12.11.19

Einführung in Data Science und maschinelles Lernen mit R

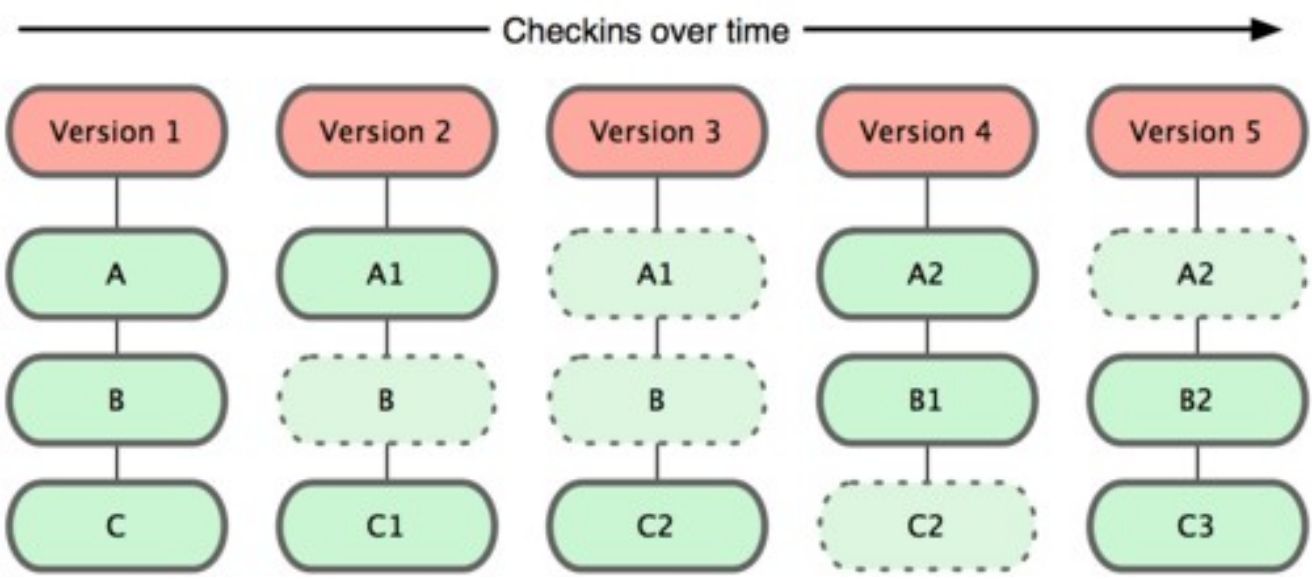
Einführung in die Versionierung mit git und die Datenaufbereitung

OPENCAMPUS.sh

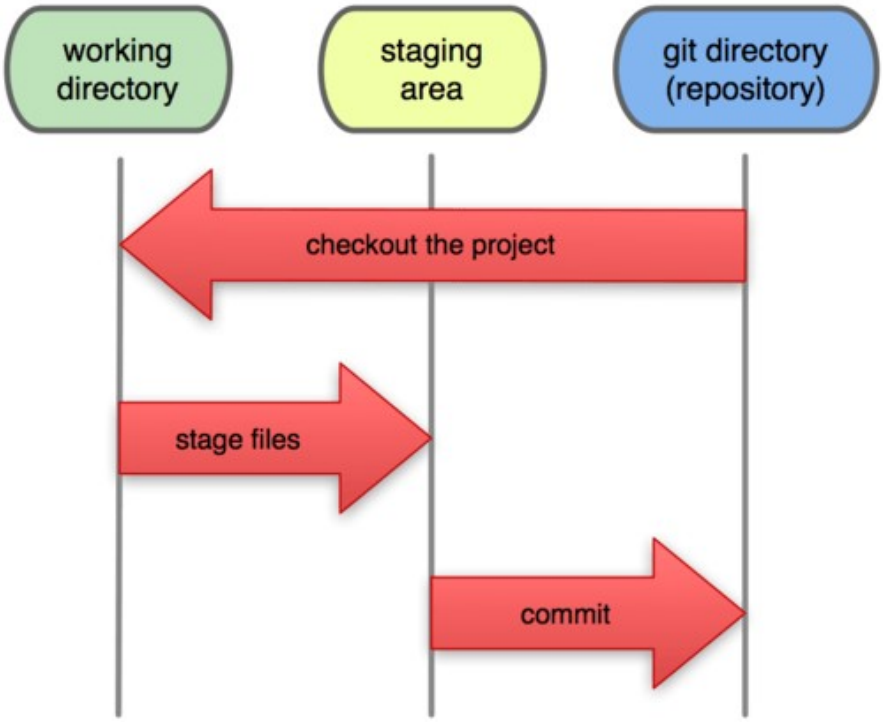
- **Besprechung Übungsaufgaben**
- **Einführung in git**
- **Zusammenführung von Dateien**
- **Einführung in Tidyverse und die Datenaufbereitung**

