24.11.20

Einführung in Data Science und maschinelles Lernen mit R

Versionierung mit git (Teil 1) und Datenaufbereitung mit Tidyverse

- **Besprechung Übungsaufgaben**
- **Einführung in die Versionierung mit git**
- **Zusammenführung von Dateien**
- **Einführung in Tidyverse und die Datenaufbereitung**

AUFGABEN

- **Erstelle ein Balkendiagramm, dass über alle Warengruppen hinweg die durchschnittlichen Umsätze je Wochentag zeigt.**

- **Füge in einem zweiten Schritt zusätzlich Konfidenzintervalle der Umsätze je Wochentag hinzu („barplot with error bars“).**

- ***Freiwillige Zusatzaufgabe:***
  Stelle die Umsätze je Wochentag getrennt nach Warengruppe dar (ein eigenes Balkendiagramm je Warengruppe)

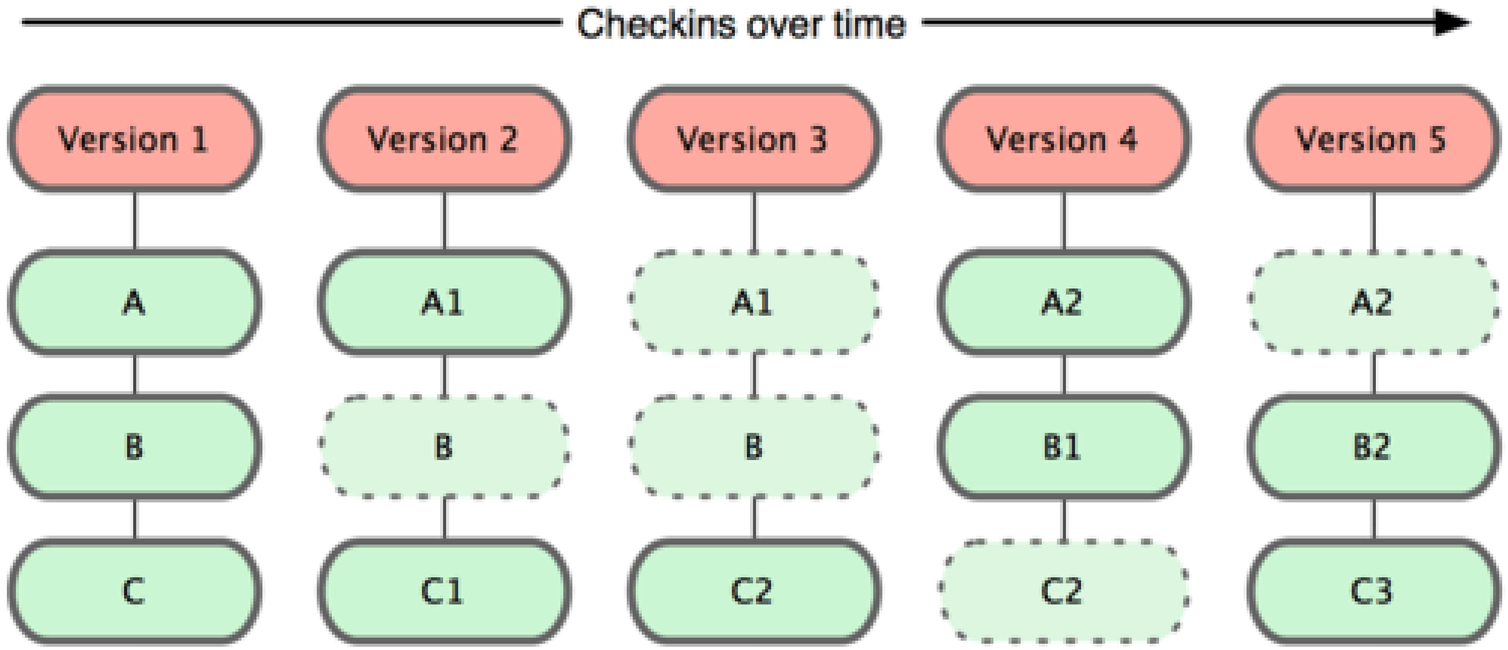
ÜBUNGSAUFGABEN

Arbeitsgruppen

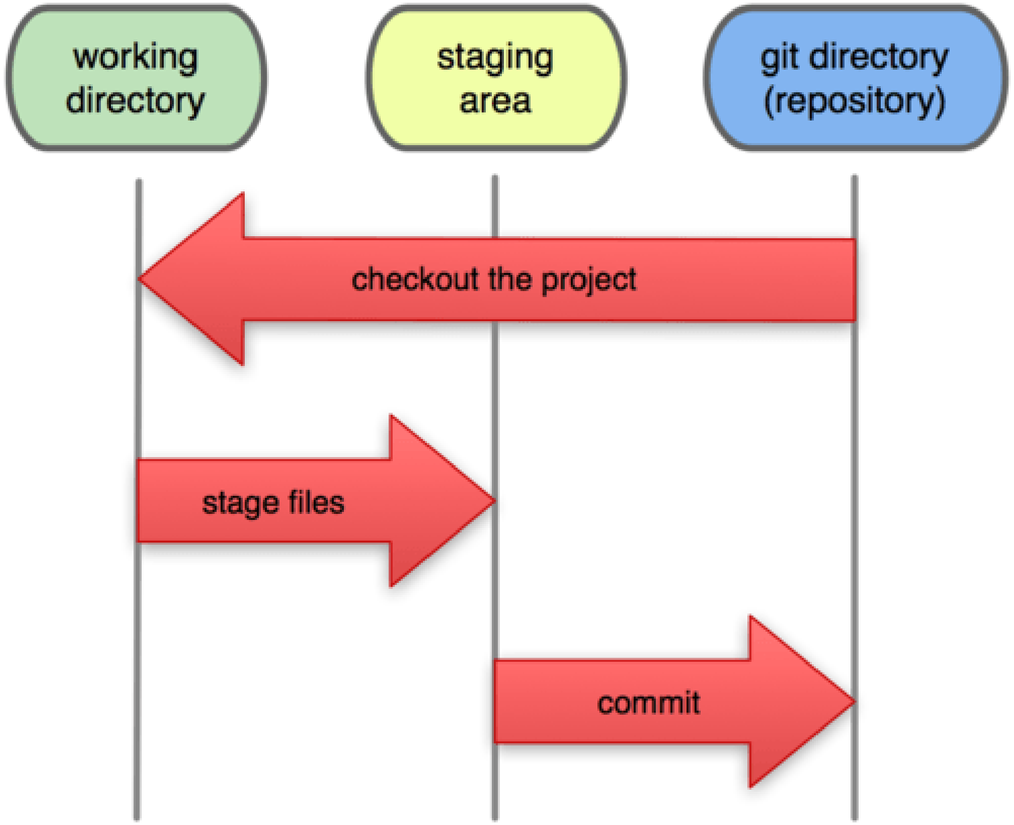
10 Minuten

VERSIONIERUNG MIT GIT

- **Alle Versionen werden in einem lokalen „Repository“ abgelegt.**
- **Jede neue Version enthält immer alle Dateien des Projektes.**

