27.04.21

Einführung in Data Science und maschinelles Lernen mit R

Versionierung mit git (Teil 1) und Datenaufbereitung mit Tidyverse

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

SKALENTYPEN

- **Nominalskaliert (kategorial)**
  [Geschlecht, Religionszugehörigkeit]

- **Ordinalskaliert**
  [Englischnote, Testantwort auf einer Skala gut–mittel–schlecht]

- **Intervallskaliert**
  [Temperatur in Celsius, Intelligenzquotient]

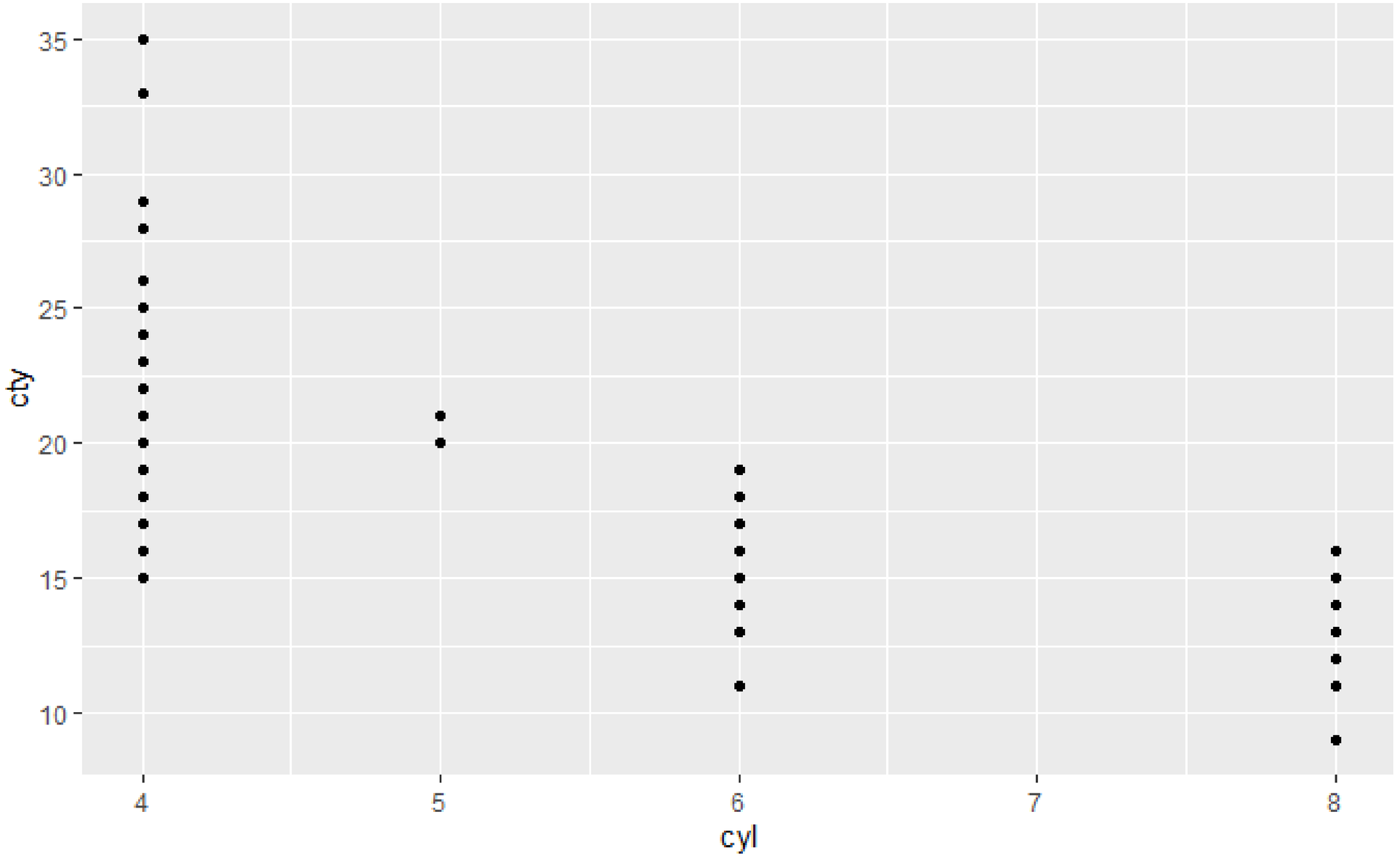
- **Verhältnisskaliert**
  [Geschwindigkeit, Einkommen]

