

Einführung in Data Science und maschinelles Lernen

VERSIONIERUNG MIT GIT (TEIL 1) UND DATENAUFBEREITUNG MIT TIDYVERSE

- **Besprechung Übungsaufgaben**
- **Weitere Beispiele zu Diagrammen**
- **Statistische Signifikanz**
- **Einführung in die Versionierung mit git**
- **Einführung in Tidyverse und die Datenaufbereitung**

BREAKOUT

- **Lege ein R-Studio-Projekt-Verzeichnis an und speichere dort die Dateien „kiwo.csv“, „umsatzdaten_gekuerzt.csv“ und „wetter.csv“ aus diesem GitHub-Repository:**
<https://github.com/opencampus-sh/einfuehrung-in-data-science-und-ml>
- **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“).**
- **Stelle die Umsätze je Wochentag getrennt nach Warengruppe dar (ein eigenes Balkendiagramm je Warengruppe)**

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 BASICS

- Eine ggplot Abbildung ist ein R-Objekt, das über eine beliebige Anzahl von „Layern“ definiert wird.
- Jedes Objekt wird mit `ggplot()` erzeugt.
- Die wichtigsten Layer sind:
 - Aesthetics* – `aes()`
Zurordnung von Daten zur Abbildung (x-Werte, y-Werte, Label, Farbwerte dargestellter Punkte, ...)
 - Geometries* – `geoms()`
Definition der Darstellungsform (Histogramm, Scatterplott, ...)
- Jeder Layer wird durch ein „+“ hinzugefügt.

WEITERE GGLOT LAYER

- ***Facets***
Layout von mehreren, nebeneinander dargestellten Abbildungen in einer Grafik
- ***Statistics***
Durchführung/Darstellung einfacher statistischer Funktionen
- ***Coordinates***
Definition/Layout des Raums, in dem die Daten dargestellt werden.
- ***Themes***
Selektion von Templates mit unterschiedlichen (datenunabhängigen) Voreinstellungen

BEISPIEL SCATTERPLOT

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

Grundlegende Datentabelle ist nicht definiert,
Datentabelle muss also immer angegeben werden.

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

Grundlegende Datentabelle wird für alle
nachfolgenden Layer definiert.

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

Aesthetics werden nur speziell für diesen
Layer definiert.

WEITERE BEISPIELE

(siehe Beispielcode zu dieser Woche)

Scatterplot

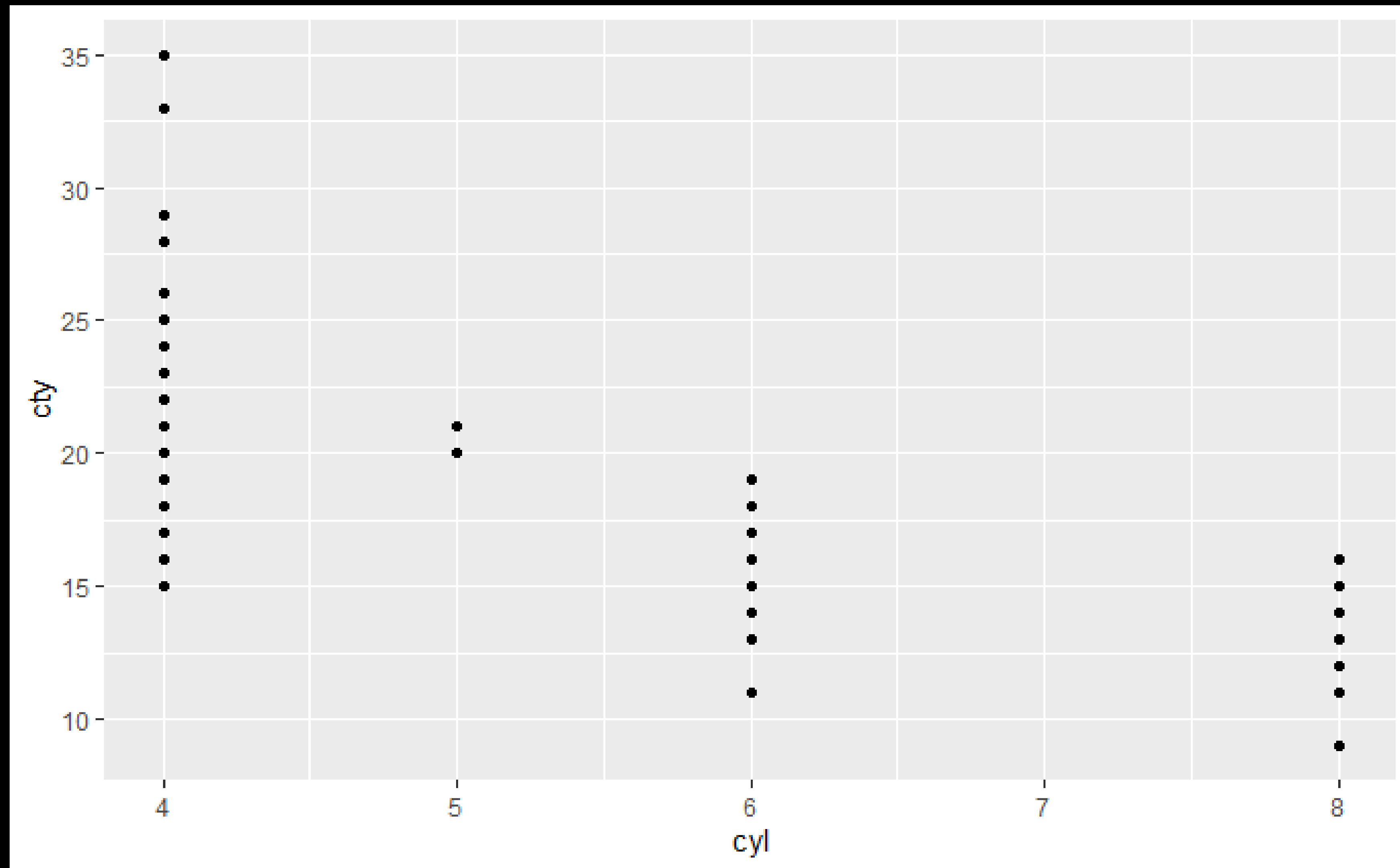
```
ggplot(mpg)+  
  geom_point(aes(x = hwy, y = cty, color = displ))
```

