
Empirisch-experimentelle Forschungsmethoden in der Anwendung

Seminar



Termine

Getting Started	
12.10.	Einführung, ULN, IT-Starthilfe
19.10.	Forschungsfrage und Faktorenraum
26.10.	Projektorganisation und Syntax
02.11.	(Brückentag): Fragebogen als Video

Projekt Teil 1: Arbeitsphase	
09.11.	Data Cleaning, Reliabilität, Sample-Size Estimation
16.11.	Deskriptive Statistik
23.11.	Boxplot, Histogramm
30.11.	T-Test + Plots
07.12.	Anova/Manova + Plots
14.12.	Korrelationen

Projekt Teil 2: Aufarbeitung und Vortrag	
11.01.	Likert Plots
18.01.	Hilfe-Stunde
25.01.	Vortrag 1
01.02.	Vortrag 2

Themen heute

- Pitch zur Hausaufgabe
- Projektorganisation mit GIT
- Live-Demo: GitHub und R Studio
- Struktur des R-Projektes
- Schreiben mit Markdown

Pitch zur Hausaufgabe

- Inhalt der Literatur?
 - Forschungsfrage?
 - Faktorenraum?
-
- max. 5 Minuten pro Gruppe

Projektorganisation mit git

- Software zur Versionsverwaltung verschiedener Daten, i.d.R. Quellcode
- Warum überhaupt Versionskontrolle?
 - Protokollierung von Änderungen
 - Backup

→ Mächtiger Schutz vor menschgemachten Fehlern
- Warum speziell git?
 - Sehr populär, dezentral, geeignet zum kollaborativen Arbeiten, Dokumentationszwang



Lokale / Zentrale / Dezentrale Versionskontrolle

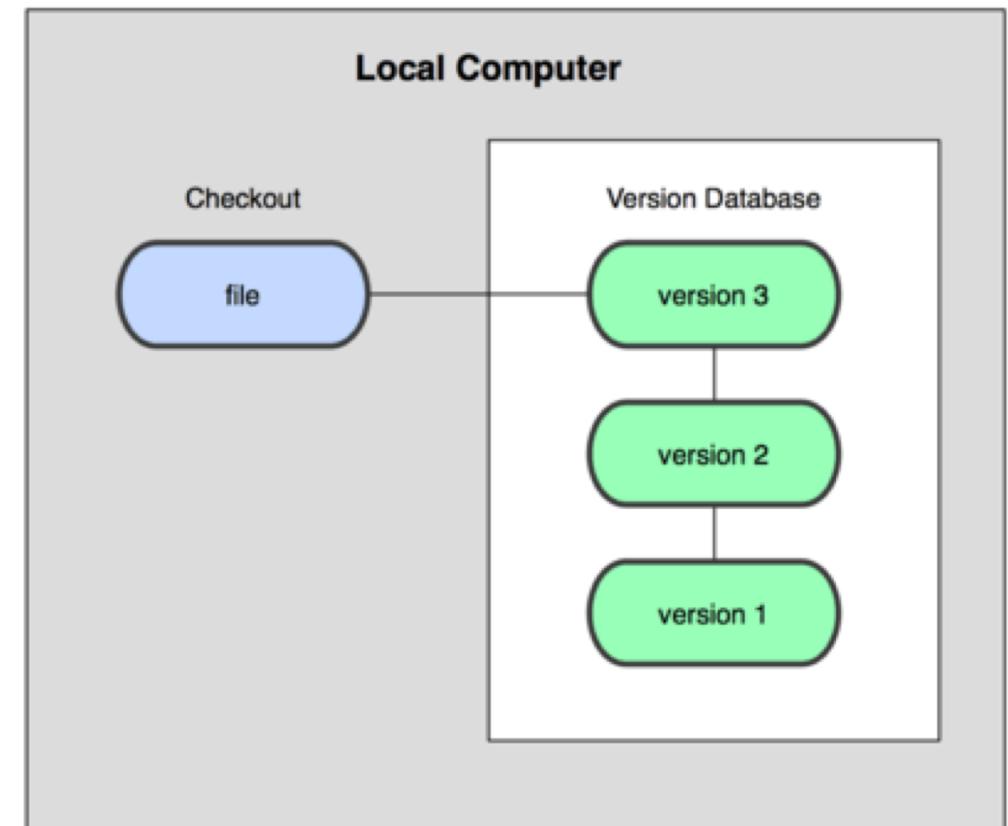
Lokal:

- In regelmäßigen Abständen wird eine Datei / ein Ordner

Projektarbeit_Stand261118

erstellt.