GÄNGIGE DIAGRAMMTYPEN

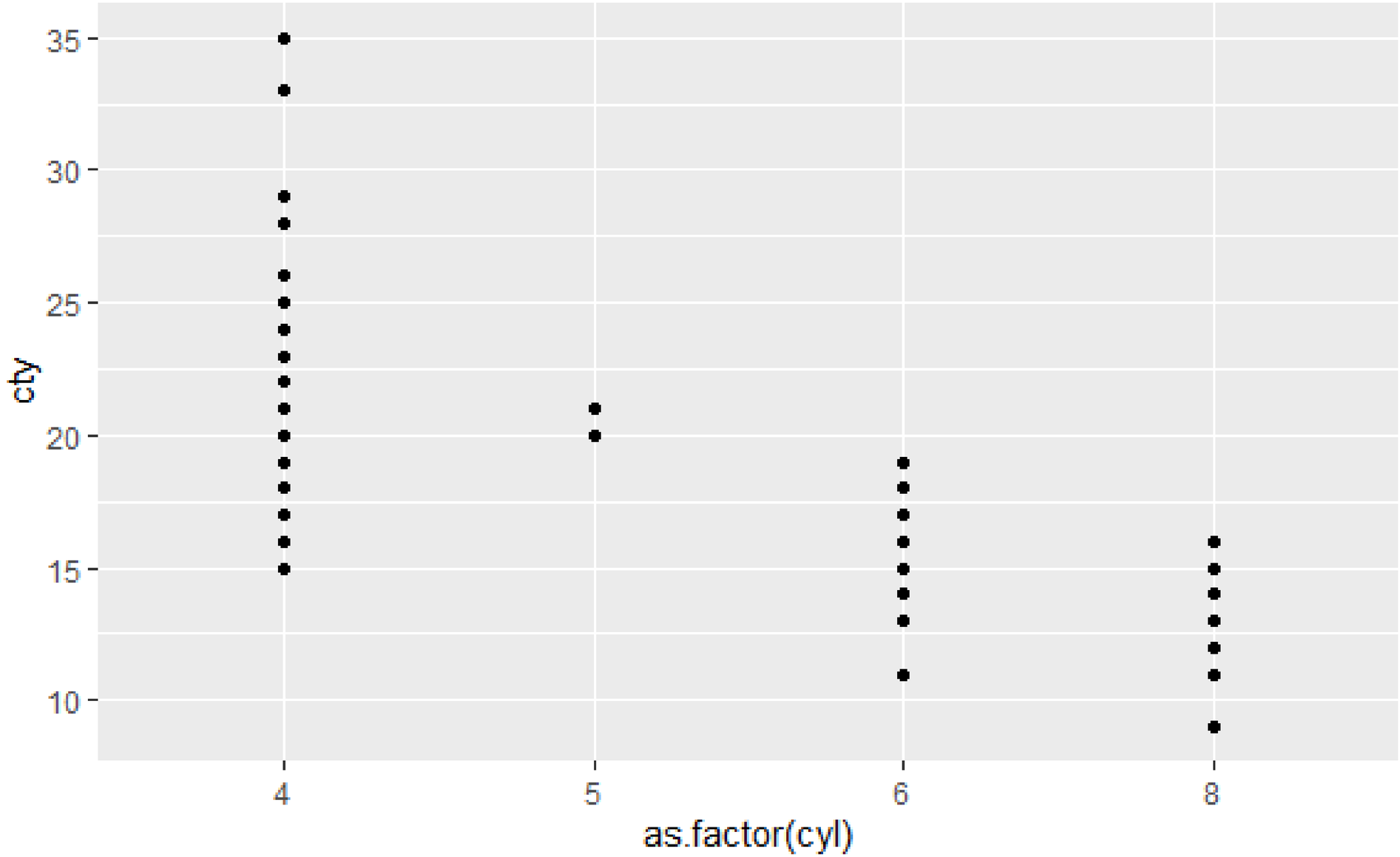
- **Histogramm**
  Darstellung der Verteilung einer numerischen (mind. ordinalen) Variable

- **Scatterplot**
  Darstellung der Beziehung von zwei numerischen (mind. ordinalen) Variablen