Histogramm

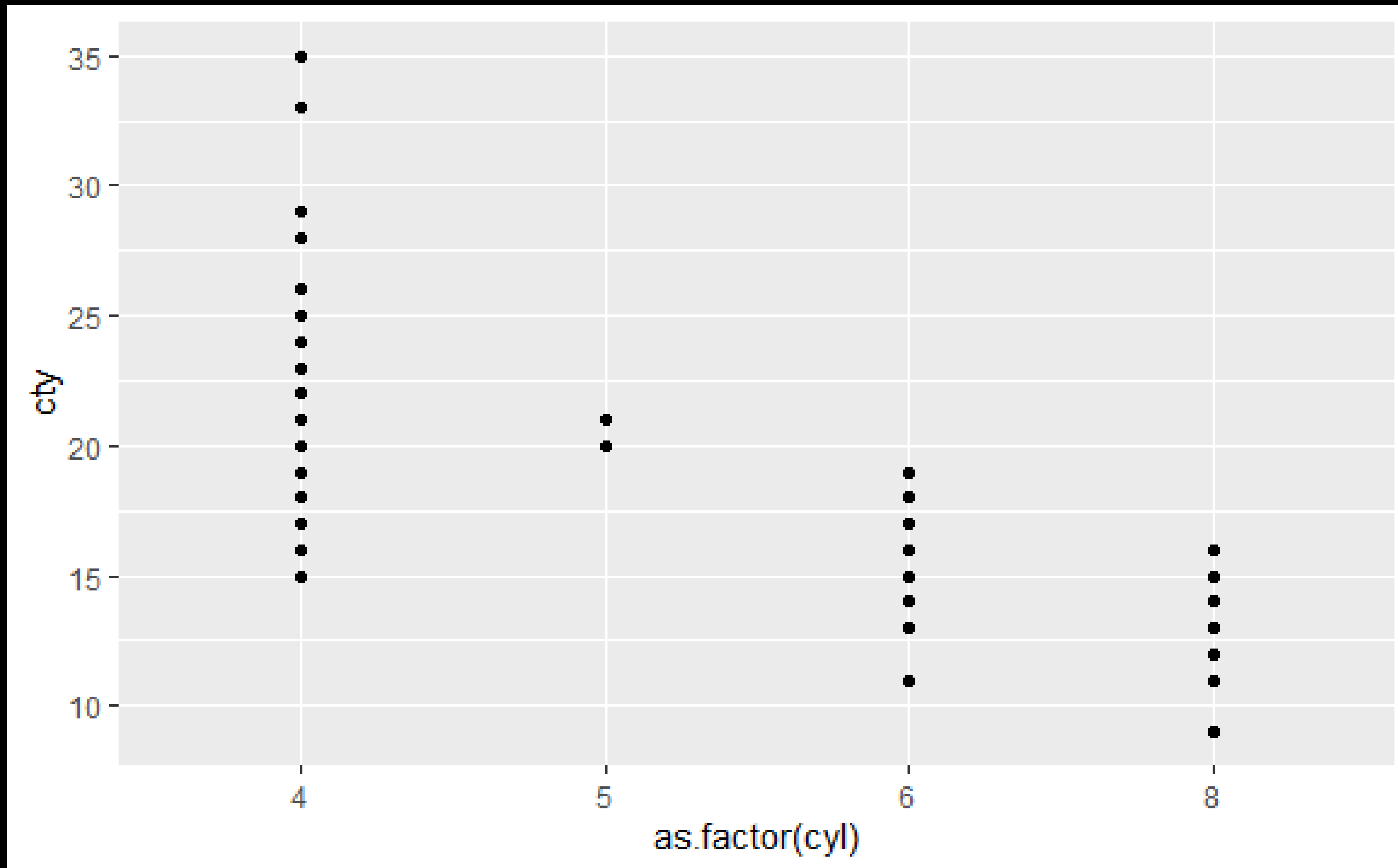
```
ggplot(mpg)+  
  geom_histogram(aes(x = cty))
```

Balkendiagramm

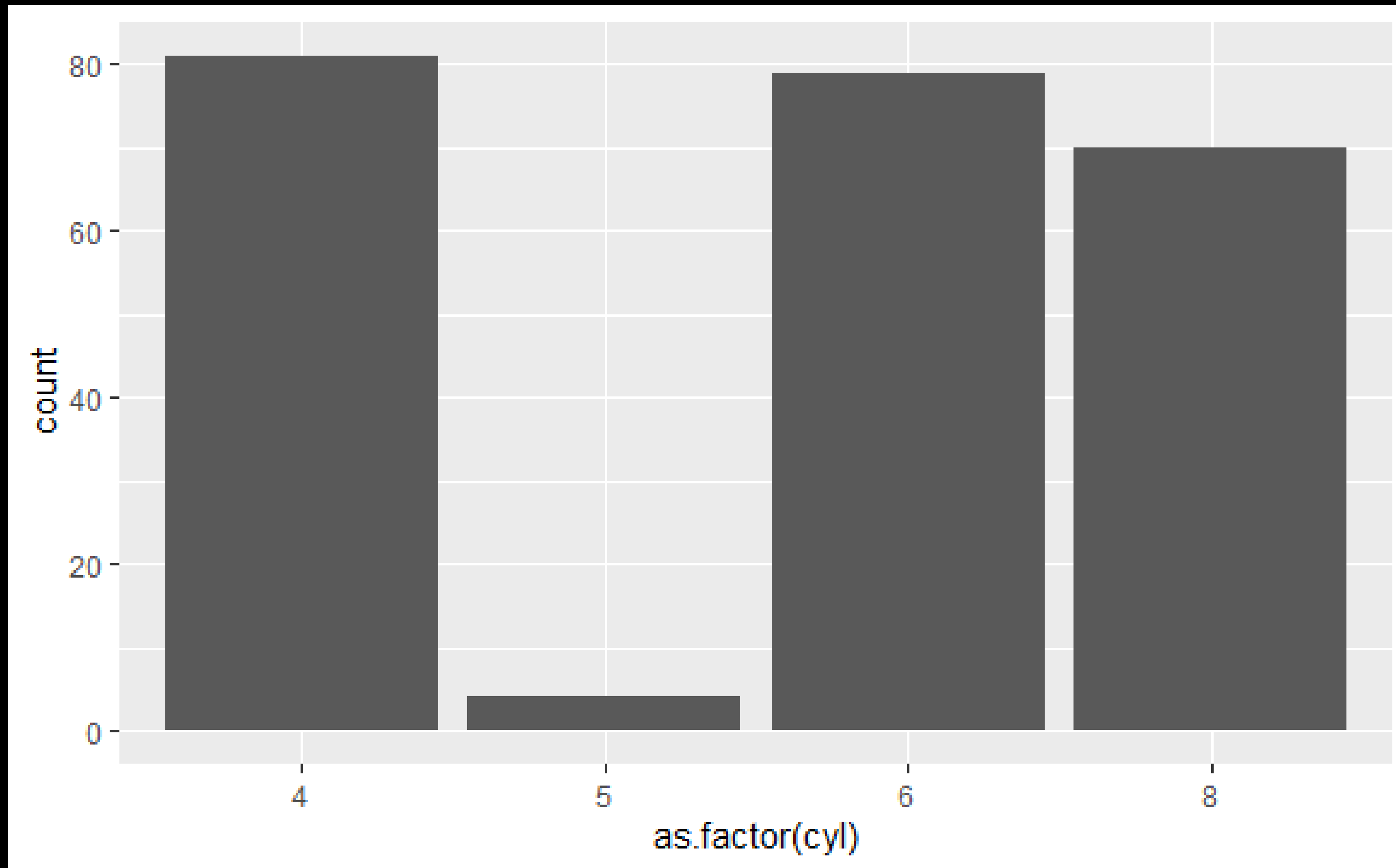
```
ggplot(mtcars)+  
  geom_bar(aes(x = as.factor(cyl), y = mpg), stat = "identity")
```