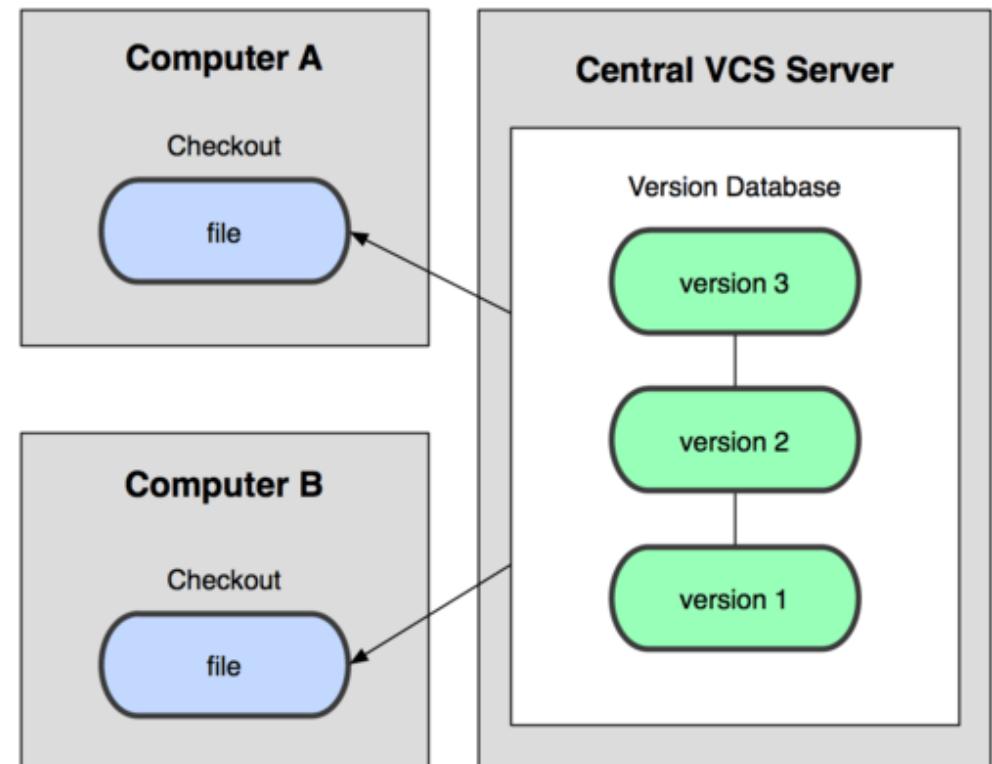
- Vorteile / Nachteile?



Lokale / Zentrale / Dezentrale Versionskontrolle

Zentral:

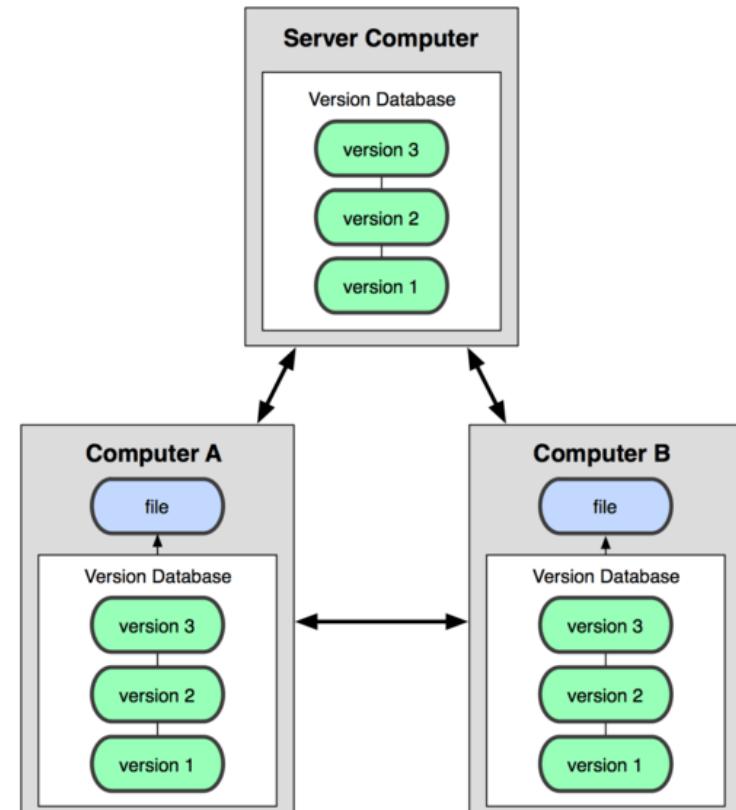
- Die Daten liegen alle auf einem Server
- Abholung (= **Checkouts**) durch Clients
- Bearbeitung
- Ablieferung
- Vorteile / Nachteile?