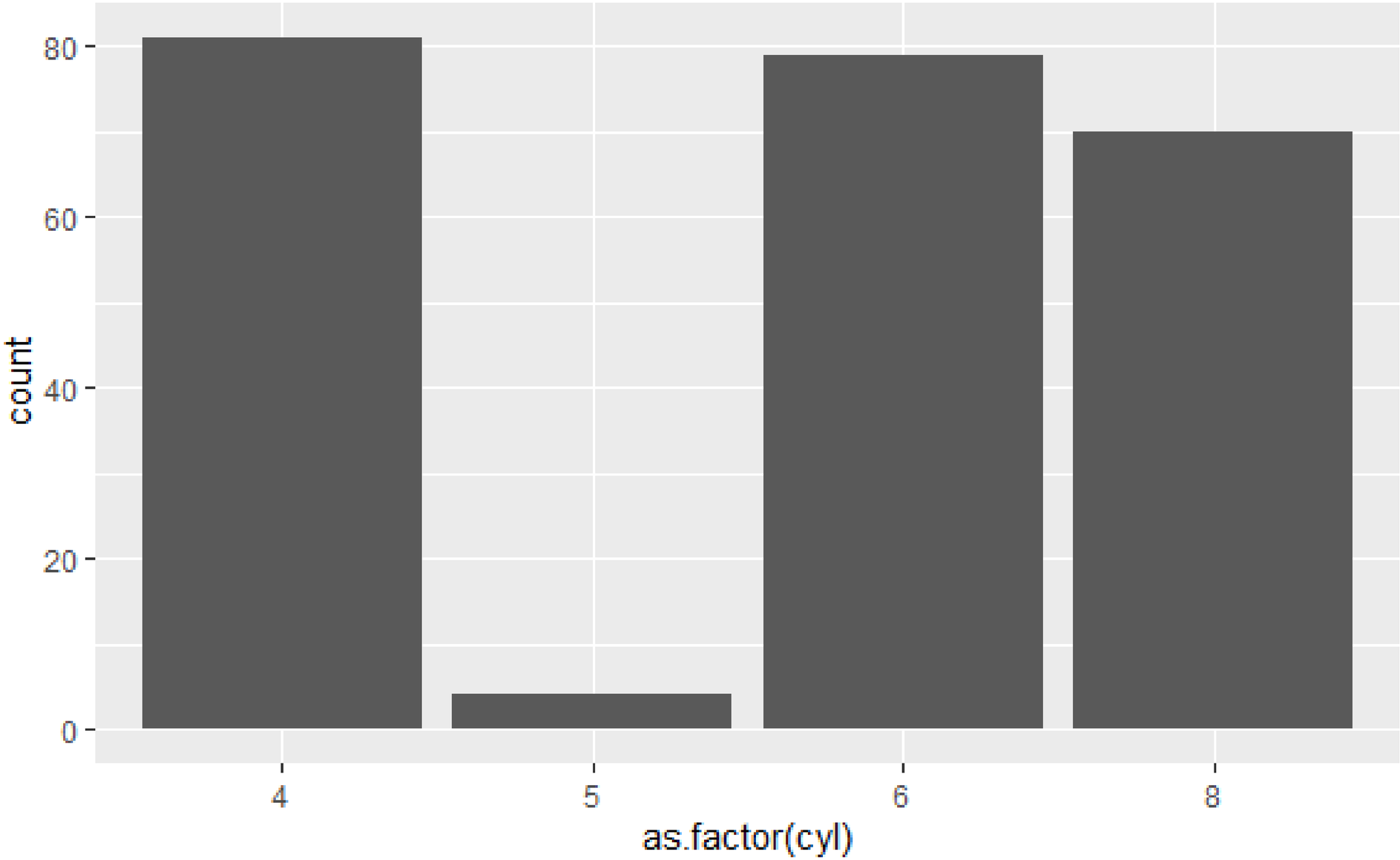
- **Balkendiagramm (Barplot)**
  Darstellung zwischen einer numerischen (mind. ordinalen Variable) und einer kategoriellen Variable