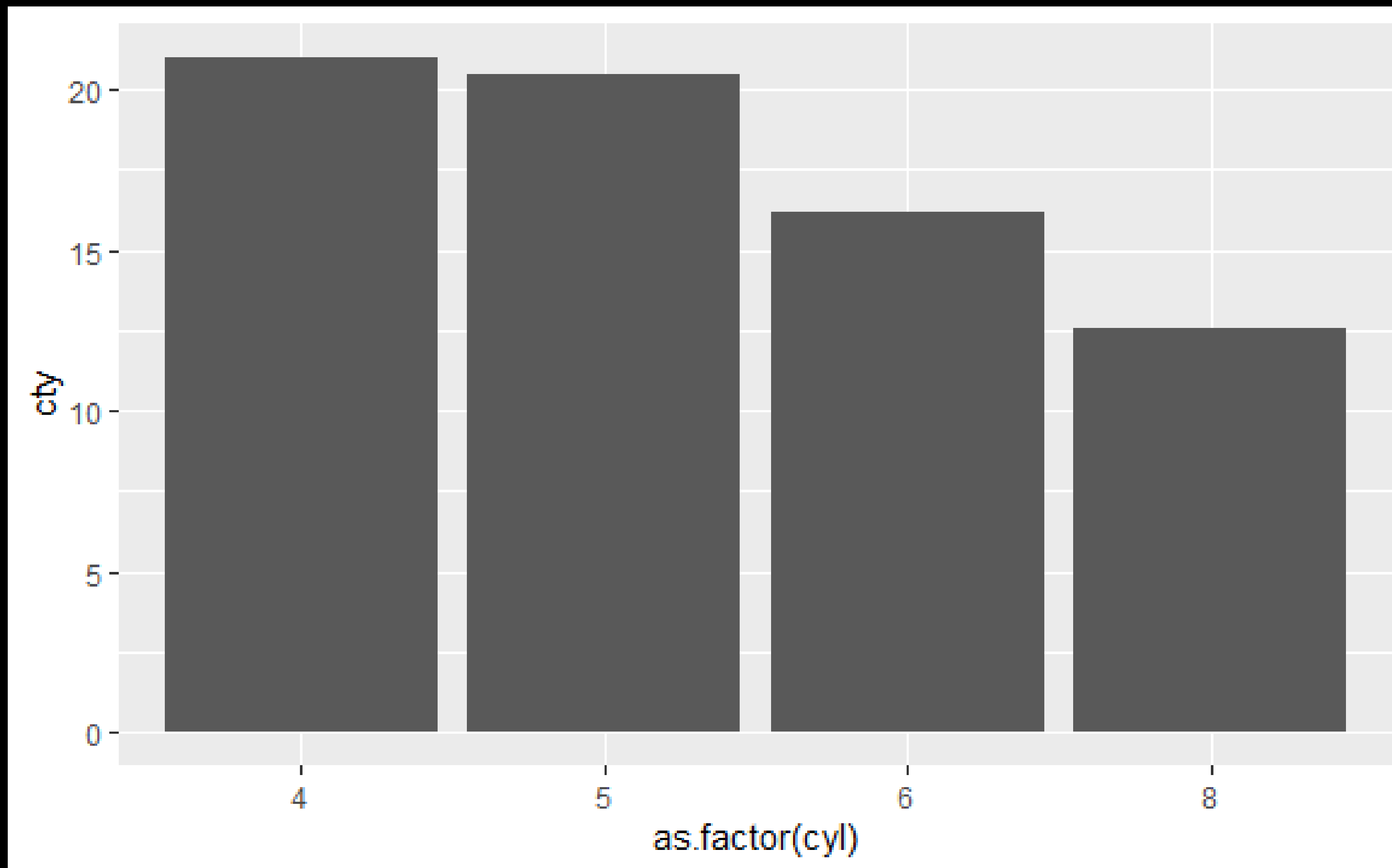
```
ggplot(mpg)+  
  geom_point(aes(x = cyl, y = cty))
```



