



GetTogetherR!

Git und R

Jonas Frost

jonas.frost@studserv.uni-leipzig.de

Peter Kannewitz

peter.kannewitz@uni-leipzig.de

13. April 2023

- monatliche Veranstaltung (jeden 2. Donnerstag im Monat)
- kurze Inputvorträge von uns oder unseren Gästen
- aktuelle Themen rund um R und RStudio:
 - Git und R (13.04.2023)
 - Netzwerkanalyse in R [Till Hovestadt] (11.05.2023)
 - ChatGPT als R-Copilot (08.06.2023)
 - Hausarbeiten mit RMarkdown (13.07.2023)
- aktueller Syllabus im [Moodlekurs](#)

1. Kurzeinstieg: Git
2. Setup mit RStudio
3. Ressourcen und Hilfe
4. Austausch

- Git entwickelt sich zum Standard für viele Anwendungsfelder
 - Kollaboration
 - Sicherung von Projekten (Backup)
 - Dokumentation
- RStudio bietet gute und einfache Möglichkeit zur Versionskontrolle via Git
- Funktion ist oft nicht Teil von Einführungskursen und wird daher wenig genutzt

Kurzeinstieg: Git



- Programm zur Versionskontrolle



- Programm zur Versionskontrolle
- kann komplett lokal genutzt werden



- Programm zur Versionskontrolle
- kann komplett lokal genutzt werden
- kommt ohne grafische Oberfläche



- Programm zur Versionskontrolle
- kann komplett lokal genutzt werden
- kommt ohne grafische Oberfläche
 - mögliche GUIs: [GitHub Desktop](#), [Source Tree](#), [GitKraken](#), bzw. RStudio



- Programm zur Versionskontrolle
- kann komplett lokal genutzt werden
- kommt ohne grafische Oberfläche
 - mögliche GUIs: [GitHub Desktop](#), [Source Tree](#), [GitKraken](#), bzw. RStudio
- Änderungen werden Schrittweise durchgeführt (`git commit`)



- Programm zur Versionskontrolle
- kann komplett lokal genutzt werden
- kommt ohne grafische Oberfläche
 - mögliche GUIs: [GitHub Desktop](#), [Source Tree](#), [GitKraken](#), bzw. RStudio
- Änderungen werden Schrittweise durchgeführt (`git commit`)
- Änderungen können Rückgängig gemacht werden (`git restore`)



- Programm zur Versionskontrolle
- kann komplett lokal genutzt werden
- kommt ohne grafische Oberfläche
 - mögliche GUIs: [GitHub Desktop](#), [Source Tree](#), [GitKraken](#), bzw. RStudio
- Änderungen werden Schrittweise durchgeführt (`git commit`)
- Änderungen können Rückgängig gemacht werden (`git restore`)
- Ältere Versionen des Projekts können wieder hergestellt werden (`git reset`)



- Programm zur Versionskontrolle
- kann komplett lokal genutzt werden
- kommt ohne grafische Oberfläche
 - mögliche GUIs: [GitHub Desktop](#), [Source Tree](#), [GitKraken](#), bzw. RStudio
- Änderungen werden Schrittweise durchgeführt (`git commit`)
- Änderungen können Rückgängig gemacht werden (`git restore`)
- Ältere Versionen des Projekts können wieder hergestellt werden (`git reset`)
- Verschiedene Versionen des Projekts können parallel bearbeitet werden und später zusammengeführt (Branching -> `git checkout`, Merging -> `git merge`)

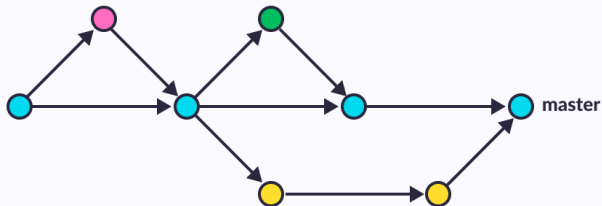


Figure 1: Inkrementale Arbeit an einem Projekt

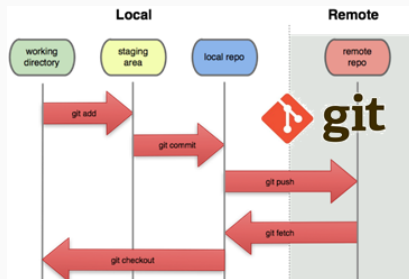


Figure 2: Zusammenarbeit über ein Remote Repository

- Git-Hoster: [GitHub](#), [GitLab](#), [Gitea](#), ...