Lokale / Zentrale / Dezentrale Versionskontrolle

Dezentral:

- Anwender und Server erhalten eine komplette (!) Kopie aller bisherigen Versionen (= **Repository**)
- Vorteile?
 - Geschwindigkeit
 - Ausfallsicherheit
- Nachteile?



git: 3 Arbeitsbereiche, 3 Zustände...

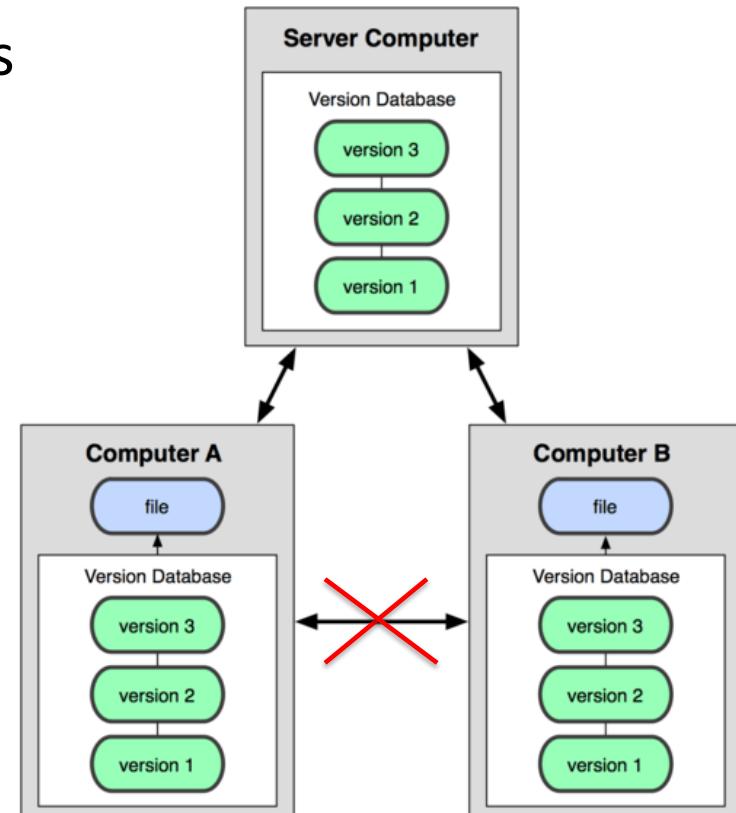
Keine direkte Kommunikation zwischen Clients

3 Arbeitsbereiche:

- *Working Directory*
- *Staging Area*
- *Repository*

3 Zustände einer Datei:

- *Committed*
- *Modified*
- *Staged*



git: 3 Arbeitsbereiche, 3 Zustände...

