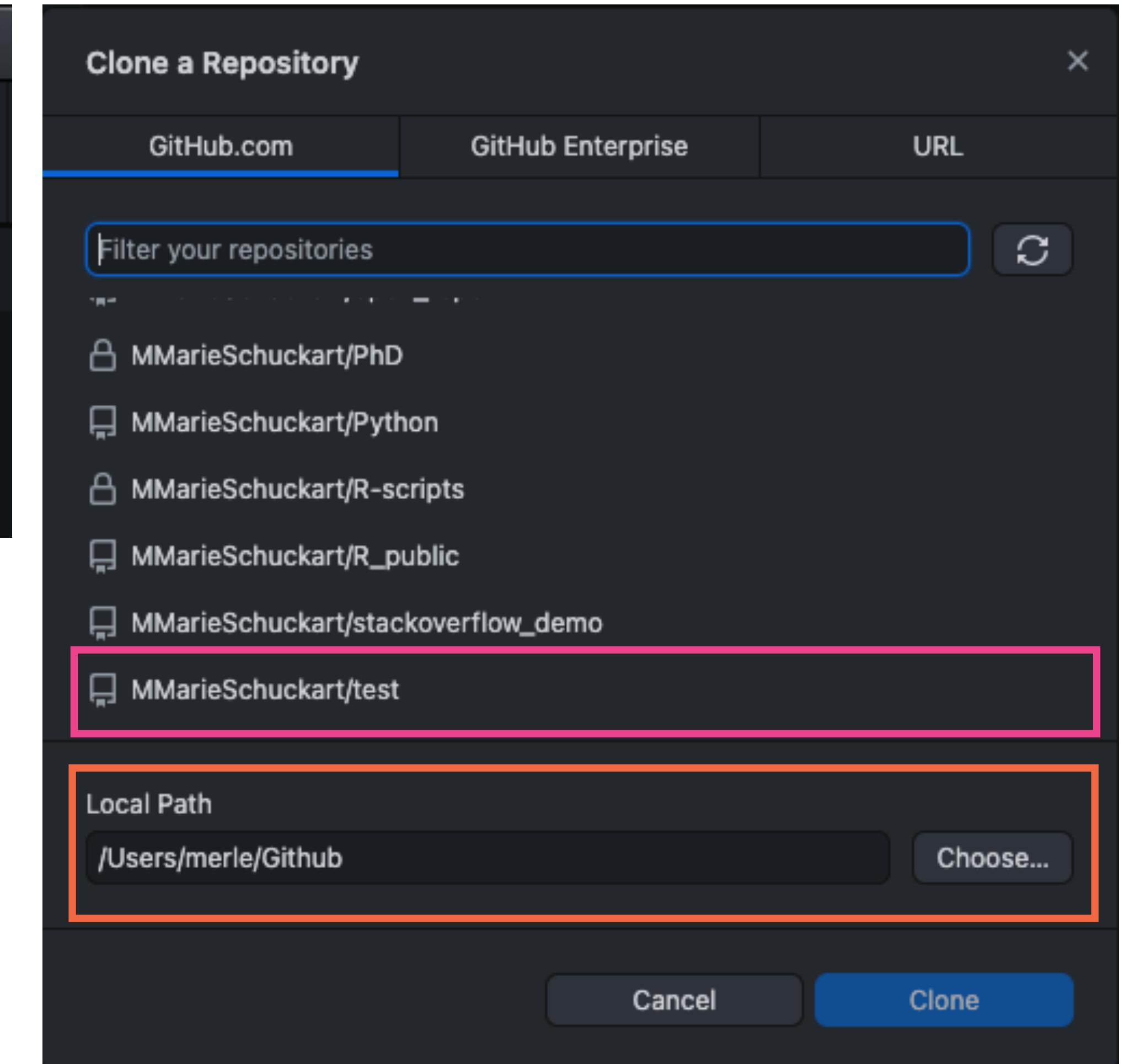
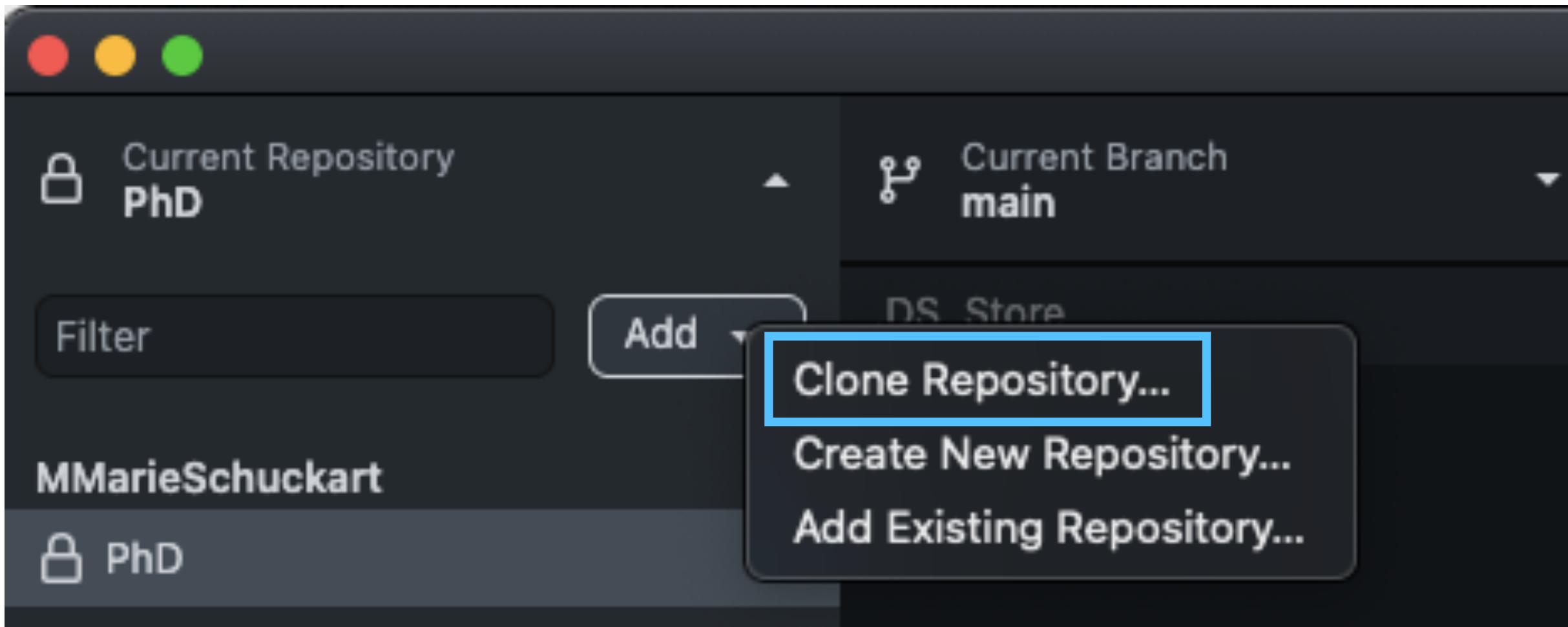
Legt euch ein Repository auf Github an.

Das Repository soll „Springschool 2023 R-Tutorial“ heißen und eine README.md-Datei enthalten.