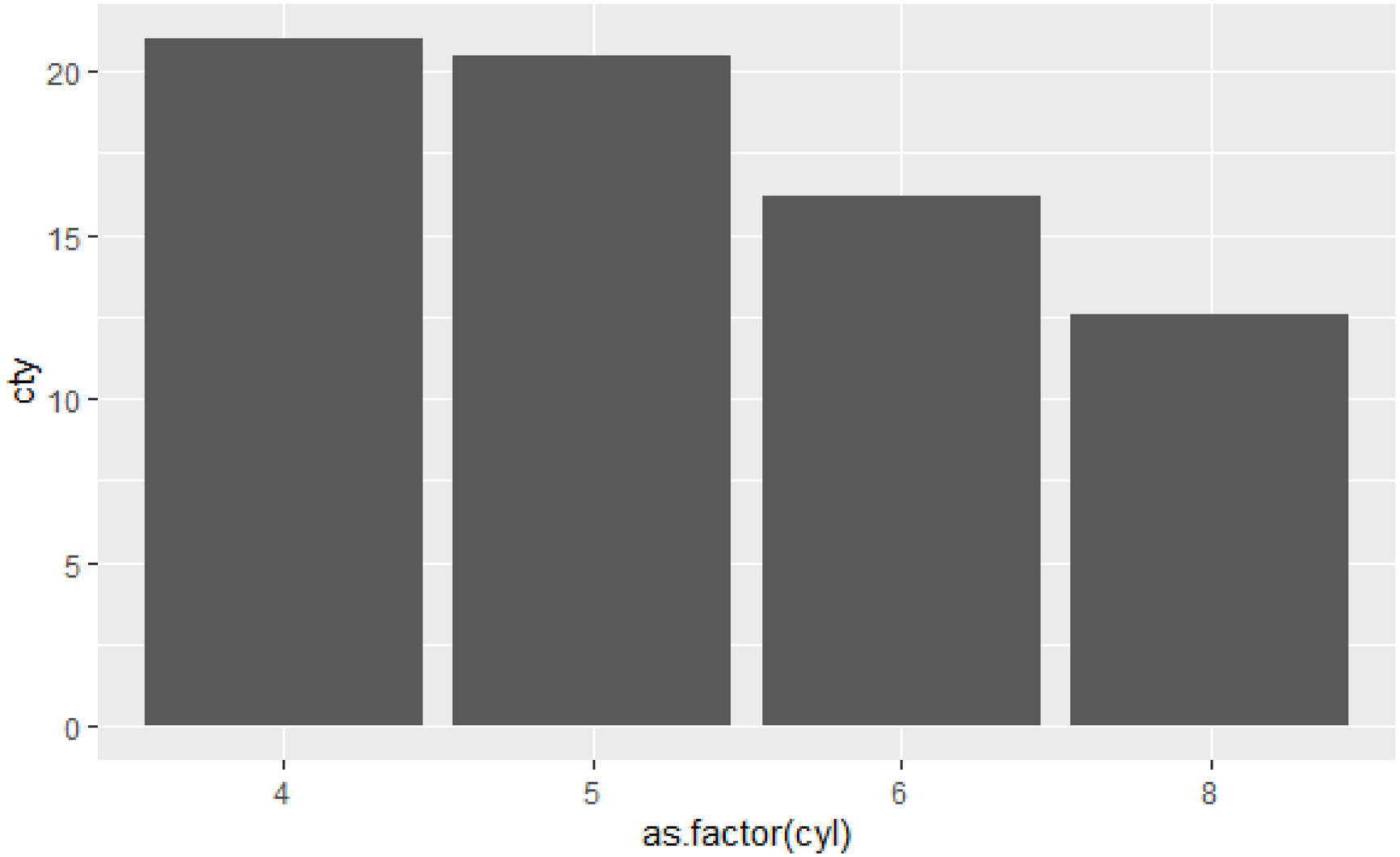

```
ggplot(mpg)+
  geom_point(aes(x = cyl, y = cty))
```




```
ggplot(mpg)+
  geom_point(aes(x = as.factor(cyl), y = cty))
```




```
ggplot(mpg)+
  geom_bar(aes(x = as.factor(cyl)))
```




```
ggplot(mpg)+
  geom_bar(aes(x = as.factor(cyl), y = cty),
           stat="summary", fun="mean")
```

BREAKOUT

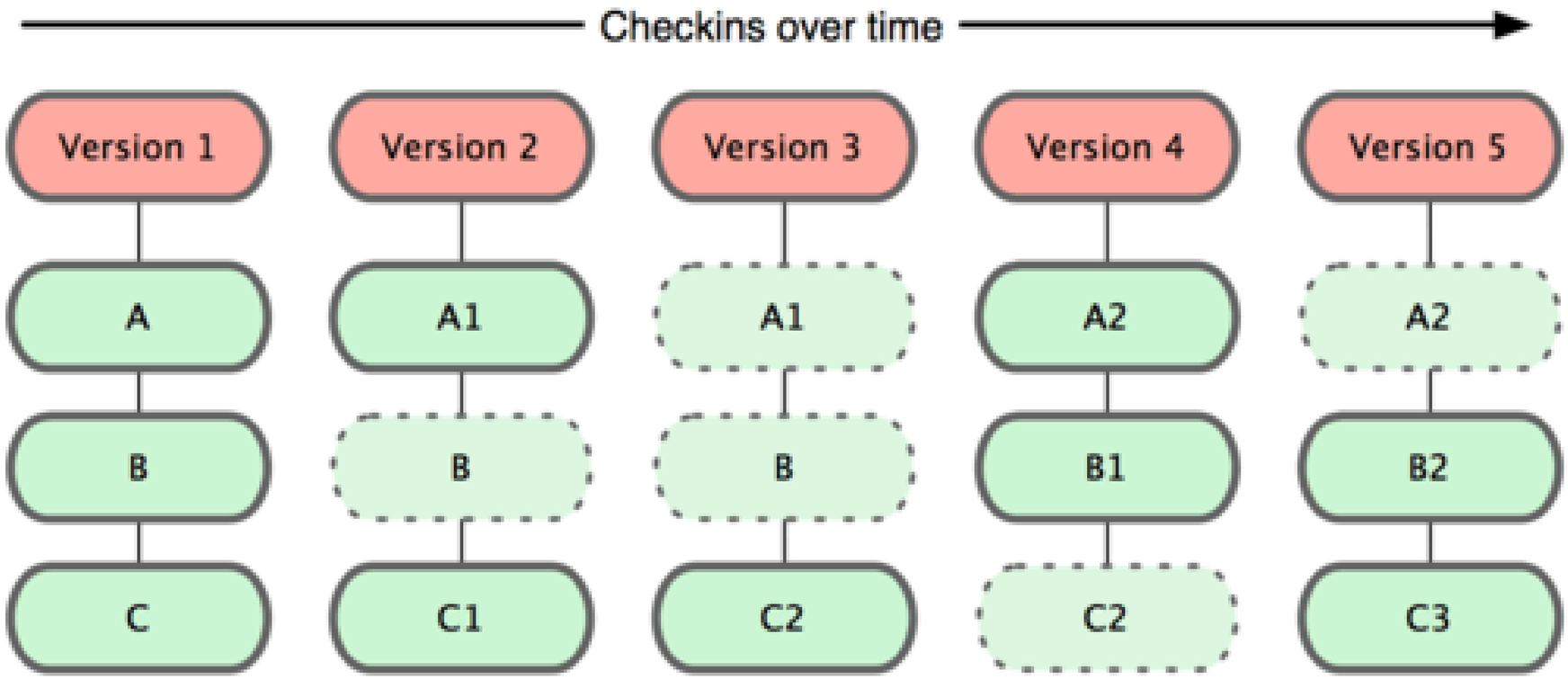
- **Erstelle ein Balkendiagramm, dass über alle Warengruppen hinweg die durchschnittlichen Umsätze je Wochentag zeigt.**