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 möglich Zustände haben:

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

KONFIGURATION VON GIT

Vor der erstmaligen Verwendung von Git, muss einmalig definiert werden, in wessen Namen die Repositories des installierten Git erzeigt 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, solltet ihr den entsprechenden Benutzername jetzt bereits verwenden.

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) Ladet die geänderten Umsatzdaten von *github* (https://github.com/opencampus-sh/ws1920-datascience) herunter und legt eine neue Version mit diesen geänderten Daten an.
5) Schaut Euch die History Eures Repositories an.

Online-Kurs

https://www.datacamp.com/courses/introduction-to-git-for-data-science

Schriftliche Einführung zum Nachlesen

https://git-scm.com/book/de/v2

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.

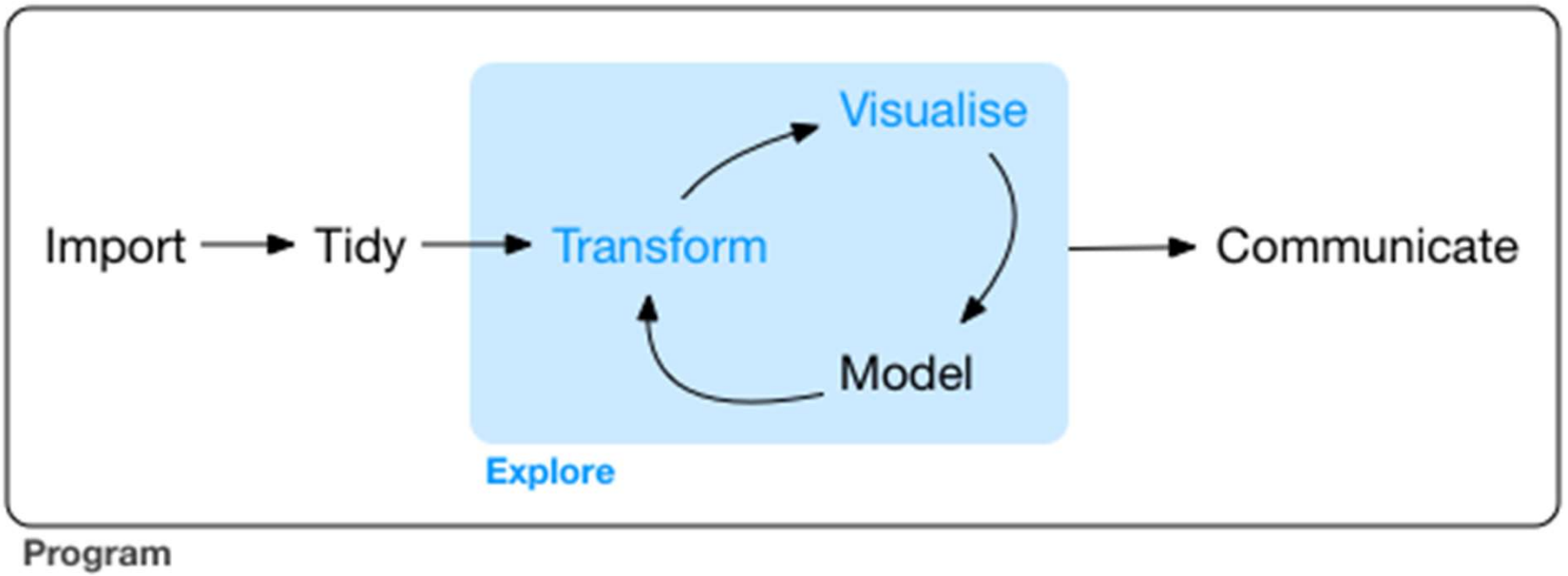
`inner_join(x, y)`

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.