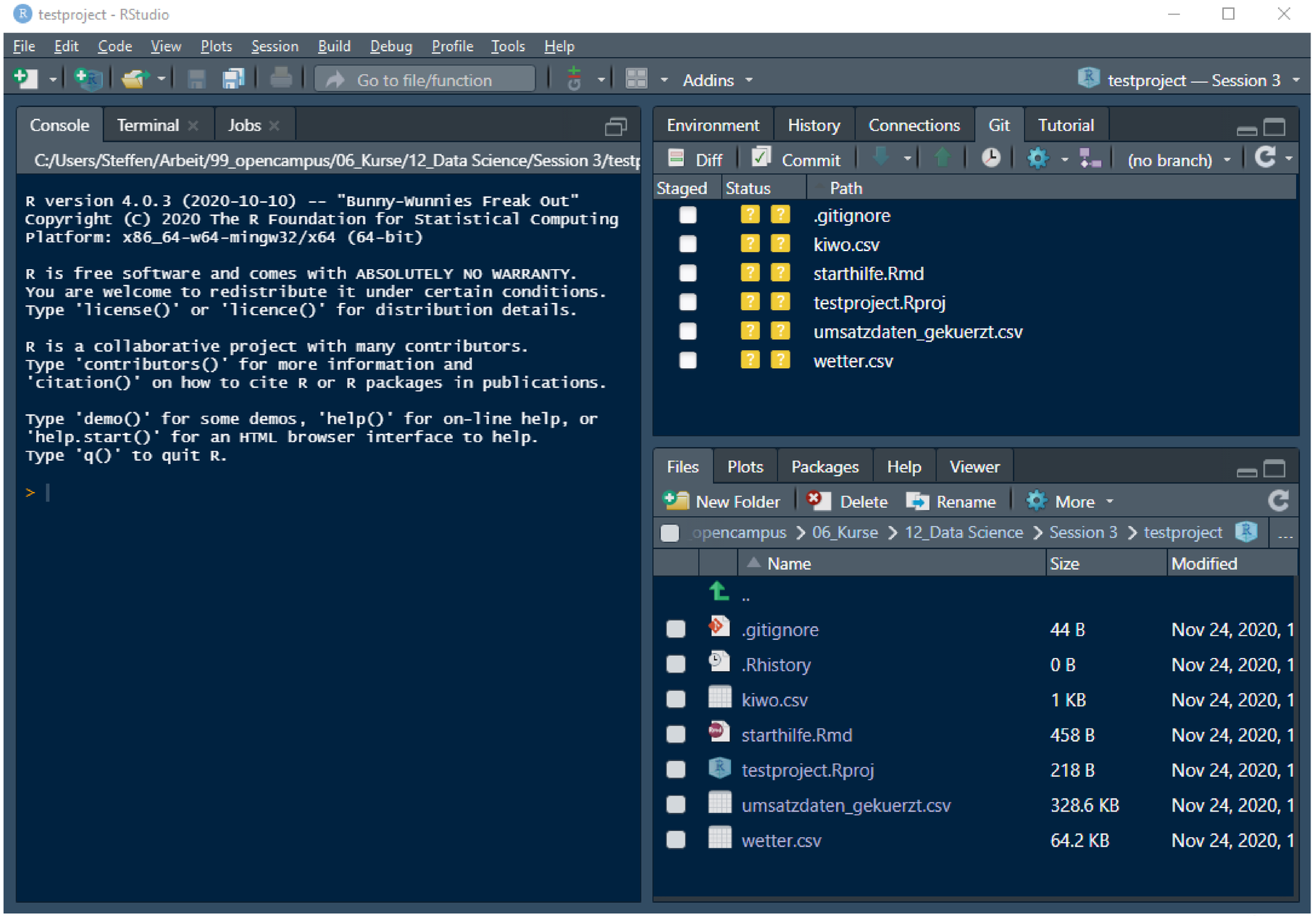


VERSIONIERUNG MIT GIT



Eine Datei kann drei mögliche Zustände haben:

- **modified („geändert“)**
- **staged („vorgemerkt“) und**
- **committed („versioniert“).**



KONFIGURATION VON GIT

Vor der erstmaligen Verwendung von Git, muss einmalig definiert werden, in wessen Namen die Repositories des installierten Git verwaltet werden.

Geht dazu bitte in das Terminal-Fenster (unten links in RStudio) und gebt Euren GitHub Benutzernamen und Eure Email-Adresse an:

```
git config --global user.name "your_username“
git config --global user.email your_email@example.com
```