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

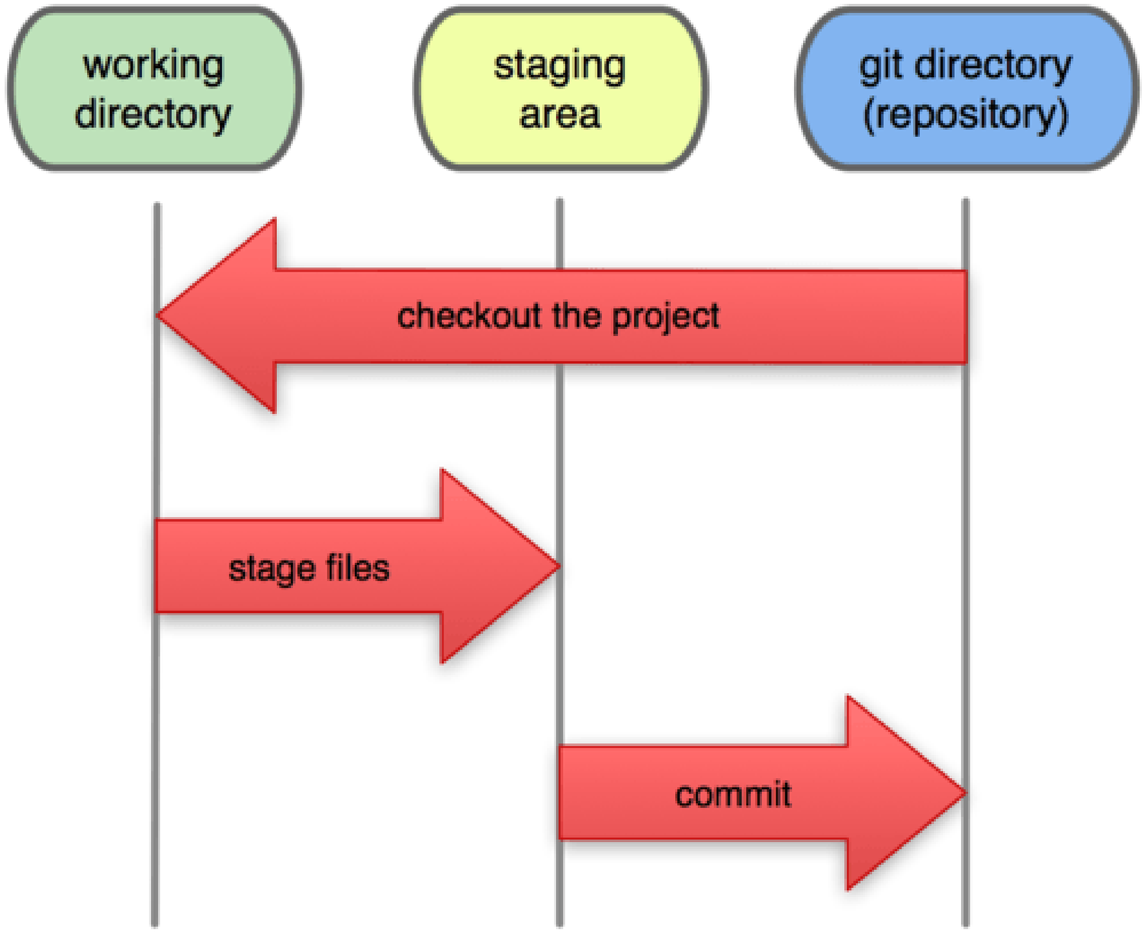
- ***Freiwillige Zusatzaufgabe:***
  Stelle die Umsätze je Wochentag getrennt nach Warengruppe dar (ein eigenes Balkendiagramm je Warengruppe)