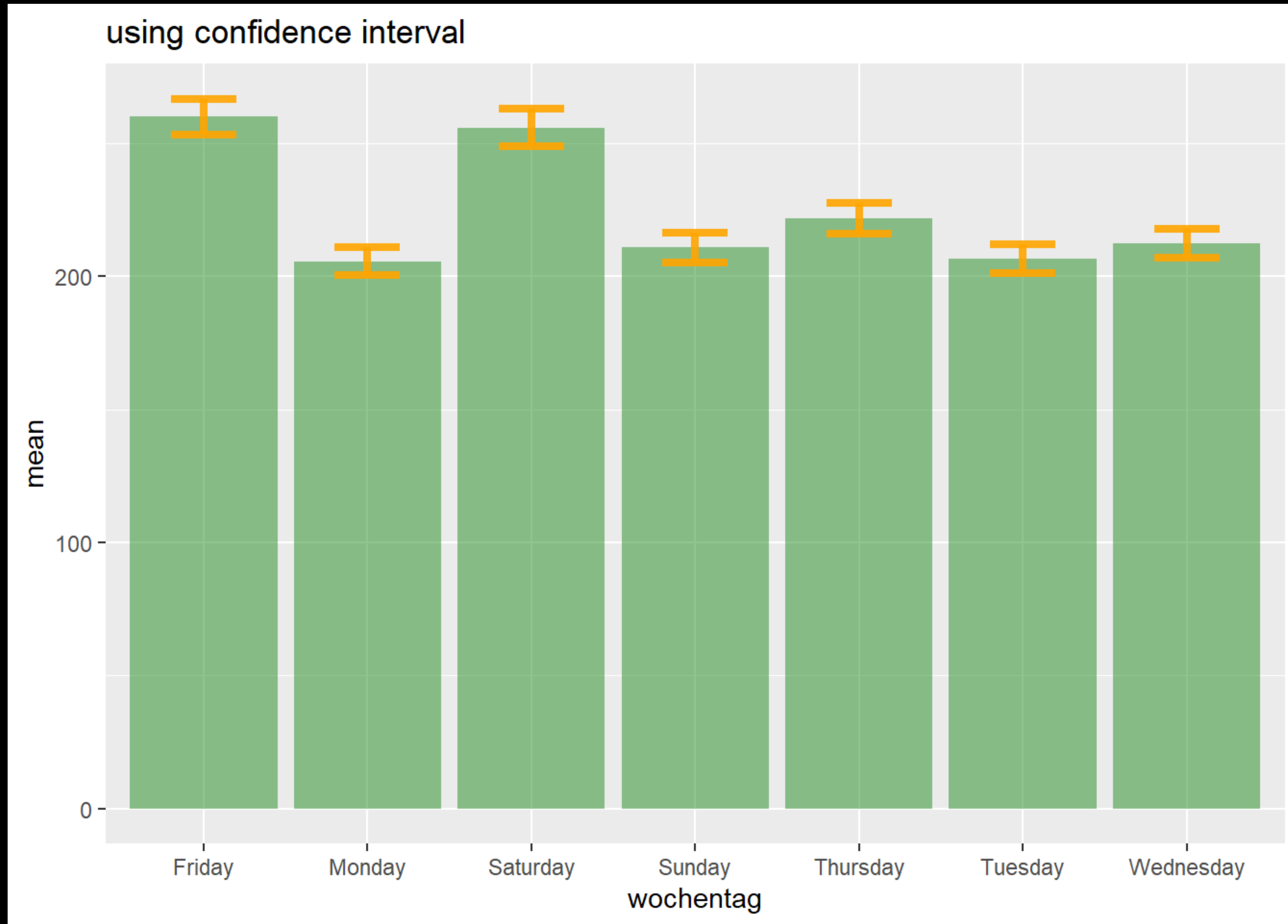
```
ggplot(mpg)+  
  geom_point(aes(x = as.factor(cyl), y = cty))
```



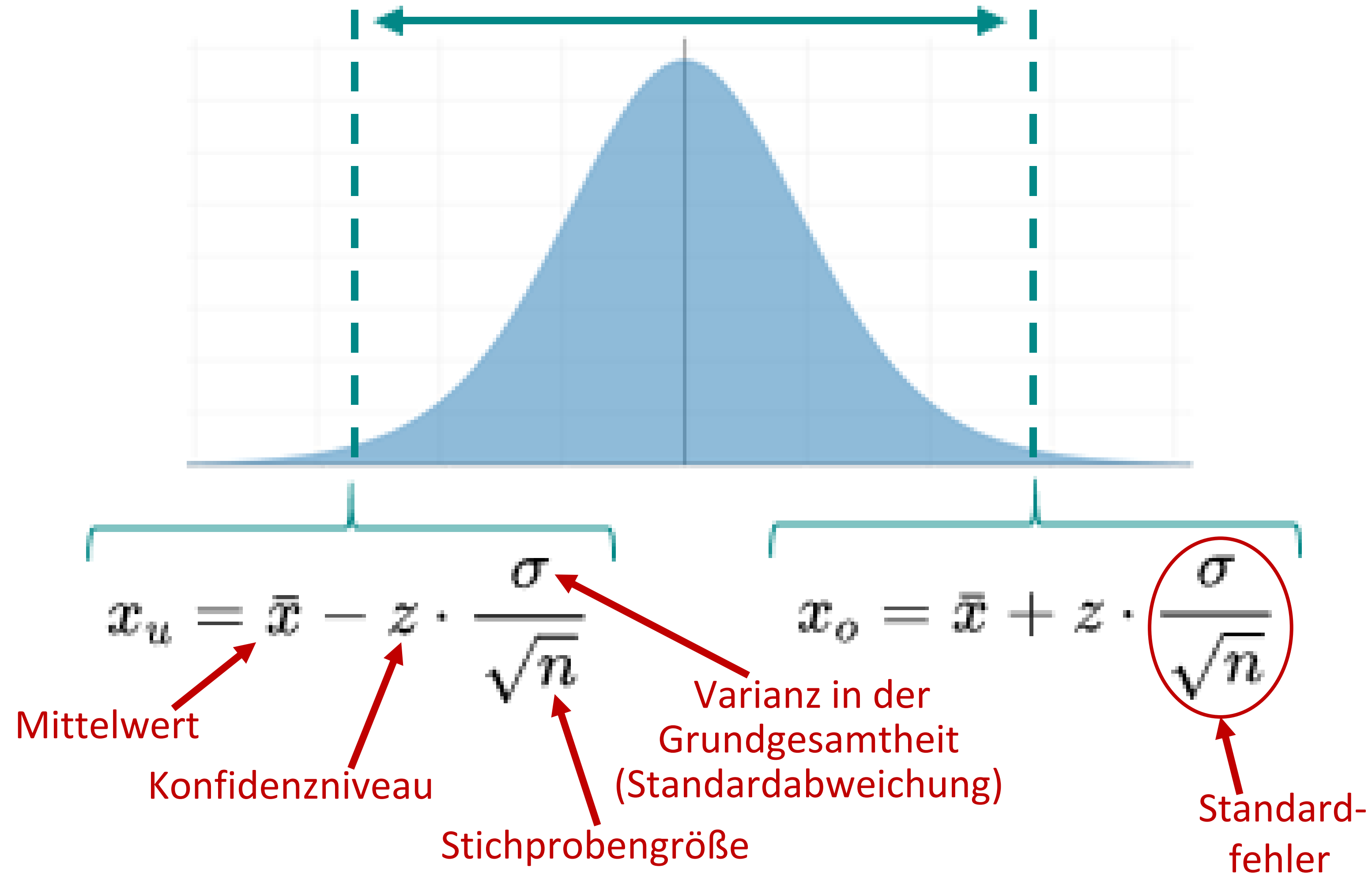
```
ggplot(mpg)+  
  geom_bar(aes(x = as.factor(cyl)))
```