- Beispiel: <https://github.com/tidyverse>
- **Repository** → Verzeichnis mit allen Projektdateien und Git-Verwaltungsdateien (Kann öffentlich oder privat sein)
- **Branches** → Abzweigungen vom Hauptentwicklungsstand, z.B. für mehrere Änderungen oder Änderungsvorschläge
- **Commit** → Eine Änderung innerhalb einer Branch
- **README.md** → Projektbeschreibung und Informationen
- **History** → Änderungsverlauf einer Datei
- **Issues** → Anregungen, Änderungsvorschläge, Kritik, etc.
- **Pull requests** → Externe Änderungsvorschlag, der in eine Branch übernommen werden soll
- **Clone** → Aktuellen Stand als *lokale* Kopie auf dem eigenen Rechner anlegen

1. Remote Repository anlegen.
 2. Lokal klonen.
 3. Änderungen durchführen.
 4. Änderungen committen (Zum aktuellen Entwicklungsstand hinzufügen).
 5. Änderungen pushen (Ins Remote-Repository übertragen).
- *Wichtig:* Immer aktuellen Stand pullen, bevor man eigene Änderungen vornimmt

Setup mit RStudio

- Was braucht man alles?
 - Git auf dem Rechner installiert
 - Account bei einem Git-Hoster (bspw. GitHub)
 - bestehendes Git-Repository
- Schritte:
 - neues R-Projekt erstellen
 - Version Control
 - URL des Repositorys angeben



- GitHub ist nicht für große Daten gedacht (Filelimit 100MB)
- man muss Änderungen regelmäßig pushen, sonst hat man nichts gewonnen
- bei Problemen (gerade beim Kollaborieren) muss man sich näher mit Git auseinandersetzen (merge conflicts)

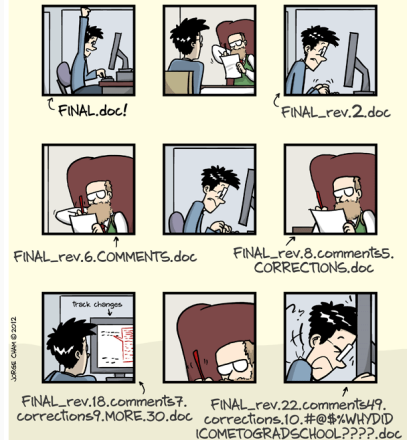
Warum sich die Mühe machen?

- Änderungen tracken, dokumentieren und rückgängig machen
- gleichzeitig im Team zusammenarbeiten und Änderungen zusammenführen
- Projekte sicher abspeichern
- GitHub bietet unglaublich viel Potential für weitere Anwendungsfälle (Aufgabenplanung in Projekten, Releases, Continuous deployment von [Bookdown](#) Dokumenten, ...)

Piled Higher and Deeper by Jorge Cham

www.phdcomics.com

"FINAL".doc



WWW.PHDCOMICS.COM

title: "notFinal.doc" - originally published 10/12/2012

Ressourcen und Hilfe

- <https://r-bio.github.io/intro-git-rstudio/> (Einstiegs Tutorial)
- <https://posit.co/resources/videos/managing-part-2-github-and-rstudio/> (Video zum Einstieg)
- <https://happygitwithr.com/> (Umfassende Referenz)

Austausch

- Welche Erfahrung habt Ihr mit Git und RStudio? Tipps? Tricks?
- Habt Ihr Ideen für eigene Anwendungsfälle? Wofür wollt Ihr die RStudio-Integration von Git benutzen?
- Öffentliche Git-Repos als gute wissenschaftliche Praxis, um Reproduzierbarkeit zu gewährleisten?

- Ausprobieren?

Danke fürs Teilnehmen!

- https://geniusee.com/storage/app/media/blog/blog223_git_branching_model/GitHub_Flow.png
- <https://git-scm.com/images/logo@2x.png>
- <https://www.inflectra.com/Images/Product-Imagery/Git.png>
- https://www.freepik.com/free-icon/business-presentation_772083.htm
- https://phdcomics.com/comics/archive_print.php?comid=1531