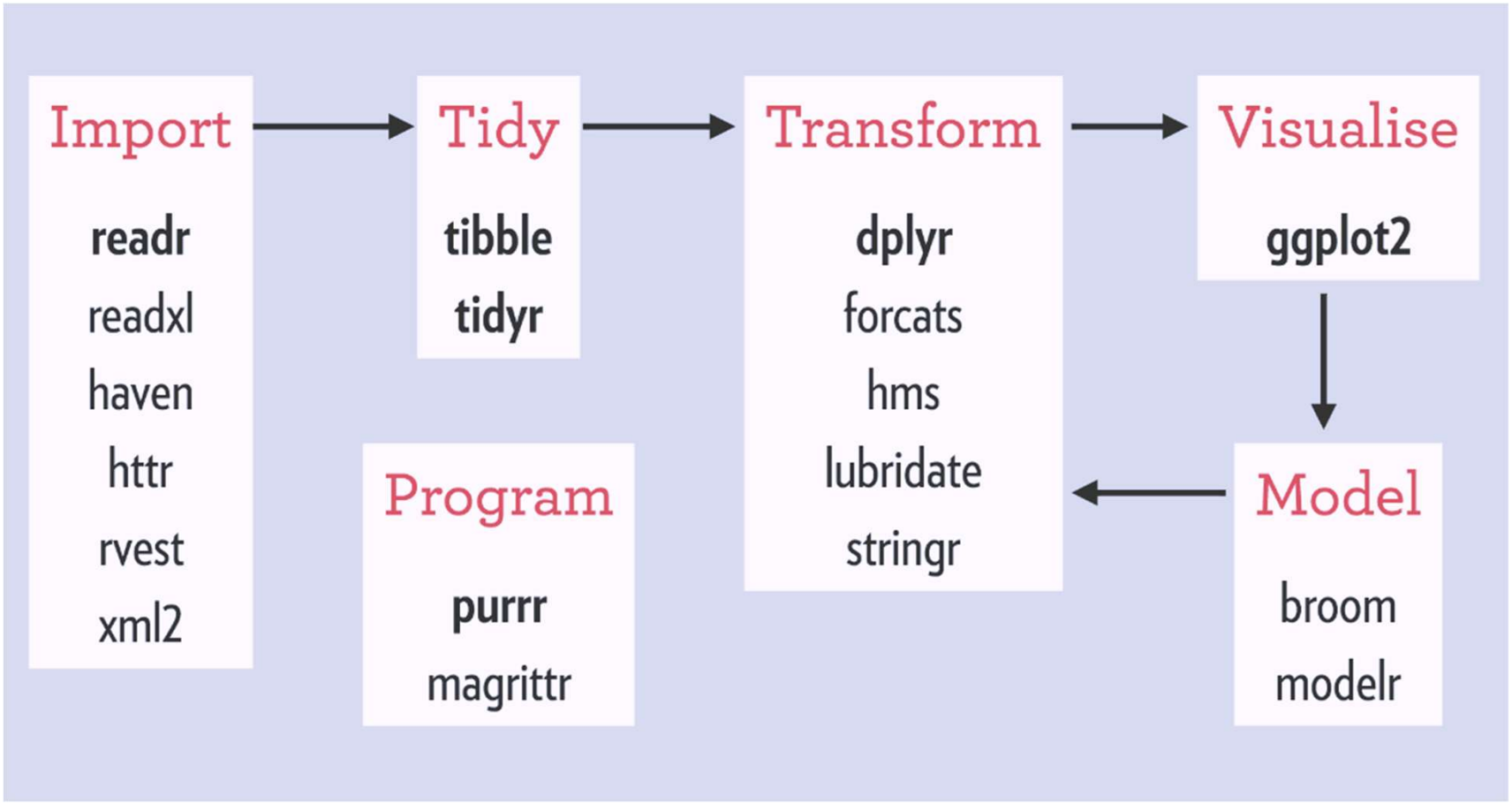
`right_join(), full_join()`

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

DATENAUFBEREITUNG



TIDYVERSE



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

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

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

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+$", "")
```

WARENGRUPPEN

1 Brot

2 Brötchen

3 Croissant

4 Konditorei

5 Kuchen

6 Saisonbrot