Die erste Datei in eurem Repo soll das R-Skript R_Tutorial_live.R sein.



Github + Github Desktop



1. Repository klonen (von Github holen)
oder neu anlegen
2. Repository von Github auswählen
3. Pfad zu einem Ordner angeben, in dem
das Repo angelegt werden soll.

Aufgaben für Euch

Legt euch ein Repository auf Github an.

Das Repository soll „Springschool 2023 R-Tutorial“ heißen und eine README.md-Datei enthalten.

Die erste Datei in eurem Repo soll das R-Skript R_Tutorial_live.R sein.

Öffnet Github Desktop und klont euer Repository. Achtet darauf, dass ihr einen sinnvollen Pfad angebt, in dem der Ordner angelegt wird.

Aufgaben für Euch

Legt euch ein Repository auf Github an.

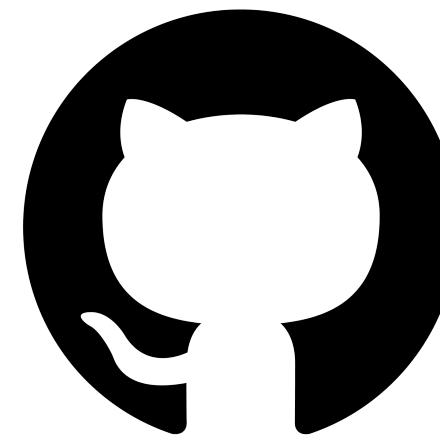
Das Repository soll „Springschool 2023 R-Tutorial_[euerName]“ heißen und eine README.md-Datei enthalten.

Die erste Datei in eurem Repo soll das R-Skript R_Tutorial_live.R sein.

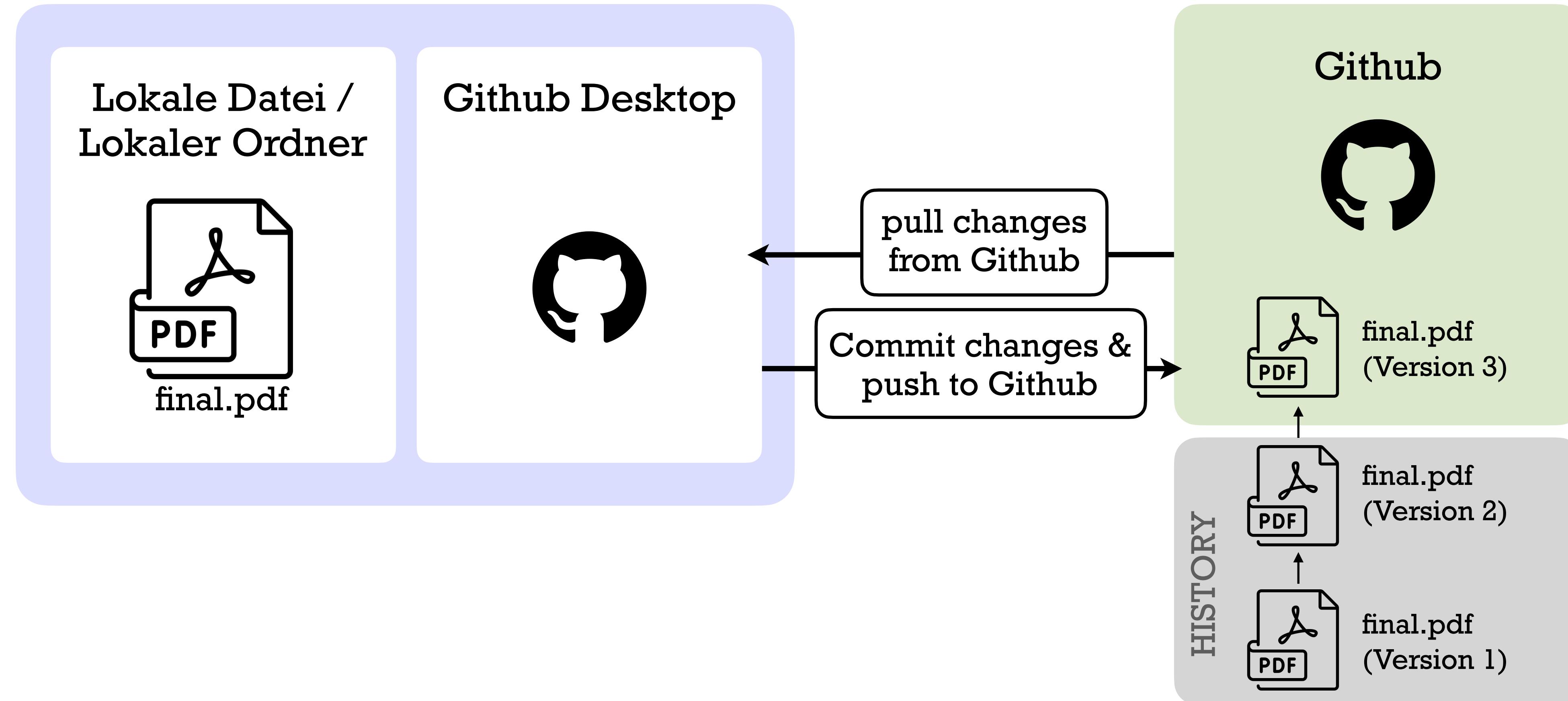
Öffnet Github Desktop und klont euer Repository. Achtet darauf, dass ihr einen sinnvollen Pfad angebt, in dem der Ordner angelegt wird.

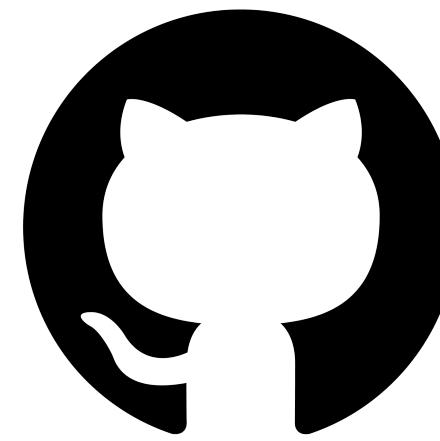
Öffnet R. Erstellt ein neues R-Skript namens „Versioning_Demo.R“ und speichert es lokal in eurem neuen Repo-Ordner. In das R-Skript schreibt ihr Folgendes:

```
# Hi, this is Version 1 of this script  
a <- 1  
b <- 2  
c <- a+b
```