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



Konfidenzintervall



T-TEST

```
> t.test(july$Temp, may$Temp)
```

```
Welch Two Sample t-test
```

```
data: july$Temp and may$Temp
```

```
t = 12.616, df = 50.552, p-value < 2.2e-16
```

```
alternative hypothesis: true difference in means is not equal to 0
```

```
95 percent confidence interval:
```

```
15.43351 21.27617
```

```
sample estimates:
```

```
mean of x mean of y
```

```
83.90323 65.54839
```

SIGNIFIKANZ VERSUS RELEVANZ

- **Eigenschaften können sich statistisch signifikant unterscheiden, aber der Unterschied kann ohne praktische Relevanz (ohne „Effekt“) sein.**

Betrachtung der Effektstärke nach Cohen:
(vereinfachter Fall gleicher Stichprobengröße und Varianz)

$$d = \frac{\mu_1 + \mu_2}{\sigma}$$

$$|d| > 0,2$$

kleiner Effekt

$$|d| > 0,5$$

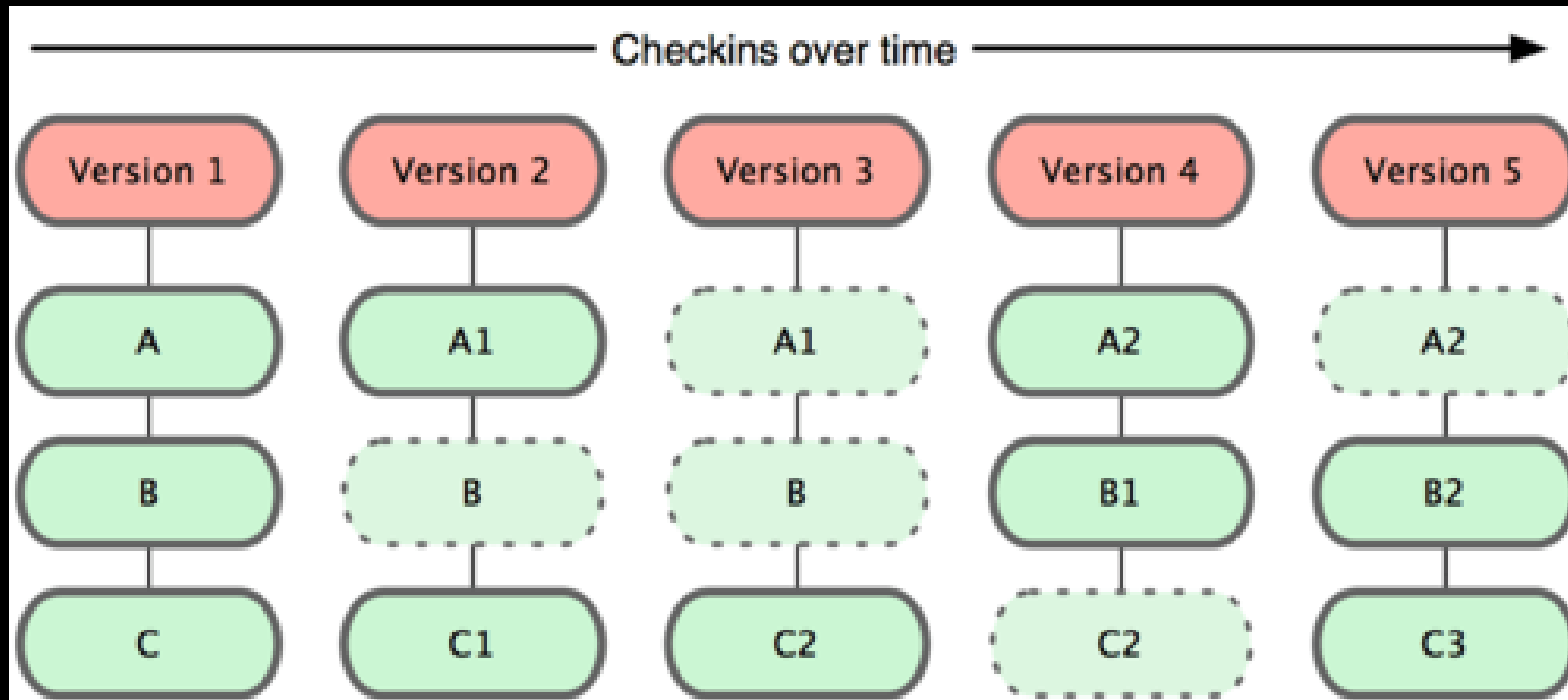
mittlerer Effekt

$$|d| > 0,8$$

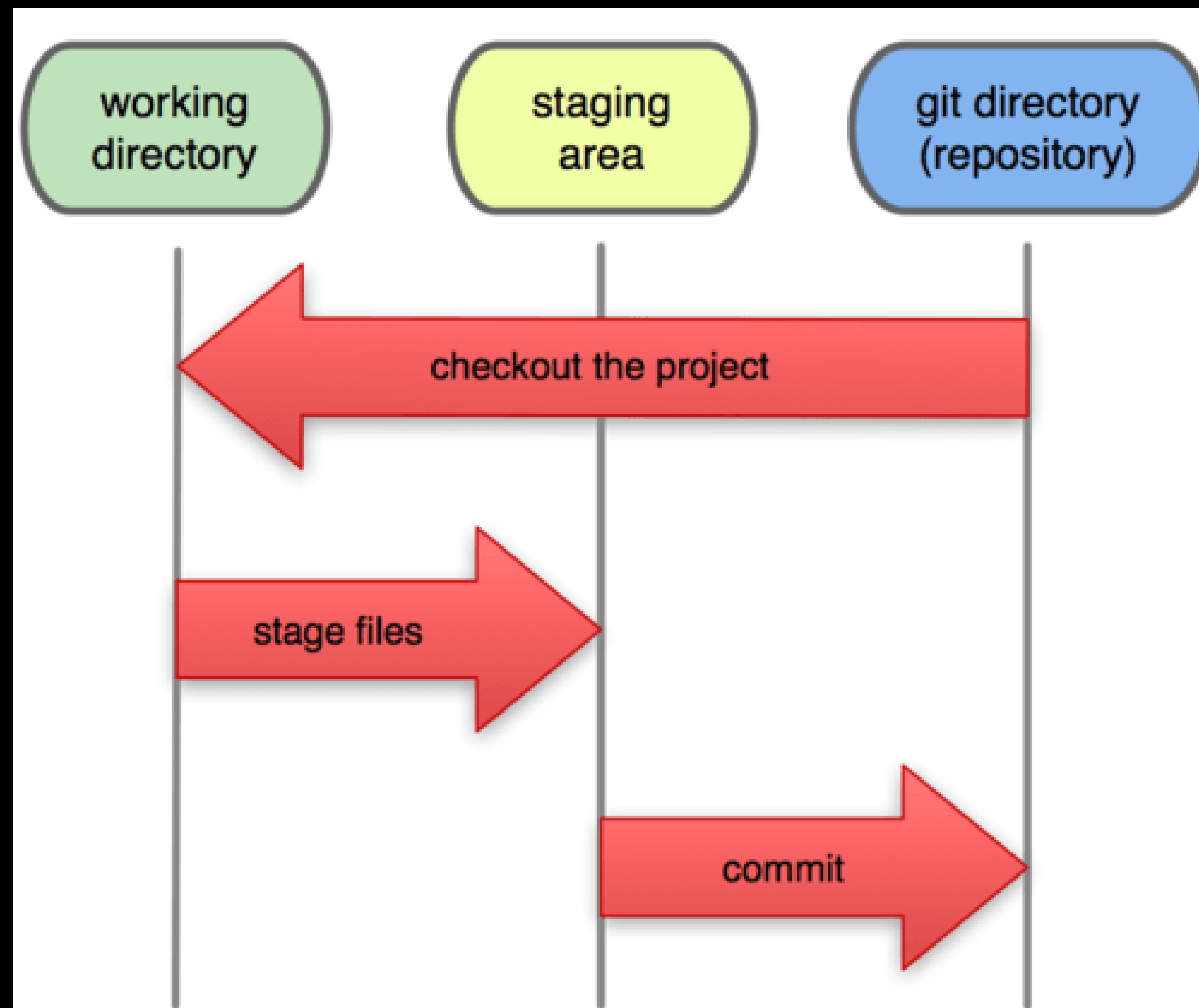
großer Effekt

VERSIONIERUNG MIT GIT

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

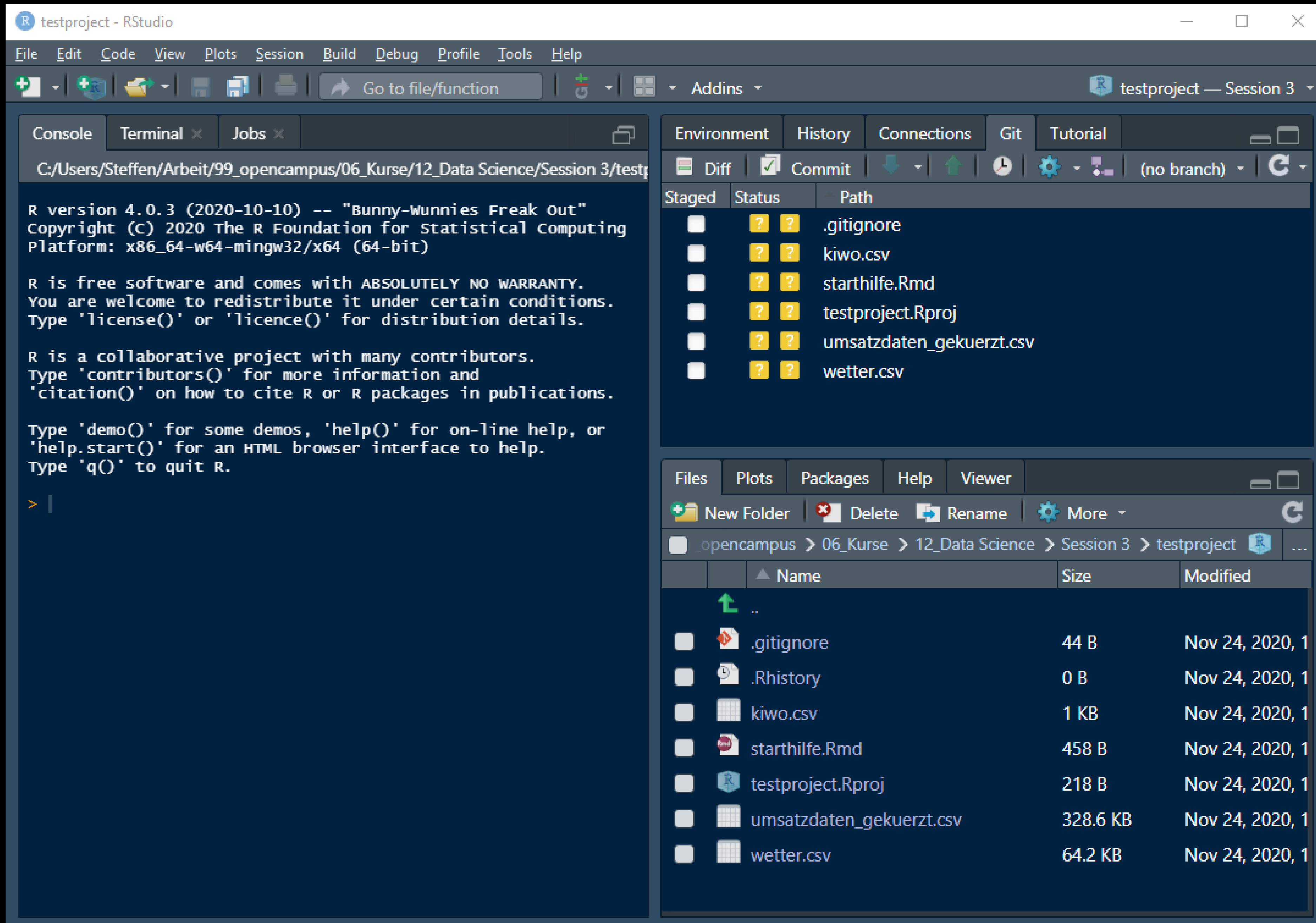


VERSIONIERUNG MIT GIT



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.