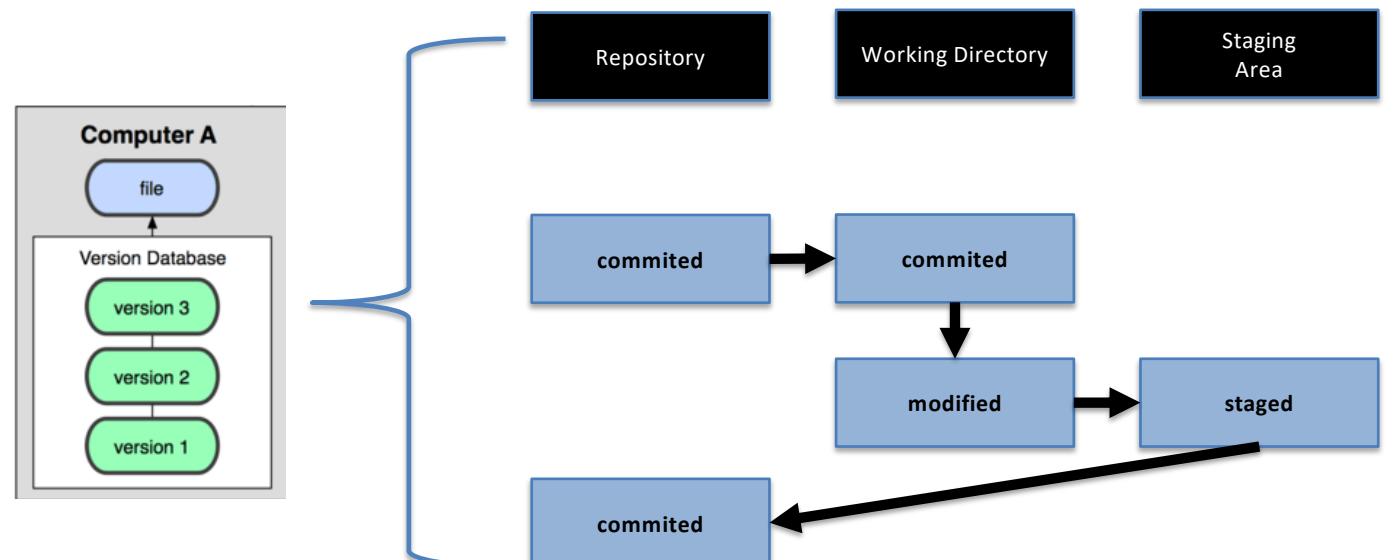
Keine Kommunikation direkt zwischen Clients

3 Arbeitsbereiche:

- **Working Directory**
- **Staging Area**
- **Repository**

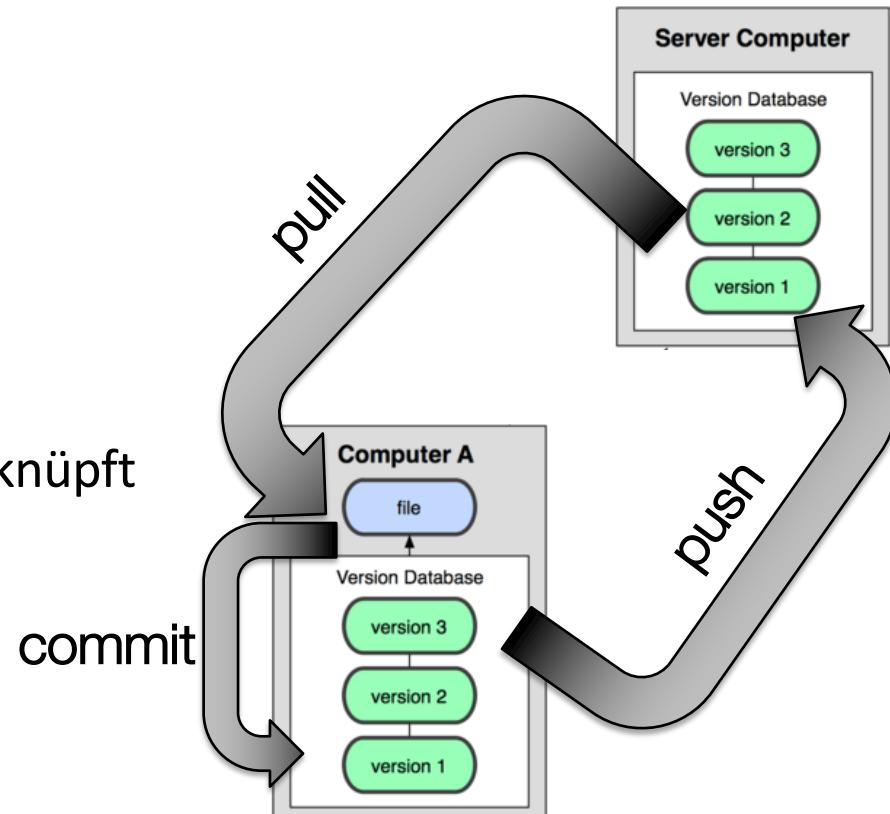
3 Zustände einer Datei:

- **Committed**
- **Modified**
- **Staged**



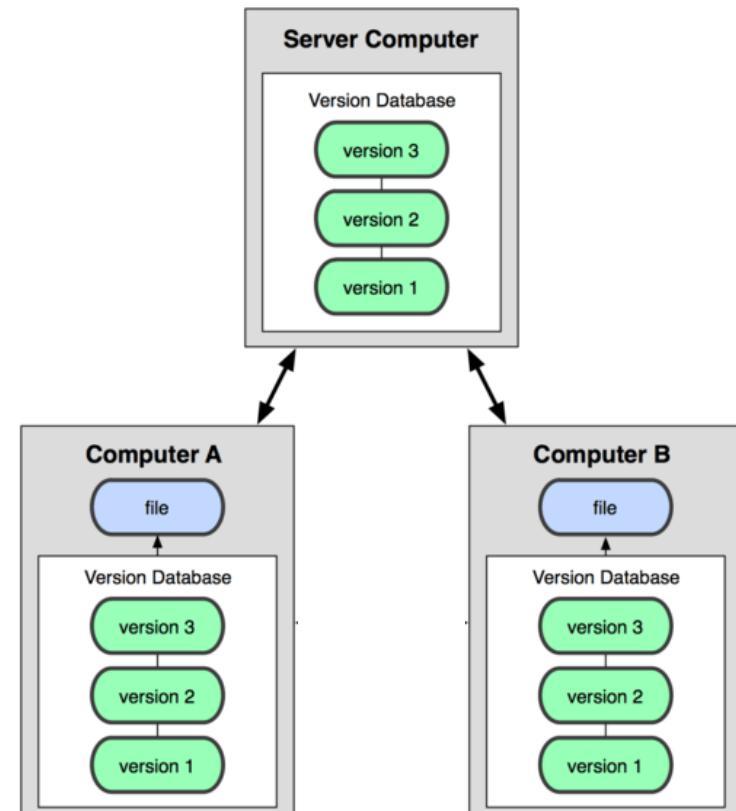
git: ...3 wichtige Aktionen

- Pull
 - Abgleich von Server → Client
- Commit
 - Änderung am lokalen (!) repository
 - Wird mit einer commit-message verknüpft
- Push
 - Abgleich Client → Server
 - Kann zu Konflikt führen



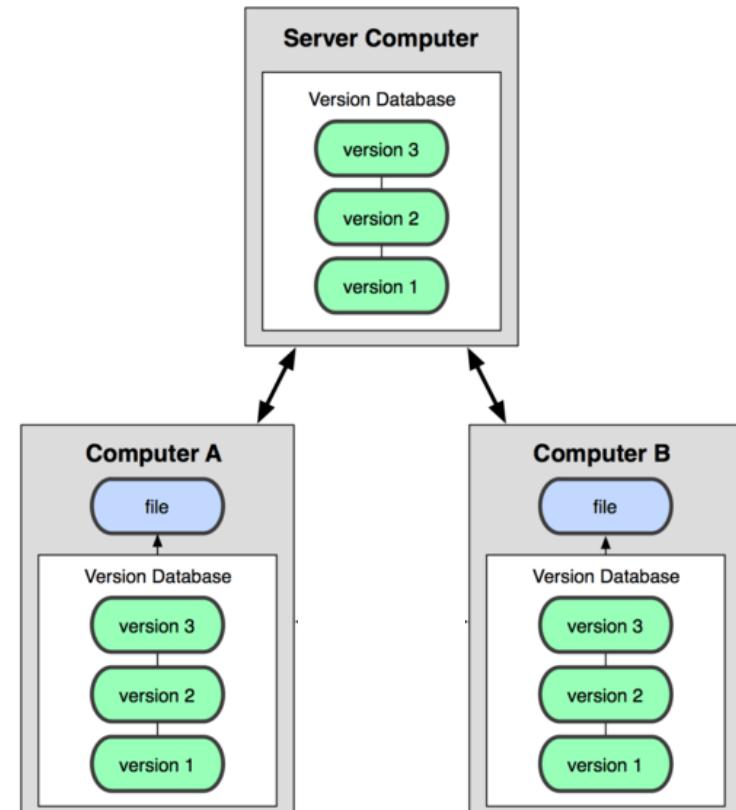
Kompletter Lifecycle ohne Konflikt

1. pull
 2. Modifikation einer Datei
 3. Markierung der Datei als staged
 4. commit (mit aussagekräftiger message)
 5. push
- Wie kann ein Konflikt entstehen?