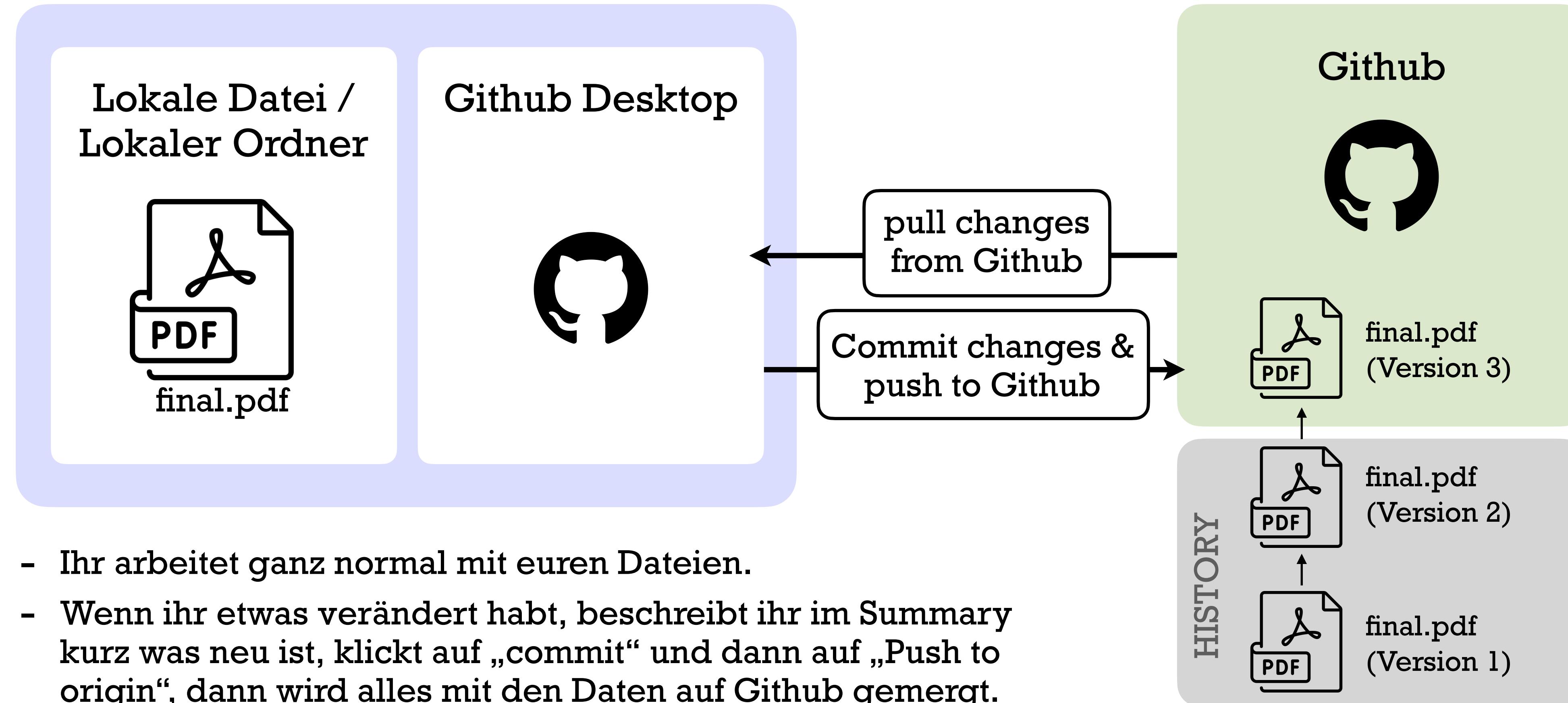


Github + Github Desktop





Github + Github Desktop



- Ihr arbeitet ganz normal mit euren Dateien.
- Wenn ihr etwas verändert habt, beschreibt ihr im Summary kurz was neu ist, klickt auf „commit“ und dann auf „Push to origin“, dann wird alles mit den Daten auf Github gemerkt.
- Wenn ihr aktuellere Versionen eurer Dateien von Github holen wollt, klickt ihr auf „Fetch origin“ und die Daten von Github werden mit euren lokalen Daten gemerkt.

Aufgaben für Euch

Erstellt ein neues R-Skript namens „Versioning_Demo.R“ und speichert es lokal in eurem neuen Repo-Ordner. In das R-Skript schreibt ihr Folgendes:

```
# Hi, this is Version 1 of this script  
a <- 1  
b <- 2  
c <- a+b
```

Öffnet wieder den Github-Desktop-Tab und schreibt euren ersten Commit. Pusht die Veränderungen auf Github.

Aufgaben für Euch

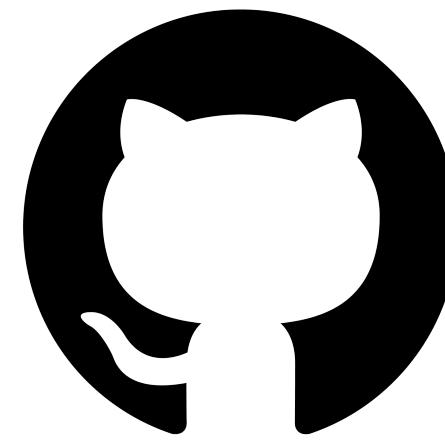
Erstellt ein neues R-Skript namens „Versioning_Demo.R“ und speichert es lokal in eurem neuen Repo-Ordner. In das R-Skript schreibt ihr Folgendes:

```
# Hi, this is Version 1 of this script  
a <- 1  
b <- 2  
c <- a+b
```

Öffnet wieder den Github-Desktop-Tab und schreibt euren ersten Commit. Pusht die Veränderungen auf Github.

Wechselt zu R. Weist in eurem kleinen Skript von eben der Variable a statt der 1 den Wert 3 zu und ändert den Kommentar zu „*Hi, this is Version 2 of this script*“. Speichert die Änderungen.

Öffnet nochmal Github Desktop und pusht die Änderungen nochmal zu Github.



Github

The screenshot shows a GitHub repository page for a file named `test / demo.R`. The main interface displays the file's content, which has been updated twice:

```
1 # Hi, this is version 2 of this script.  
2  
3 a <- 3  
4 b <- 2  
5 c <- a + b
```

Latest commit `0bf0b54` 2 minutes ago by **MMarieSchuckart** Update demo.R

1 contributor

5 lines (4 sloc) | 66 Bytes

A modal window titled "History for test / demo.R" is open, showing the commit history for February 4, 2023. It lists two commits:

- Update demo.R** (Verified) by **MMarieSchuckart** committed 9 minutes ago. This commit hash is highlighted with a blue box.
- Create demo.R** (Verified) by **MMarieSchuckart** committed 9 minutes ago. This commit hash is highlighted with a green box.

Below the commit list is a detailed view of the file diff between the two versions:

```
@@ -1,5 +1,5 @@  
- # Hi, this is version 1 of this script.  
+ # Hi, this is version 2 of this script.  
- a <- 1  
+ a <- 3  
- b <- 2  
+ b <- 2  
- c <- a + b  
+ c <- a + b
```

Annotations with arrows point from the UI elements to explanatory text:

- An arrow points from the "History" button in the main header to the text: "History = alte Versionen des Dokuments anschauen".
- An arrow points from the commit hash `0bf0b54` to the text: "Was wurde bei diesem Commit verändert?"
- An arrow points from the commit hash `15c839f` to the text: "Wie sah das Dokument zu diesem Zeitpunkt aus?"