Der Benutzername kann prinzipiell beliebig sein, da wir aber später GitHub benutzen werden, sollte dieser dann mit dem Benutzernamen bei GitHub übereinstimmen, sobald ihr dort einen Nutzer habt.

AUFGABEN

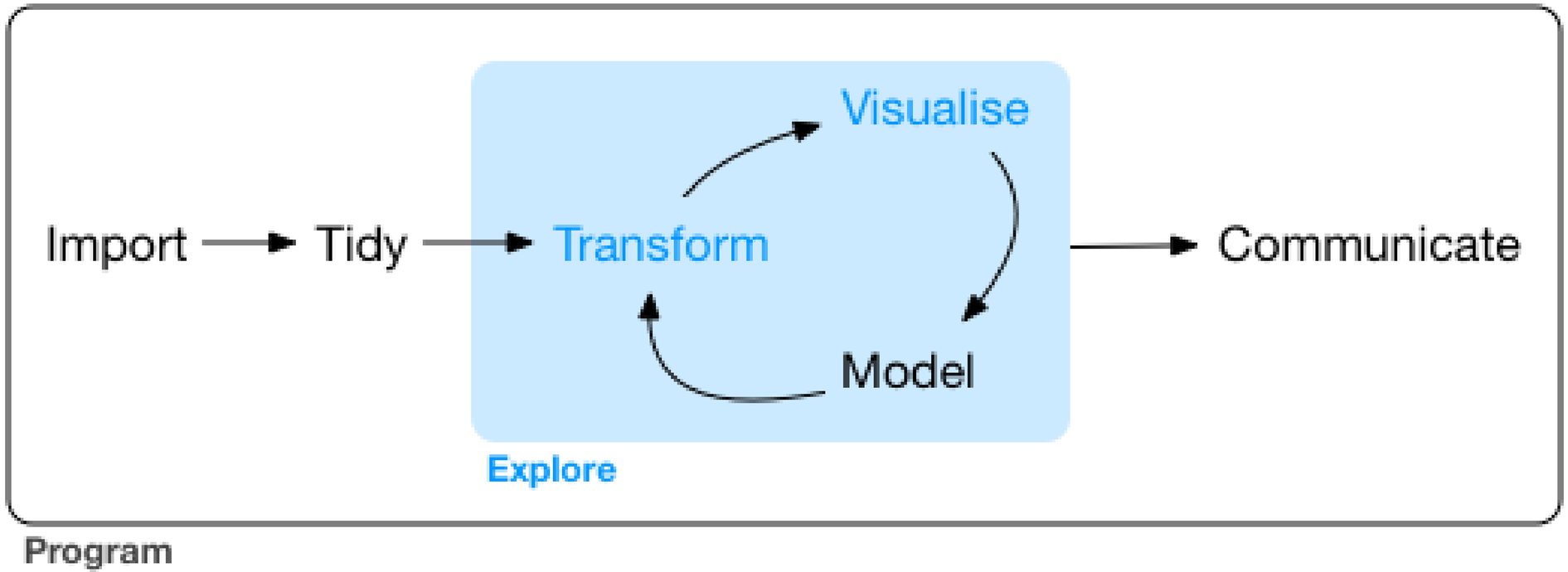
1) **Wählt *git* als Versionierungsanwendung für Euer Projektverzeichnis aus.**
2) **„Staged“ alle Dateien (markiert sie für das nächste Commit), die Ihr versionieren wollt und „committed“ sie dann.**
3) **Führt ein erstes „Commit“ aus, um eine erste Projektversion mit allen bisherigen Dateien anzulegen.**
4) **Legt ein neues R-Notebook im Projektverzeichnis an und legt eine neue Version einschließlich des Notebooks an.**
5) **Schaut Euch die History Eures Repositories an.**