- **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

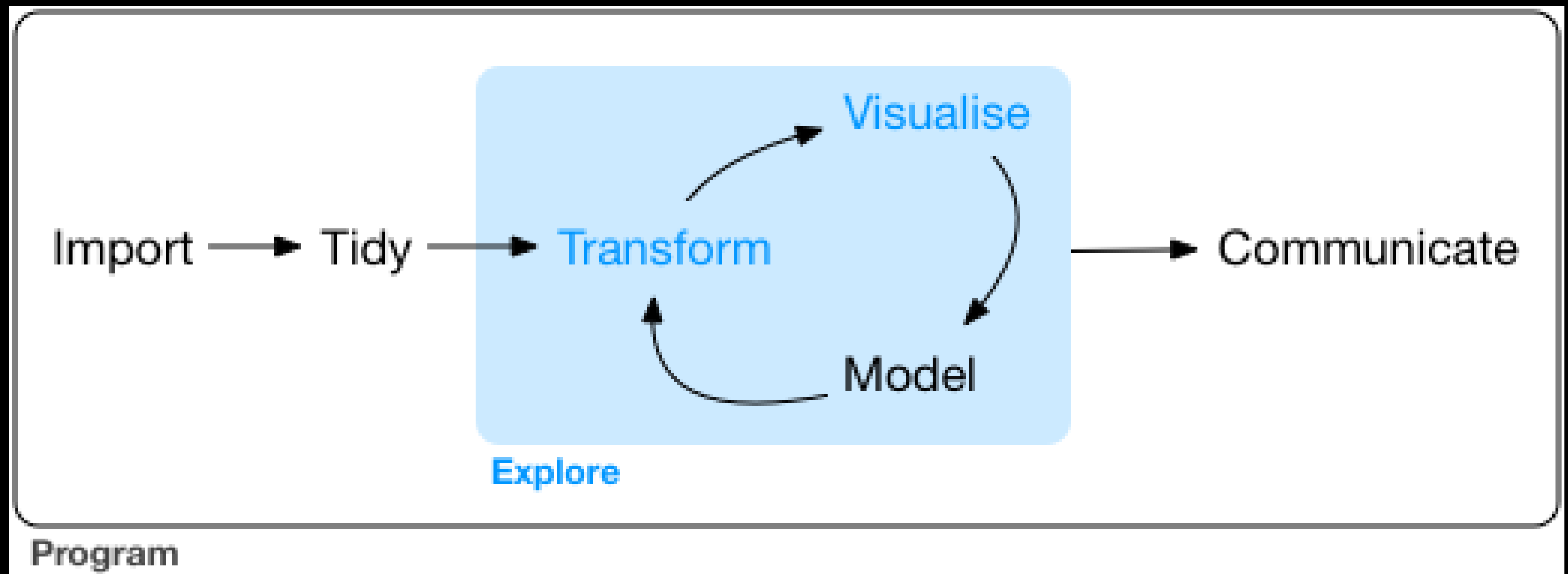
- 1) Wählt git als Versionierungsanwendung für Euer Projektverzeichnis aus.**
- 2) „Staged“ alle Dateien (markiert sie für das nächste Commit), die Ihr versionieren wollt und „committed“ sie dann.**
- 3) Führt ein erstes „Commit“ aus, um eine erste Projektversion mit allen bisherigen Dateien anzulegen.**
- 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

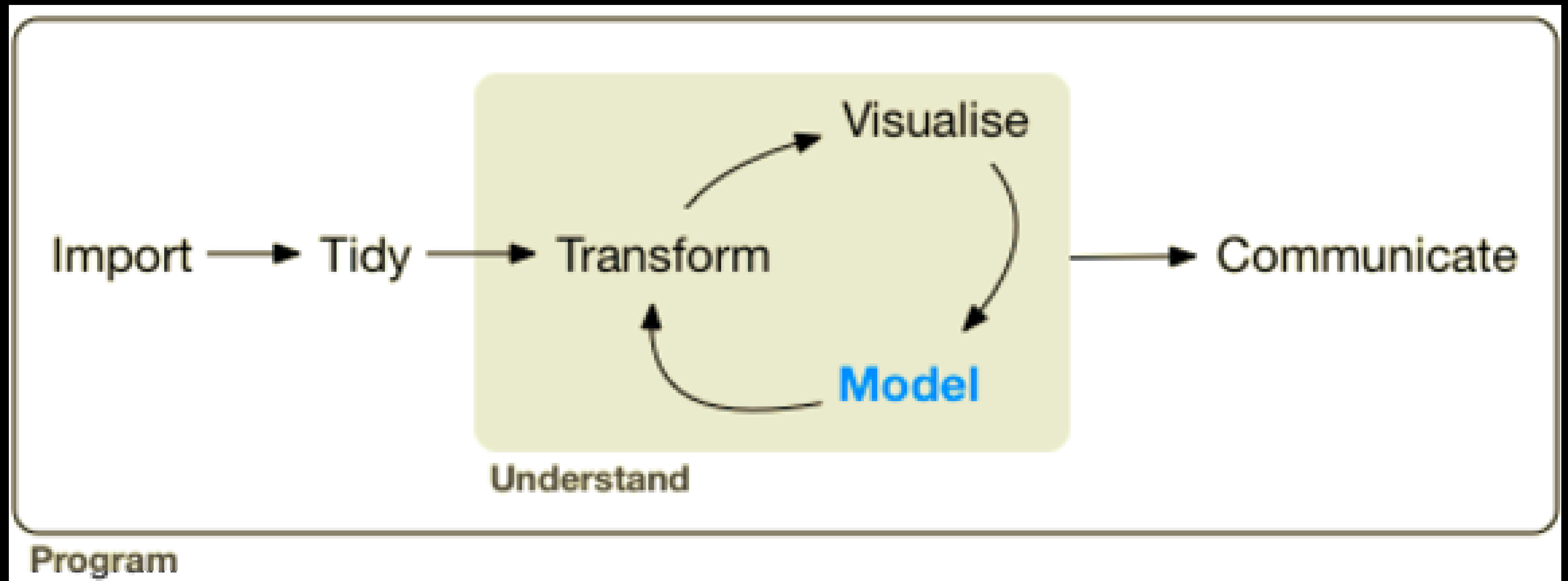
Teil 2: Synchronisation der lokalen Versionierung mit einer Remote-Versionierung und Arbeiten im Team