History = alte Versionen
des Dokuments
anschauen

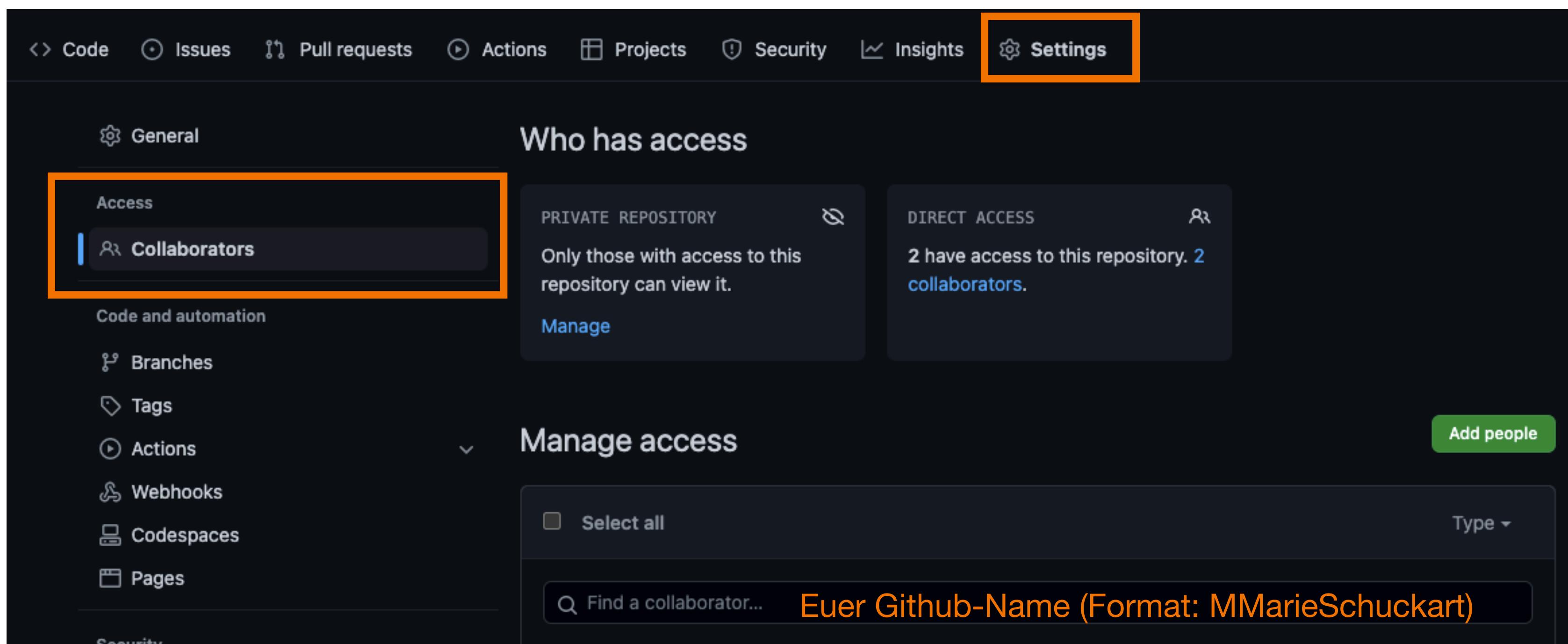
Was wurde
bei diesem
Commit
verändert?

Wie sah das
Dokument zu
diesem Zeitpunkt
aus?

Aufgaben für Euch

Bildet 2er-Teams. Bittet die jeweils andere Person aus eurem Team, euch als Contributor zu ihrem Repo hinzuzufügen. Ihr braucht dafür euren Github-Namen. Ihr bekommt eine E-Mail mit einer Einladung, am Repo mitzuarbeiten (die müsst ihr annehmen).

Wichtig: Lasst bitte erstmal nur eine einzige andere Person Contributor in eurem Repo sein und verändert selbst nichts an eurem R-Code während dieser Aufgabe.



Aufgaben für Euch

Wenn ihr erfolgreich zu einem fremden Repo hinzugefügt wurdet, könnt ihr es in Github Desktop klonen. Ihr seht jetzt zwei Repos und darüber ganz klein jeweils entweder den Github-Namen von euch oder von eurem Contributor.

Öffnet lokal das R-Skript *Versioning_Demo.R* aus dem fremden Repo und ändert etwas (egal was). Öffnet wieder Github Desktop und pusht eure Veränderungen online.

Sofern ihr selbst auch jemanden zu eurem Repo hinzugefügt habt, könnt ihr (wenn alle fertig sind) in Github Desktop auf „fetch“ und dann „pull from origin“ klicken.

Schaut in euren R-Code: Ihr seht jetzt auch lokal alle Veränderungen, die euer Contributor hochgeladen hat.

Aufgaben für Euch

In dieser Aufgabe tretet ihr euch nun beim Pushen auf die Füße, d.h. wir bauen mit Absicht einen *Merge-Conflict*. Ihr müsst dafür wieder zusammen arbeiten.

Entscheidet euch, wer Person A und wer Person B ist.

Ladet euch zuerst alle neuen Änderungen von Github herunter, falls ihr nicht die aktuellste Version des Dokuments auf eurem Laptop habt.

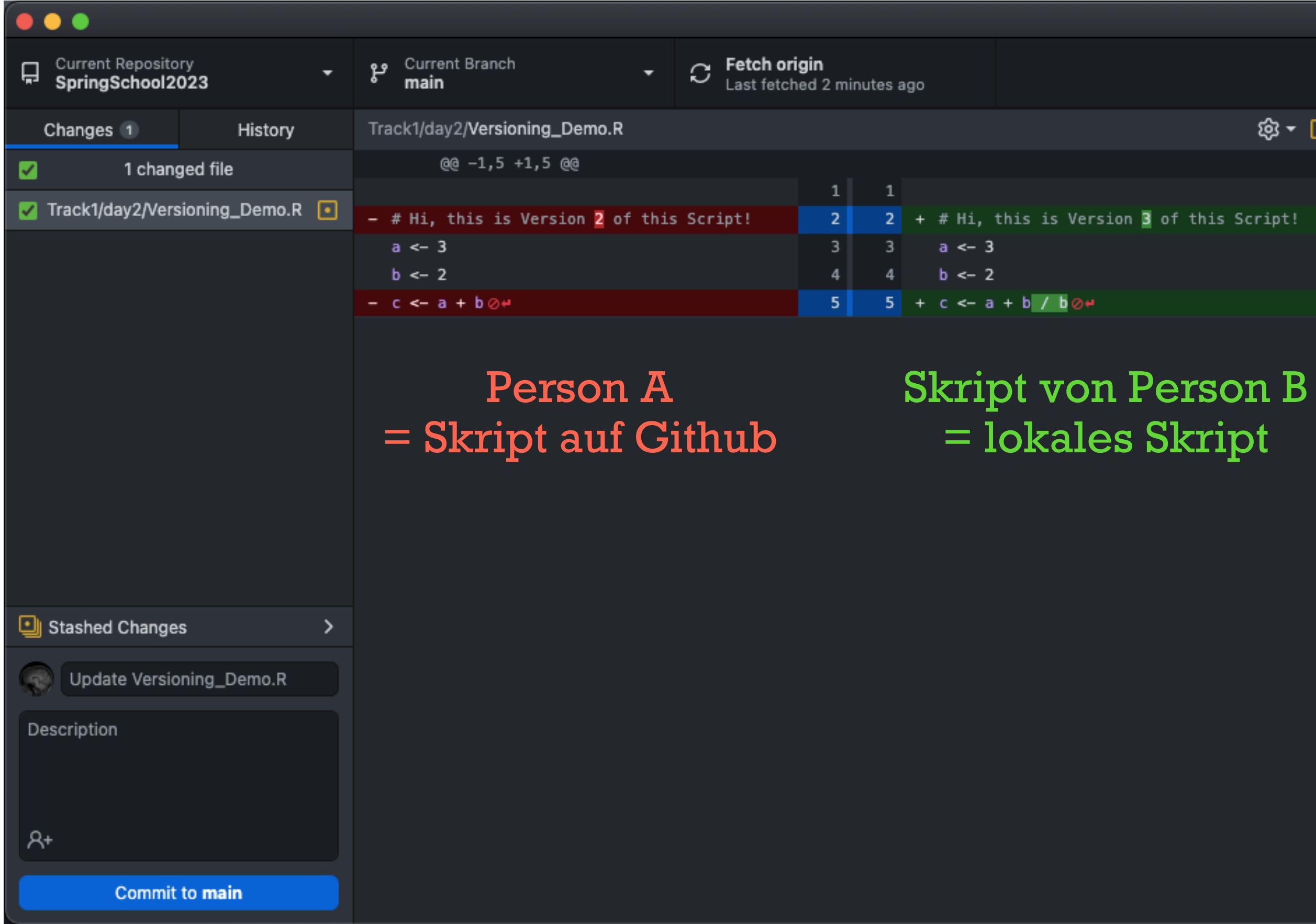
Person A schreibt in das R-Skript `Versioning_Demo.R` nun eine neue Zeile, z.B. einen Kommentar. Die Veränderungen pusht sie auf Github.