VERSIONIERUNG MIT GIT

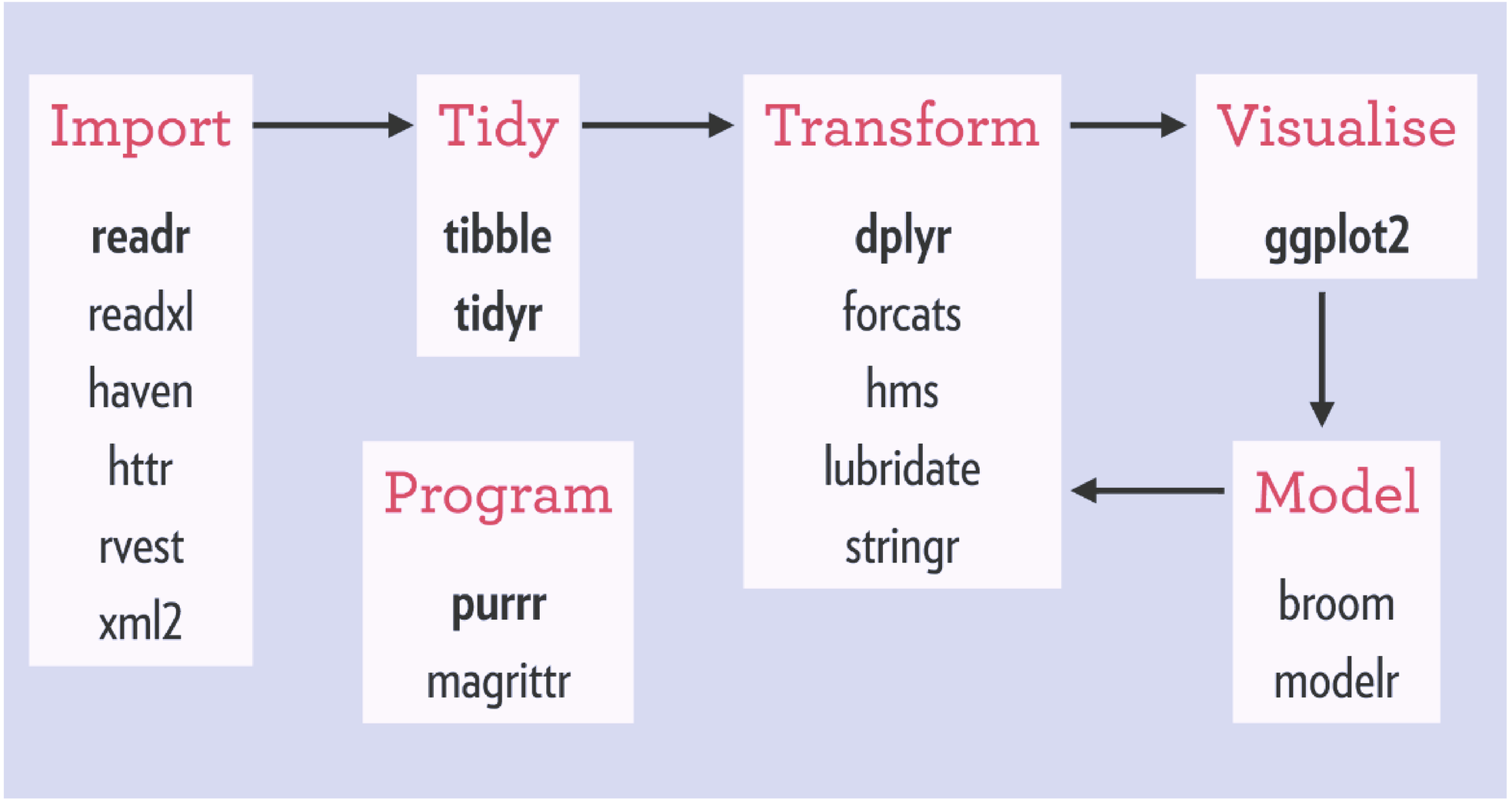
Teil 1: **Lokale Versionierung**

Teil 2: **Synchronisation der lokalen Versionierung mit einer Remote-Versionierung und Arbeiten im Team**

DATENAUFBEREITUNG



TIDYVERSE



Mehr Info: https://rviews.rstudio.com/2017/06/08/what-is-the-tidyverse/

```
mpg %>%
  group_by(cyl) %>%
  summarise(n(), t.test(cty,hwy)$p.value)
```

Pipe Operator: %>%

- Schrittweise Datenaufbereitung
- Vermeidung von Hilfsvariablen
- Erhöhung der Lesbarkeit des Programmcodes

Gruppierung von Daten: group_by()

- Vermeiden von Hilfsvariablen
- Deutliche Verkürzung des Programm-Codes
- Erhöht die Lesbarkeit des Programm-Codes

ZUSAMMENFÜHREN VON DATENTABELLEN

`left_join(x, y)`

return all rows from x, and all columns from x and y. Rows in x with no match in y will have NA values in the new columns. If there are multiple matches between x and y, all combinations of the matches are returned.

`inner_join(x, y)`

return all rows from x where there are matching values in y, and all columns from x and y. If there are multiple matches between x and y, all combination of the matches are returned.

`right_join(), full_join()`

```
daten <- left_join(umsatzdaten, kiwo)
```

DPLYR

Variablen (Spalten) auswählen: `select()`

Fälle (Zeilen) auswählen: `filter()`

Variablen hinzufügen: `mutate()`

```
mpg %>%
  select (class, hwy, cty) %>%
  filter (class=="suv") %>%
  mutate (mix = .5*hwy + .5*cty)
```

LUBRIDATE

Umwandlung von Strings in ein Datumsformat

- Zum Beispiel: `dmy()` oder `ymd()`