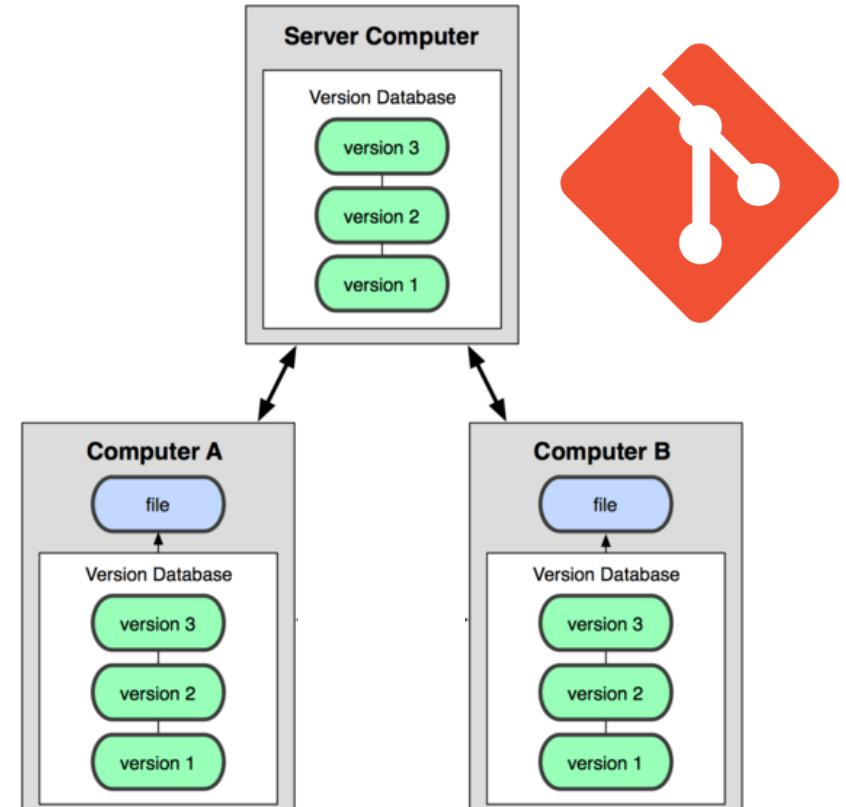
Kompletter Lifecycle mit Konflikt

1. pull
 2. Modifikation einer Datei
 3. Markierung der Datei als staged
 4. commit (mit aussagekräftiger message)
 5. push führt zu Konflikt 
1. pull
 2. Auflösung des Konfliktes (*merge*)
 3. commit der Konfliktlösung
 4. push



Alles klar?

- pull, commit, push, merge
- committed, modified, staged
- working directory, staging area, repository
- Es gibt noch mehr Befehle:
 - init: Erstellt ein repository
 - clone: Legt eine lokale Kopie eines repository an
 - add: Fügt eine Datei zum repository hinzu
- git kann noch viel mehr: branches



Branches

- Nicht notwendig für ihr Projekt!
- „Zweige“ zur verbesserten Koordination großer Projekte

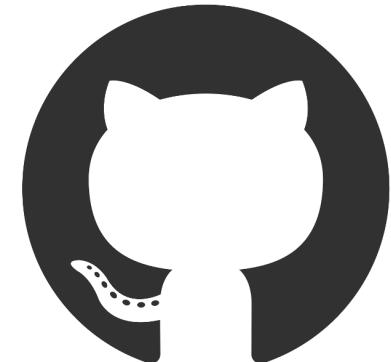


Github

- Onlinedienst zum filehosting von git repositories
- Öffentliche repositories sind kostenlos

Jetzt:

- Installation von git: <https://git-scm.com/downloads>
- Kostenloser Account auf github: <https://github.com/>
- Live Demo: Git und R Studio