Setzt euch nun beide vor den Laptop von Person B.

Person B schreibt nun auch eine neue Zeile in das R-Skript. Versucht nun, die Veränderungen ebenfalls zu pushen. Was passiert?

Merge Conflict

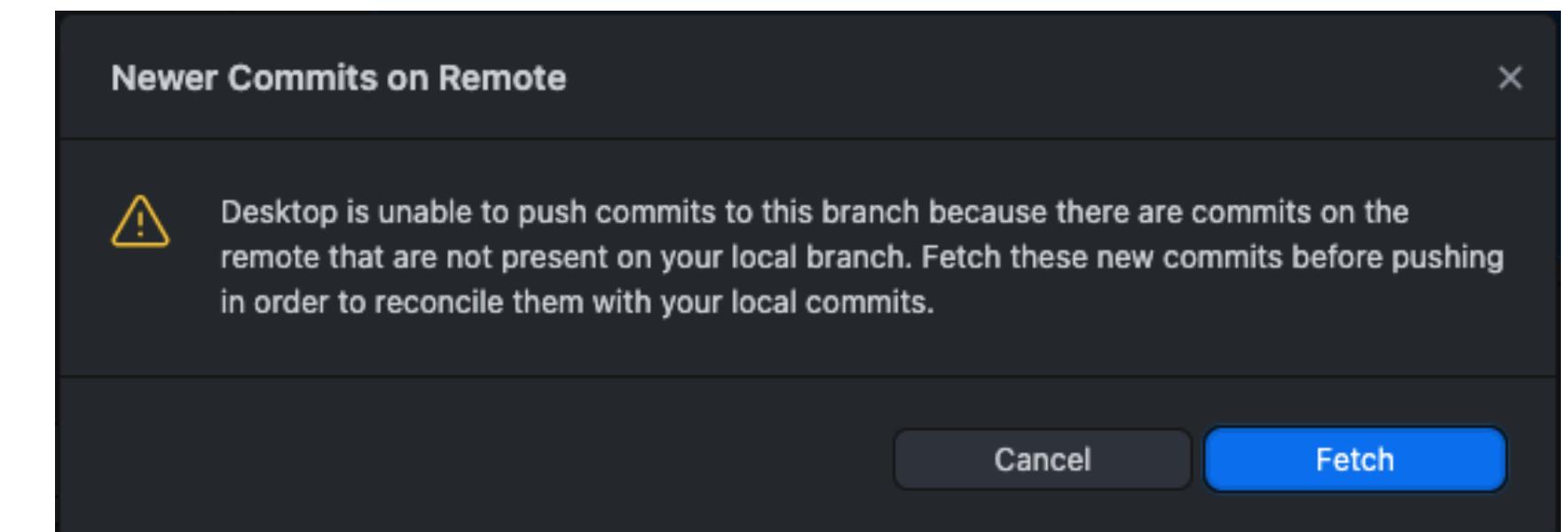
Noch vor eurem Commit müsstet ihr so etwas sehen:



Person A
= Skript auf Github

Skript von Person B
= lokales Skript

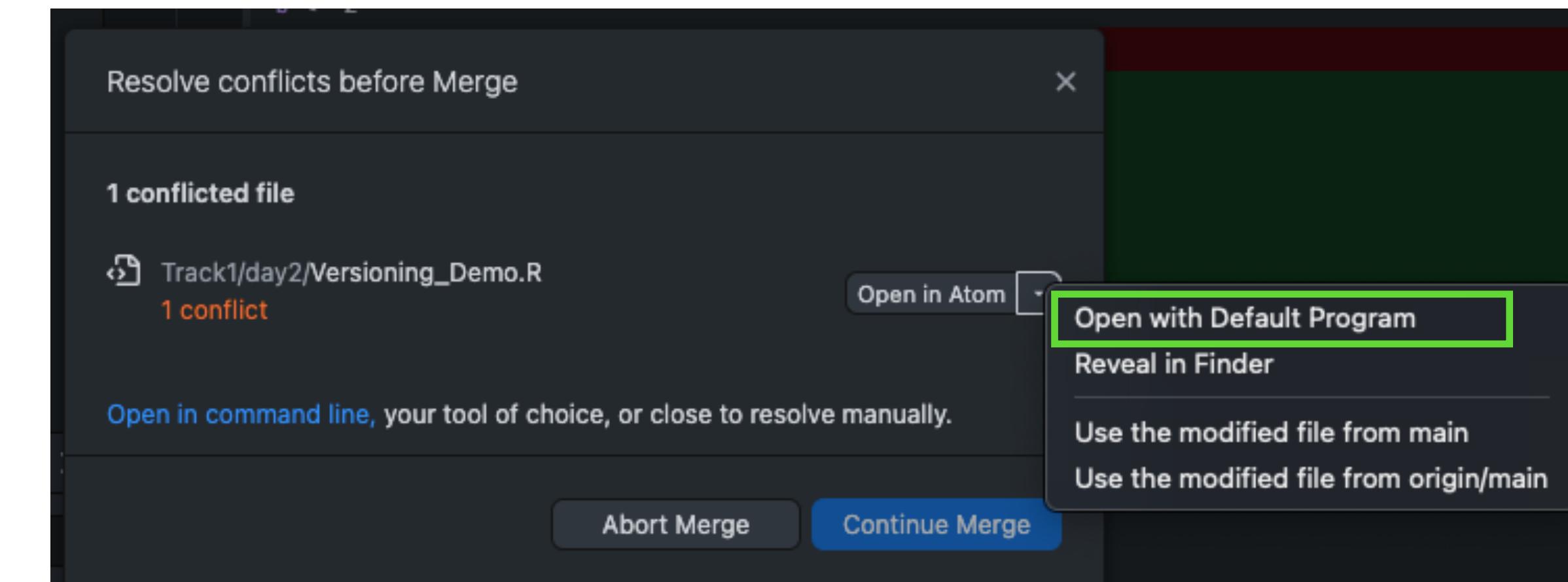
Nach dem Commit wird euch dann diese Fehlermeldung angezeigt:



Aufgaben für Euch

Nun löst ihr den Merge-Conflict:

Klickt in der Fehlermeldung auf *Fetch* und dann auf *pull origin*. Ihr kriegt nun diese Fehlermeldung angezeigt:



Klickt auf „Open with Default Program“. Github Desktop öffnet nun RStudio und zeigt euch in eurem Dokument eure Änderungen und die Änderungen von Person A untereinander an:

```
1
2 # Hi, this is Version 3 of this Script!
3 a <- 3
4 b <- 2
5 <<<<< HEAD
6 c <- a + b / b ] Änderungen von Person B (ihr selbst)
7 =====
8 c <- a / (b + a) ] Änderungen auf Github (von Person A)
9 >>>>> 2fc584e7923c2f95dd7ea334ca05f41ee0caf575
10
```

Aufgaben für Euch

Löscht alle Zeilen, die ihr nicht behalten möchtet.