VERSIONIERUNG MIT GIT

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

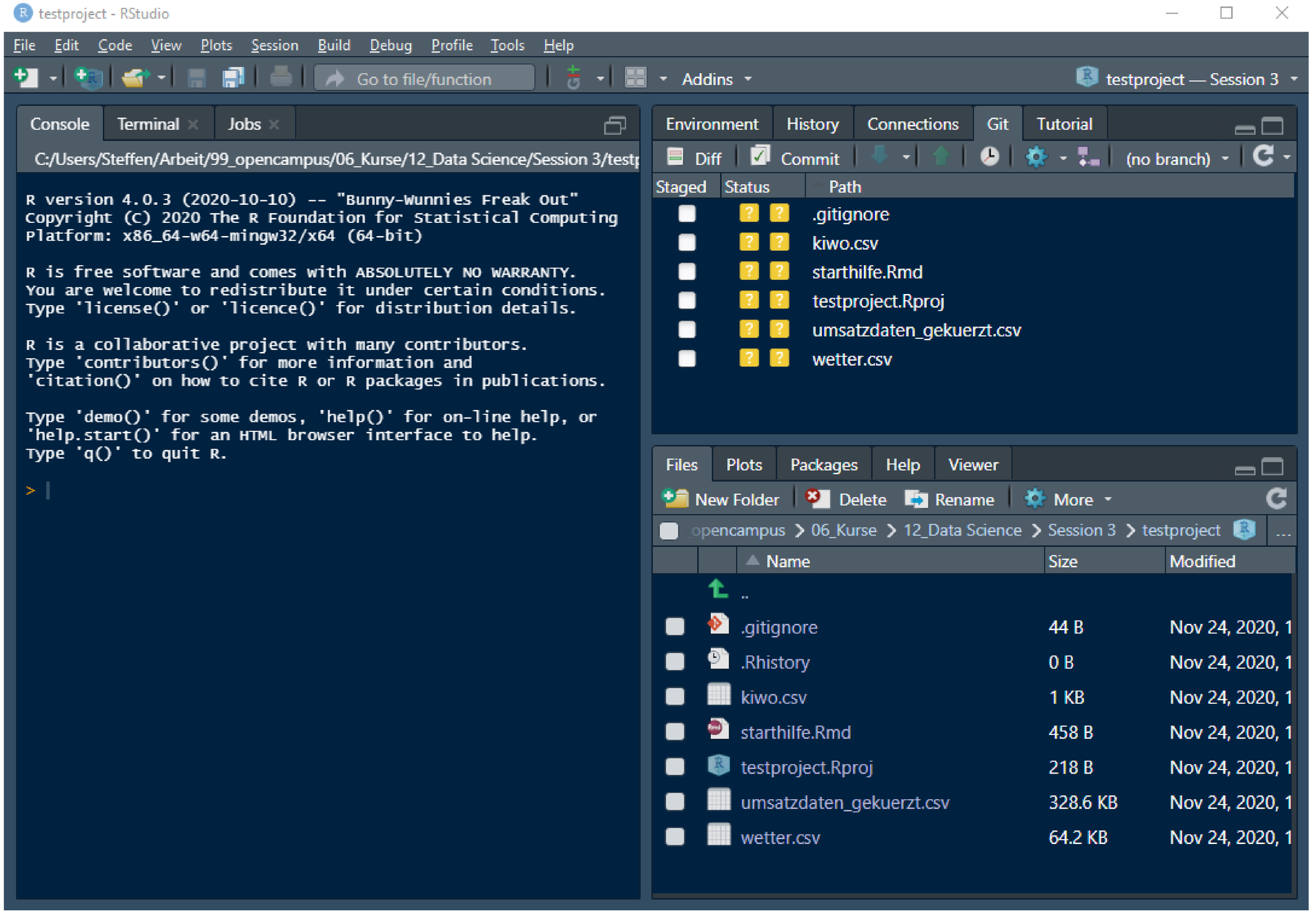


VERSIONIERUNG MIT GIT



Eine Datei kann drei mögliche Zustände haben:

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


testproject - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
testproject — Session 3
Console
Terminal
Jobs
C:/Users/Steffen/Arbeit/99_opencampus/06_Kurse/12_Data Science/Session 3/testp
R version 4.0.3 (2020-10-10) -- "Bunny-Wunnies Freak Out"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment
History
Connections
Git
Tutorial
Diff
Commit
(no branch)
Staged
Status
Path
.gitignore
kiwo.csv
starthilfe.Rmd
testproject.Rproj
umsatzdaten_gekuerzt.csv
wetter.csv
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
opencampus
06_Kurse
12_Data Science
Session 3
testproject
Name
Size
Modified
..
.gitignore 44 B Nov 24, 2020, 1
.Rhistory 0 B Nov 24, 2020, 1
kiwo.csv 1 KB Nov 24, 2020, 1
starthilfe.Rmd 458 B Nov 24, 2020, 1
testproject.Rproj 218 B Nov 24, 2020, 1
umsatzdaten_gekuerzt.csv 328.6 KB Nov 24, 2020, 1
wetter.csv 64.2 KB Nov 24, 2020, 1

KONFIGURATION VON GIT

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

- **Gebt dazu im Terminal-Fenster (unten links in RStudio) einen 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, üblich ist zum Beispiel den Benutzernamen aus GitHub zu nehmen, falls Ihr dort schon einen habt.

BREAKOUT

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.

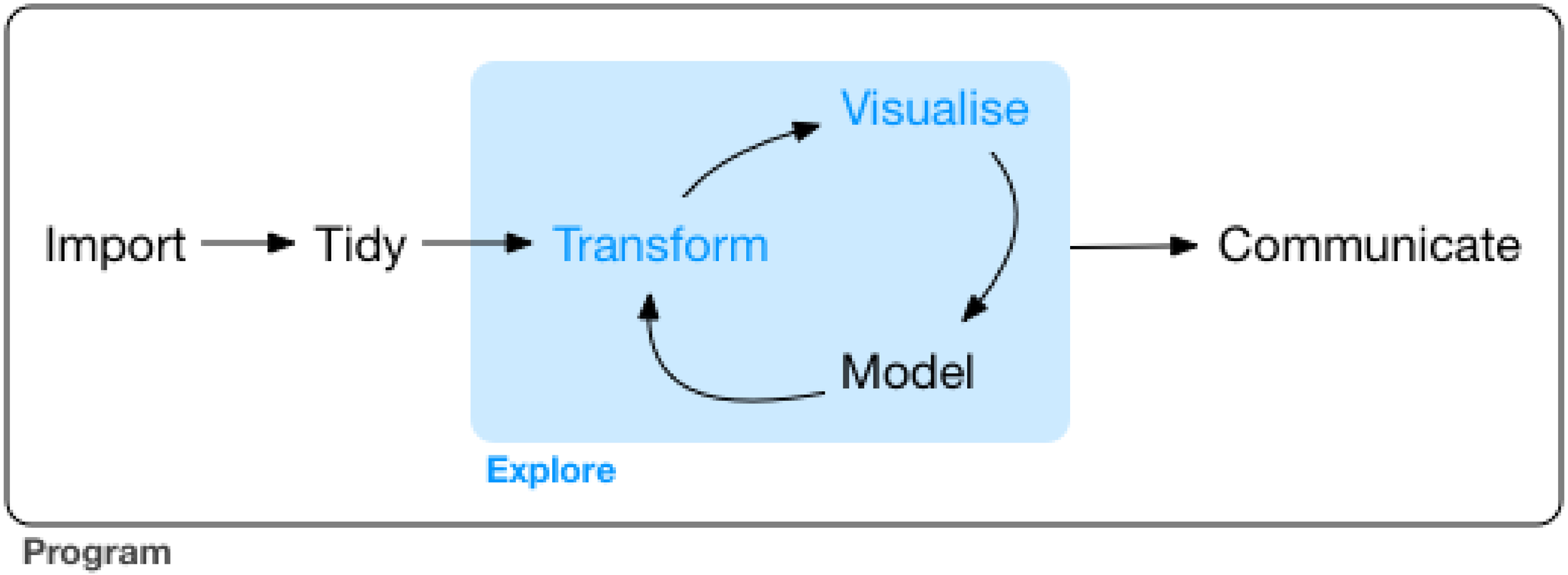
5) Schaut Euch die History Eures Repositories an.