- Erkennt automatisch unterschiedlich Formatierungen

Umwandlung von Datumformaten in kategorische Variablen

- Zum Beispiel: `mday()` oder `wday()`
- Erkennt automatisch unterschiedlich Formatierungen

```
mdy("4/1/17")

economics %>%
  mutate(weekday=wday(date))
```

STRINGR

Zeichenersetzung: `str_replace()`

- Erlaubt die Verwendung von „regular expressions“

Führende und nachstehende Leerzeichen entfernen: `str_trim()`

- Zahlreiche ähnliche „Wrapper-Funktionen“ von `str_replace()`

```
str_replace("AAA", "A", "B")
str_replace("AAA", "A$", "B")

str_trim("   Vorname   ")
str_replace("   Vorname   ", "^\\s+ || \\s+$", "")
```

PACKAGES MIT ÜBERBLICKSFUNKTIONEN

Skimr

Gibt anhand verschiedener Statistiken einen schnellen Überblick zu den Variablen in einer Datentabelle. Je nach Inhalt der Variablen sind die einzelnen Statistiken ausagekräftig.

DataExplorer

Enthält verschiedene Funktionen für grafische Darstellungen zu allen Variablen in einer Datentabelle, etwa mit Hilfe von Histogrammen.

AUFGABEN

- Die grundlegende Funktionsweise von git kann man ggf. sehr gut in Kapitel 1.3 dieses Buches hier noch einmal nachlesen.
- Übt das Durchführen eines Commit in RStudio und macht Euch noch einmal die einzelnen Schritte klar und wozu diese benötigt werden.
- Legt einen Account bei GitHub für Euch an und notiert Euch Euren User-Namen (Ihr braucht ihn in der Session nächste Woche).
- Für eine genauere Einführung in die Möglichkeiten von Regular Expressions, schaut Euch bitte dieses Video (11 Minuten) an.