Die folgenden Zeilen

```
<<<<<< HEAD  
=====  
>>>>> 2fc584e79...
```

könnt ihr auch löschen, das sind nur Markierungen für euch.

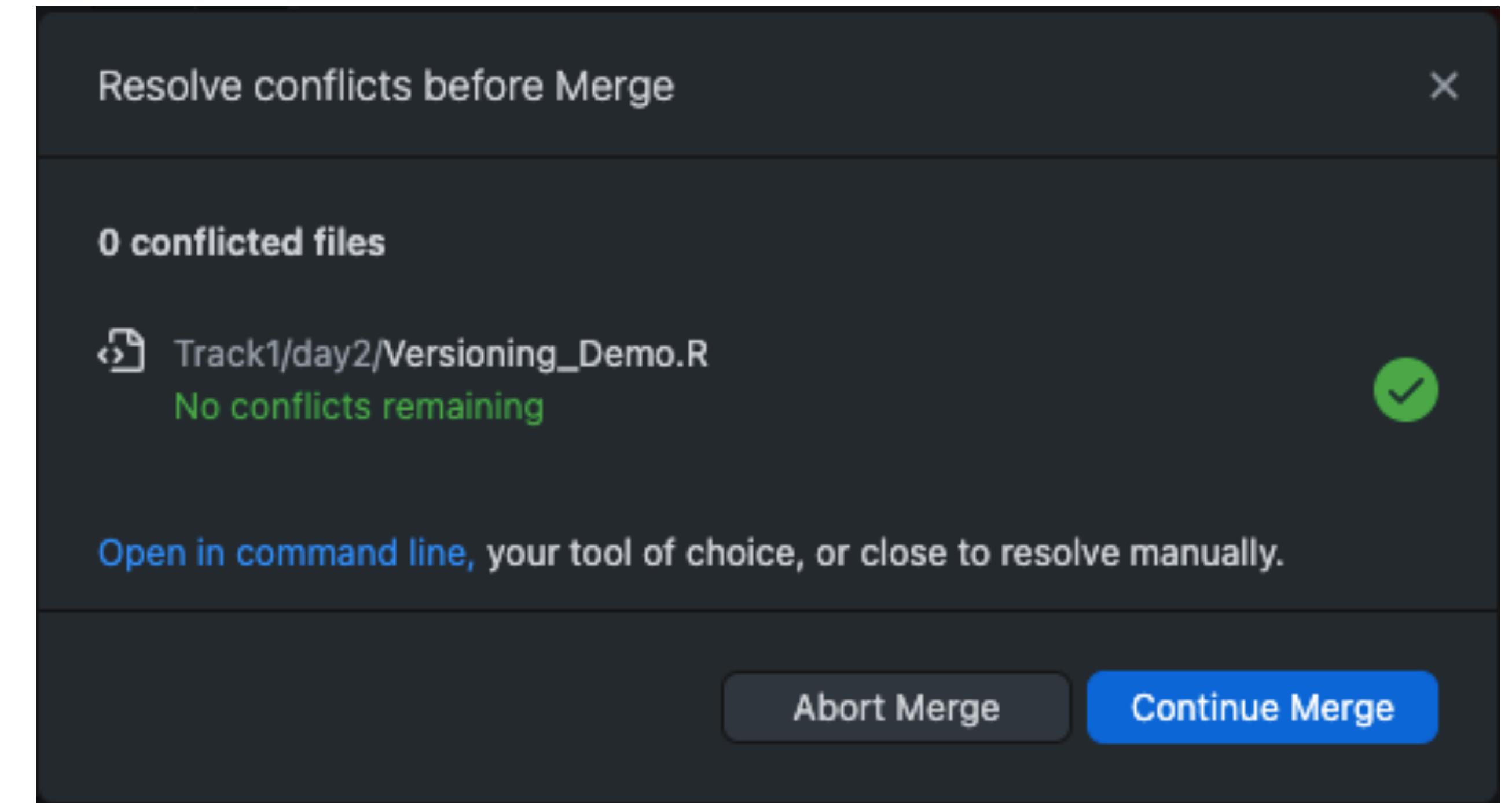
```
1  
2 # Hi, this is Version 3 of this Script!  
3 a <- 3  
4 b <- 2  
x 5 <<<<<< HEAD  
6 c <- a + b / b  
x 7 =====  
8 c <- a / (b + a)  
x 9 >>>>> 2fc584e7923c2f95dd7ea334ca05f41ee0caf575  
10
```

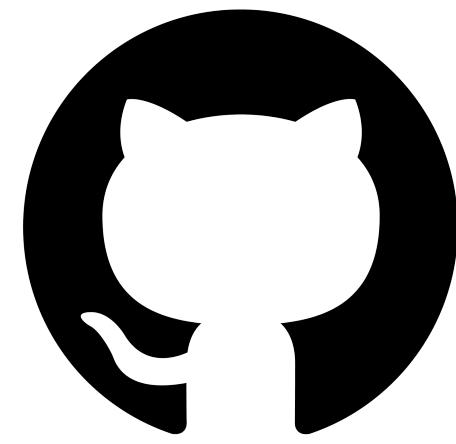
```
1  
2 # Hi, this is Version 3 of this Script!  
3 a <- 3  
4 b <- 2  
5  
6 c <- a + b / b  
7  
8  
9  
10
```

Aufgaben für Euch

Wenn ihr entschieden habt, welche Veränderungen bleiben sollen und welche nicht, könnt ihr das R-Skript speichern und wieder auf Github Desktop klicken. Ihr müsstet dort diese Nachricht sehen:

Klickt auf „Continue Merge“ und dann auf „push origin“. Fertig!



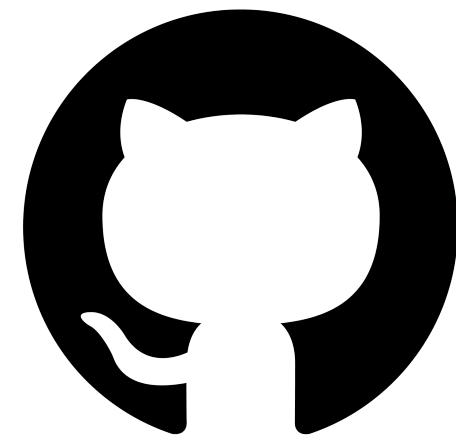


Github

Zusammenfassung:

Ihr könnt jetzt...