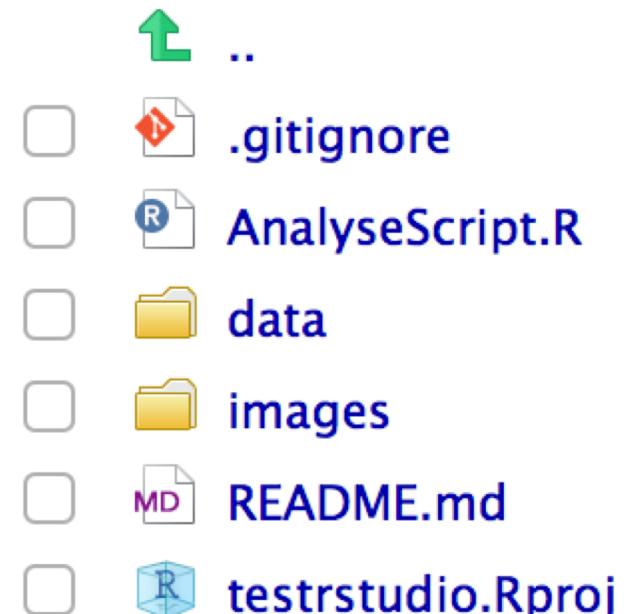
Projektorganisation und Syntax

- Regeln für git:

- Das repository soll immer funktionsfähig sein!
- Aussagekräftige commit-messages
 - Erste Zeile im Imperativ, danach Leerzeile
 - Was wurde warum geändert?
 - Kurz und prägnant

- .gitignore nutzen

- Nicht alle Dateien im repository sind notwendig
- Die erhobenen Daten sollten z.B. nicht an die Öffentlichkeit!



AnalyseScript.R

- Kommentare mit #
- Abschnitte als Kommentar mit ---- am Ende
- Erste Zeile ist der Dateiname als Kommentar
- Pakete mit library() laden
- Abschnitte:
 - Data Cleaning, Skalenberechnung (09.11.18)
 - Analyse und Grafiken (ab 16.11.18)

```
# Analyse Script

library(haven)
library(tidyverse)

# Data Cleaning ----

# Skalenberechnung ----

# Analyse ----

# * Analyse 1 ----
# Wir prüfen ob sich die Variable X1 von 0
Unterscheidet.
print("Hypothese 1")
t.test(anscombe$x1)

# Grafiken ----
```

readme.md als Markdown

Überschrift

Diese Beschreibung ist in Markdown verfasst. Der Rest des Projektes ist in der _Progammiersprache_ `r` erstellt.

Unterüberschrift

Man kann sogar mit Windows arbeiten. Die Installation von Git ging ganz einfach, man muss nur sehr oft auf `ok` klicken.

Unterunterüberschrift

* Liste 1

* Liste 2

1. Liste 1

2. Liste 2

![tooltip](githubfiles/Faktorenraum.png)

| Hallo | Test |

|-----|-----|

| Hallo | Test |

Überschrift

Diese Beschreibung ist in Markdown verfasst. Der Rest des Projektes ist in der *Progammiersprache r* erstellt.

Unterüberschrift

Man kann sogar mit Windows arbeiten. Die Installation von Git ging ganz einfach, man muss nur sehr oft auf ok klicken.

Unterunterüberschrift

- Liste 1

- Liste 2

1. Liste 1

2. Liste 2

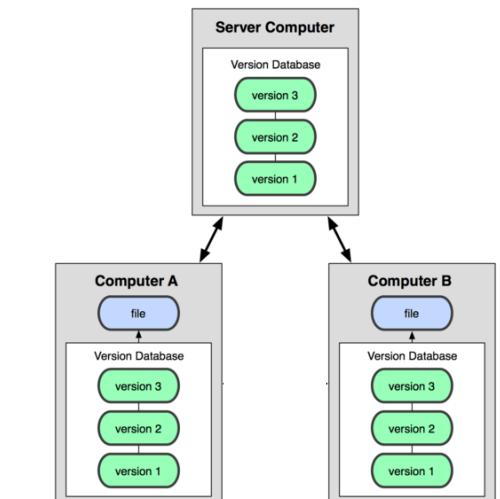


Hallo	Test
Hallo	Test



Zusammenfassung

- Git: pull, commit, push, merge, etc.
- GitHub: Repository anlegen und verwalten, Readme.MD
- RStudio: Best practice, Projektorganisation
- Markdown: Syntax



Der nächste Termin

- Nächste Woche: 02.11.2018 Fragebogen (als Video)
- Hausaufgabe in Kleingruppe:
 - siehe L2P und Slack
- Hausaufgabe individuell:
 - Datacamp Übung zu Matrizen und Faktoren