DATENAUFBEREITUNG



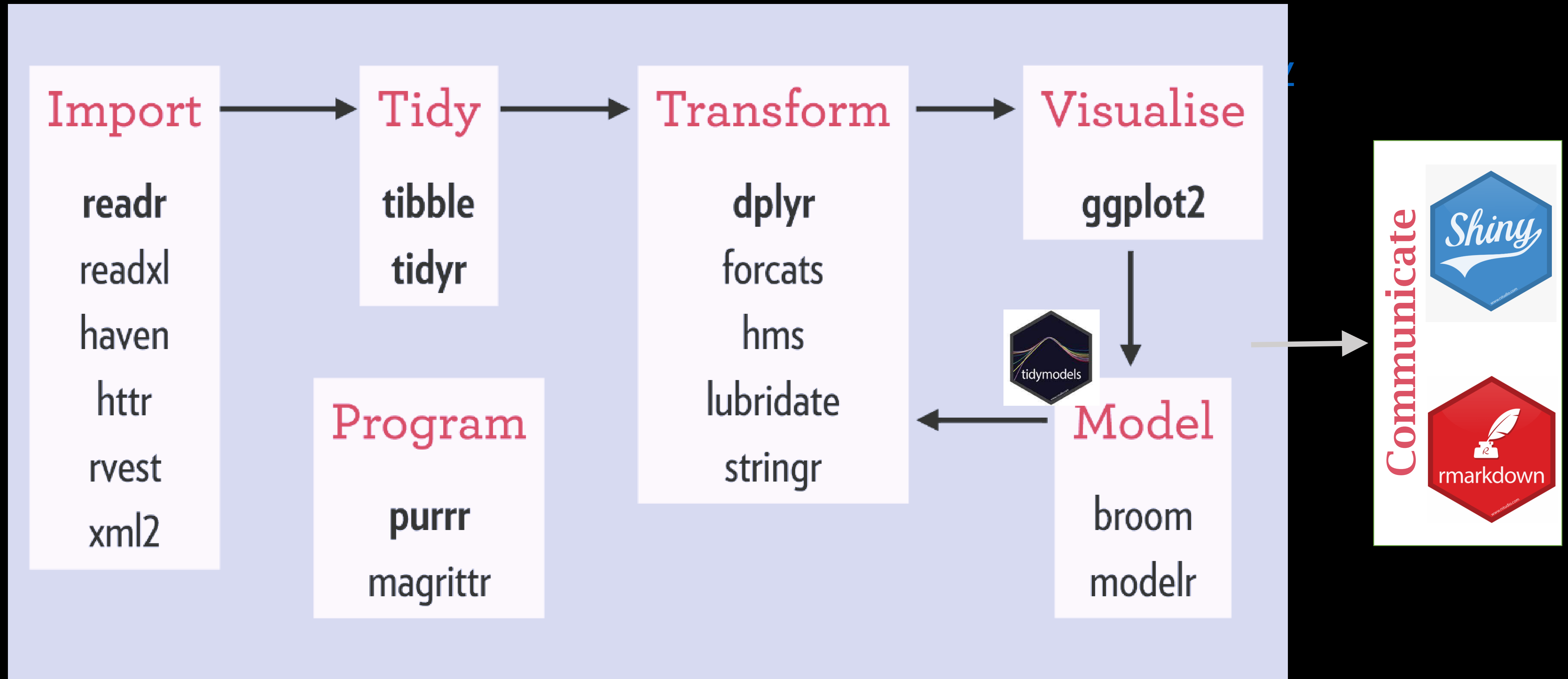
(Wickham & Grolemund, 2016)

DATENMODELLIERUNG



(Wickham & Grolemund, 2016)

TIDYVERSE



DPLYR – DATEN SELEKTIEREN & HINZUFÜGEN

select()

Variablen (Spalten) auswählen

```
mpg %>%
```

filter()

Fälle (Zeilen) auswählen

```
  select (class, hwy, cty) %>%
```

```
  filter (class=="suv") %>%
```

```
  mutate (mix = .5*hwy + .5*cty)
```

mutate()

Variablen hinzufügen

BEISPIEL 1

```
mpg %>%
```

```
  mutate(mean_milage = (cty+hwy)/2) %>%
```

```
  filter(mean_milage > 20) %>%
```

```
  select(manufacturer, mean_milage)
```

Pipe Operator: %>%

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

BEISPIEL 2

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

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)
```

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(" Vorname ", "^[\\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

- **Legt einen Account bei [GitHub](#) für Euch an**
- **Führt in RStudio unten links im Fenster Terminal (nicht verwechseln mit dem Fenster Console!) die folgenden zwei Zeilen mit Eurem GitHub-Benutzernamen und Eurer Email-Adresse aus:**

```
git config --global user.name "your_github_username"  
git config --global user.email your_email@example.com
```
- **Für eine genauere Einführung in die Möglichkeiten von Regular Expressions, schaut Euch [dieses](#) Video (11 Minuten) an.**
- **Importiert die Daten `umsatzdaten_gekuerzt.csv` und `kiwo.csv` und führt sie in einem Tibble zusammen.**
- **Committed die neu erstellte Datei in eine neue Version in Eurem Projektverzeichnis. Macht Euch die einzelnen Schritte noch einmal klar und wozu diese benötigt werden.**

ZUSÄTZLICHE RESSOURCEN

- **Coding Styleguide:**
<https://style.tidyverse.org/index.html>
- **Package um den Code zu stylen:**
<https://styler.r-lib.org/>