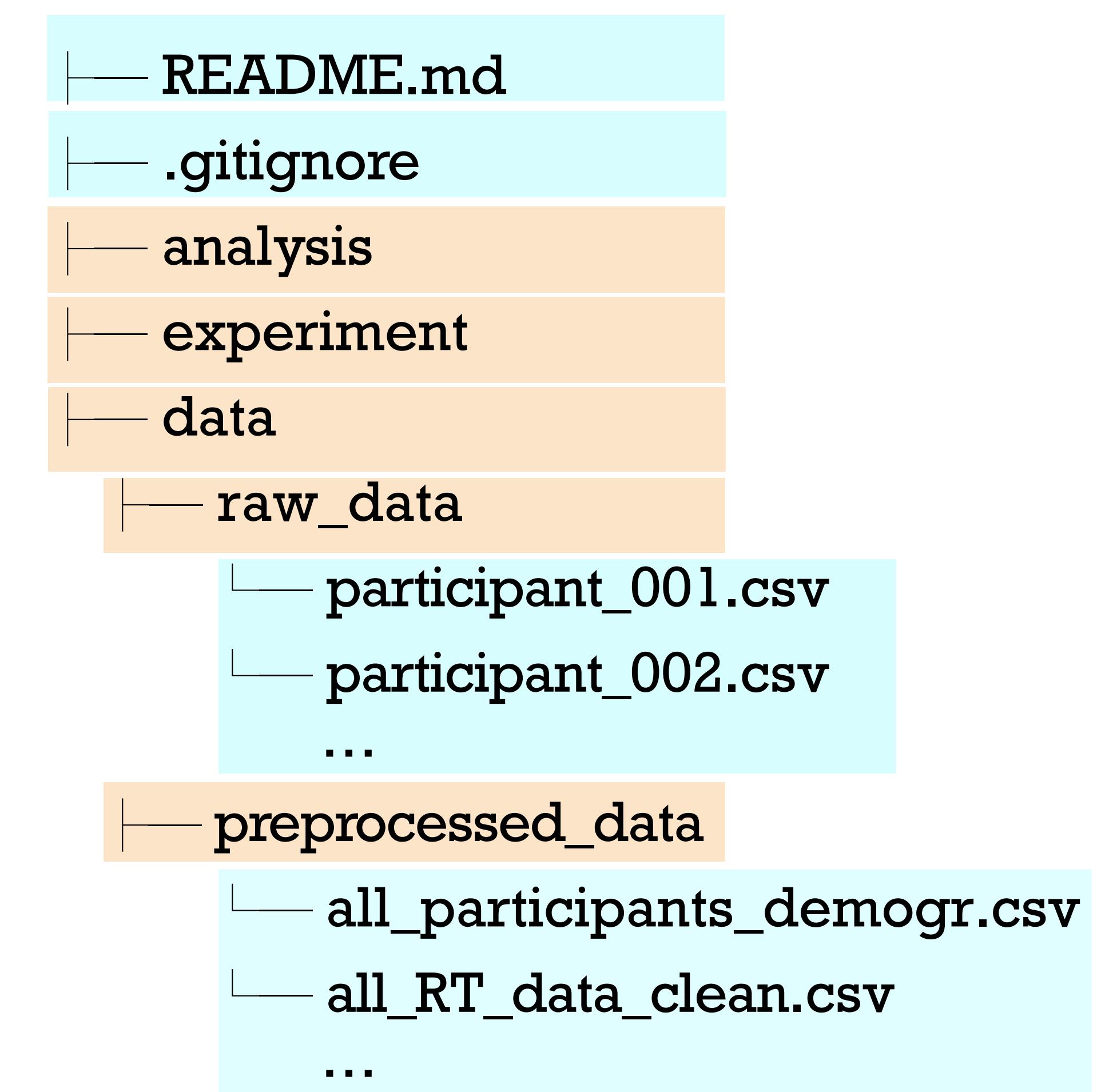
- Dateien lokal speichern (auch auf mehreren PCs) und mit Github synchronisieren.
- alte Versionen eurer Skripte einsehen und Gelöschtes retten.
- Skripte und Daten mit anderen teilen und gemeinsam an Projekten arbeiten.
- Merge-Conflicts lösen.
- auf eurem Lebenslauf mit euren Github-Skills angeben.

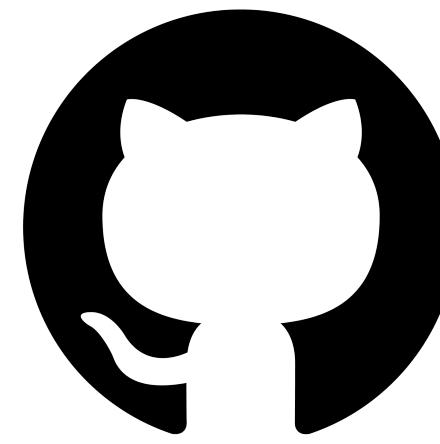


Github

Was macht ein gutes Repo aus?

- verständliche Dokumentation
- aufgeräumte Struktur
- relative Pfade





Github

```
├── README.md  
└── .gitignore  
└── analysis  
└── experiment  
└── data  
    ├── raw_data  
    │   └── participant_001.csv  
    │   └── participant_002.csv  
    │   ...  
    └── preprocessed_data  
        └── all_participants_demogr.csv  
        └── all_RT_data_clean.csv  
        ...
```

= Guide für euer Projekt oder einen Unterordner

Inhalt z.B.:

- Projekttitel / Ordner-Titel
- Was tut euer Projekt? Was enthält euer Ordner?
- Welche Ressourcen braucht man, um euren Code laufen zu lassen?
- Inhaltsverzeichnis für das Projekt / den Ordner
- Credits (Wer hat euch beim Projekt geholfen?)
- Lizenz (Wer darf euren Code unter welchen Bedingungen benutzen?)

Aufgaben für Euch

Öffnet auf Github die leere README.md-Datei in eurem Repository. Beschreibt darin kurz den Inhalt des Repositorys. In der Preview könnt ihr eure Veränderungen sehen.

Was passiert, wenn ihr vor eine Zeile einen Hashtag setzt? Was ist mit 2, 3 oder 4 Hashtags hintereinander?

Was passiert, wenn ihr vor und hinter ein Wort zwei Sternchen setzt (Beispiel: **Wort**)? Was passiert bei je einem Sternchen?

Was passiert, wenn ihr ein > vor eine Zeile setzt?

Wie erstellt man Aufzählungen?

Anlegen einer Projektstruktur

```
├── README.md  
└── .gitignore  
├── analysis  
├── experiment  
├── data  
│   ├── raw_data  
│   │   └── participant_001.csv  
│   │   └── participant_002.csv  
│   ...  
└── preprocessed_data  
    └── all_participants_demogr.csv  
    └── all_RT_data_clean.csv  
    ...
```

= Welche Dateien sollen bei der Versionierung ignoriert werden?