VERSIONIERUNG MIT GIT

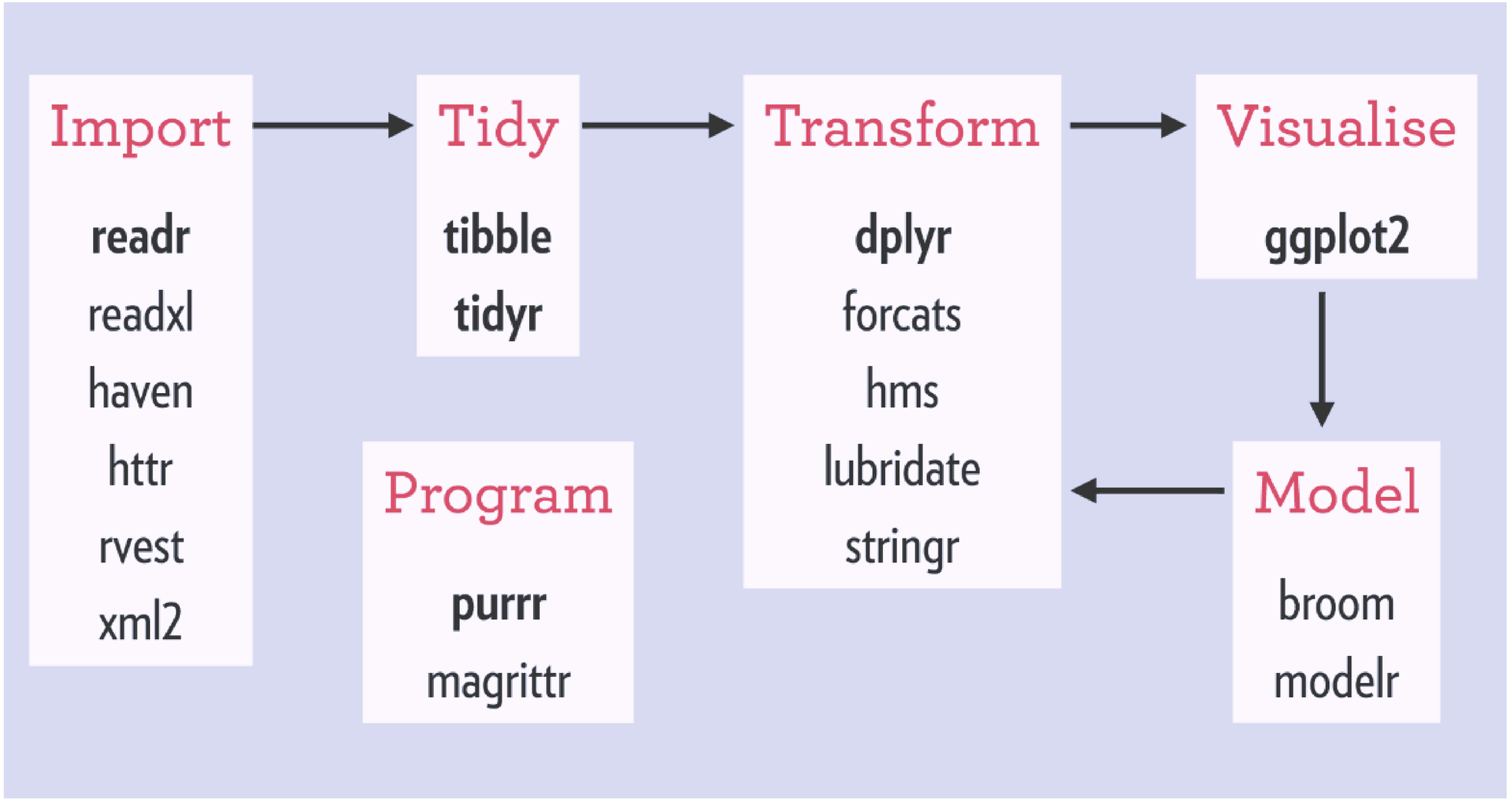
Teil 1: Lokale Versionierung

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/

BEISPIEL

```
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 Programmcodes
- Erhöht die Lesbarkeit des Programmcodes

DPLYR - DATENTABELLEN ZUSAMMENFÜHREN

`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 - DATEN SELEKTIEREN & HINZUFÜGEN

select()

Variablen (Spalten) auswählen

filter()

Fälle (Zeilen) auswählen

mutate()

Variablen hinzufügen

```
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

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

Umwandlung von Datumformaten in kategoriale Variablen

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

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

STRINGR

Allgemeine Funktion zur Zeichenersetzung

- `str_replace()`
- Erlaubt die Verwendung von „regular expressions“

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

Funktionen für spezielle Aufgaben

- „Wrapper-Funktionen“ von `str_replace()`
- Z.B. zum entfernen führender und nachstehender Leerzeichen:

```
str_trim("   Vorname   ")
```

→ `str_replace_all("   Vorna me   ","^[ \\s]+|[ \\s]+$", "")`

ANDERE NÜTZLICHE PACKAGES

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

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.