Inhalt z.B.:

- Sensible Daten, die ihr nicht online speichern möchten
- Große Dateien, die nicht verändert werden
- Abbildungen
- Versteckte Systemdateien (z.B. .DS_Store bei Apple-PCs)
- automatisch generierte Dateien von Apps (z.B. .Rhistory-Dateien)

→ meistens 1 .gitignore-Datei in der obersten Ebene des Repos (gilt für das ganze Repo)

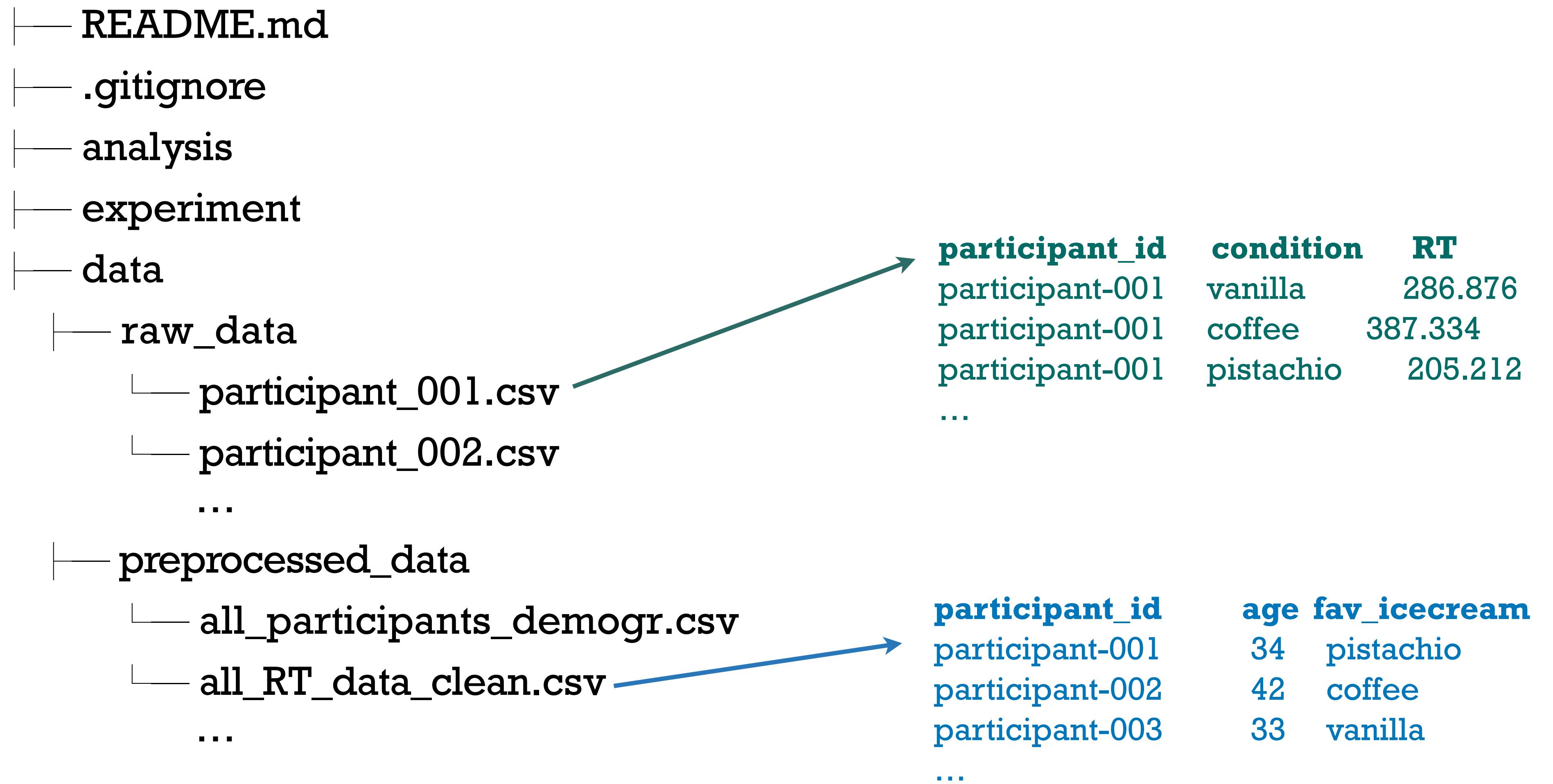
Aufgaben für Euch

Legt eine .gitignore Datei an, in der ihr alle Dateien von der Versionierung ausschließt, die .DS_STORE und .Rhistory heißen.

Hinweis 1: Erstellt die .gitignore-Datei online auf Github.

Hinweis 2: Die .gitignore-Datei hat nicht das übliche Dateibezeichnungsformat (z.B. *ignored_files.gitignore*), sondern nur das Dateikürzel *.gitignore* als Namen.

Anlegen einer Projektstruktur



Anlegen einer Projektstruktur

```
├── README.md  
├── .gitignore  
└── analysis  
    ├── Preproc.R  
    └── experiment  
        ├── data  
        │   └── raw_data  
        │       └── participant_001.csv  
        │       └── participant_002.csv  
        ...
```

Wie navigiere ich zu anderen Files?

Option 1: absolute Pfade

- Problem: Pfade sind auf jedem Gerät anders
- Beispiel: „Users/**merle/Desktop**/my_folder/my_file.R“
„Users/**ms/Dokumente**/my_folder/my_file.R“

Option 2: relative Pfade

- Navigation ausgehend vom aktuellen Working Directory
- flexibel, solange man die Ordnerstruktur nicht verändert
- Beispiel: „./my_folder/my_file.R“

Punkt ersetzt
Teil des Pfads

Anlegen einer Projektstruktur

```
└── my_project_folder
    ├── my_folder_1
    ├── my_folder_2
    └── my_folder_3
        └── my_file.R
```

Relative Pfade:

Pfad ausgehend von aktuellem working directory:

“`./my_folder_3/my_file.R`”

eine Ordner-Ebene höher anfangen:

“`../my_folder_2/my_folder_3/my_file.R`”

zwei Ordner-Ebenen höher anfangen:

“`../../my_folder_1/my_folder_2/my_folder_3/my_file.R`”

Aufgaben für Euch

Passt das Analyseskript R_Tutorial_live.R in R so an, dass...

- ... es einen sinnvolleren Namen hat.
- ... die Pfade am Anfang zu den richtigen Ordnern und Dateien führen.
- ... die neuen Datensätze am Ende beim Speichern an einem sinnvollen Ort abgelegt werden.

Hinweise:

Das aktuelle Working Directory kann man sich mit getwd() ausgeben lassen.

Installiert und ladet das package „rstudioapi“ und schreibt folgenden Befehl, um in der Variable *current_path* den Pfad zum aktuellen Dokument abzulegen:

```
current_path <- rstudioapi::getActiveDocumentContext()$path
```

Mit setwd(“*Pfad*“) kann man ein Working Directory für das Skript setzen (das ist dann der Ausgangspfad für alle anderen relativen Pfade danach).

Ende des Workshops!

Ihr wisst jetzt...

- **wie man eine Onlinestudie in lab.js baut.**
- **wie man über Open Lab und OSF online Daten erheben kann.**
- **wie man R nutzen kann um Daten zu analysieren.**
- **wie man Versionierung nutzen kann, um alte Versionen von Skripten zu retten, nicht in Bergen von Files zu versinken und mit anderen an Projekten zusammen zu arbeiten.**

Materialien: <https://github.com/BioPsychKiel/SpringSchool2023/tree/main/Track1>