

Einführung in Data Science und maschinelles Lernen mit RStudio

GRAFISCHE DARSTELLUNG VON DATEN

- **Wiederholung Datenstrukturen**
- **Besprechung Aufgaben**
- **Einlesen von Daten**
- **Erstellen eines Balkendiagramms**
- **Struktur der Funktionen in ggplot**
- **Erstellen eines Balkendiagramms mit Schätzfehlern**

DATENSTRUKTUREN

Es gibt drei Grundtypen für ein Datum

- **Boolean** (TRUE / FALSE / NA)
- **Numeric** (1.1392 / NA)
- **String** ("Text" / NA)

und zusätzlich abgeleitete, speziellere Typen (integer, date, ...)

- **Integer** (**Untertyp von Numeric**; 12)
- **Date** (**Untertyp von Numeric**; "2019-04-11")
- **Factor** (**Untertyp von String**; "female"/"male")

VEKTOREN

Alle Elemente eines Vektors haben den gleichen Typ
→ Jeder Vektor hat einen eindeutigen Typ

```
v1 <- c(FALSE, TRUE, NA, TRUE)
```

```
ort <- c("kiel", "hamburg", NA, "berlin")
```

```
v2 <- c(NA, 112, 343, 235)
```

```
alter <- 12
```

MATRIZEN

Bestehen aus Vektoren gleicher Länge und gleichen Typs
→ Jede Matrix hat einen eindeutigen Typ.

```
m <- matrix(c(1,2,3,4), 2, 2)
```

entspricht: $\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$

LISTEN

Können beliebige Elemente enthalten:

- Vektoren und Matrizen unterschiedlichen Typs
- sonstige R-Objekte (etwa auch Funktionen)

→ Listen haben keinen eindeutigen Typ

```
l1 <- list(v1, alter, c(4,7,9))
```

```
l2 <- list(name="Fred", child.ages=c(4,7,9), l1)
```

Listenelemente können
benannt werden

SPEZIELLE LISTEN

DATENTABELLEN

bestehen aus Vektoren gleicher Länge
(aber potentiell unterschiedlichen Typs)

→ Datentabellen haben keinen eindeutigen Typ

- Data Frame (base Package)
- Tibble (tidyverse Package)
- Data Table (data.table Package)

SELEKTION VON SPALTEN (VARIABLEN)

Am Beispiel des Data Frames zu mtcars

Selektion genau einer Spalte als *Vektor*

- `mtcars$mpg` oder `mtcars[["mpg"]]` (**besser nicht:** `mtcars[, "mpg"]`)
- `mtcars[[1]]` (**besser nicht:** `mtcars[, 1]`)

Selektion einer oder mehrerer Spalten als *Data Frame*

- `mtcars["mpg"]` oder `mtcars[1]`
- `mtcars[c("mpg", "cyl")]` oder `mtcars[c(1, 2)]`
- **besser nicht:** `mtcars[, c("mpg", "cyl")]` oder `mtcars[, c(1, 2)]`

SELEKTION VON ZEILEN (FÄLLEN)

Selektion einer oder mehrerer Zeilen als Data Frame

- `mtcars[1,]`
- `mtcars[c(1,2,3),]` oder `mtcars[c(1:3, 5:20),]`

Löschen einer oder mehrerer Zeilen

- `mtcars[-1,]`
- `mtcars[-c(1,3),]`

→ Selektion als Vektor nicht möglich / Keine Selektion über den Namen

SELEKTION MIT BOOLESCHEMEN VEKTOREN

- **Konstruktion des Vektors**

```
mtcars$hp < 100
```

```
mtcars$gear == 5
```

- **Selektion der Fälle (Zeilen) mit dem Wert TRUE**

```
mtcars[mtcars$hp<100, ]
```

```
mtcars[mtcars$gear==5, ]
```

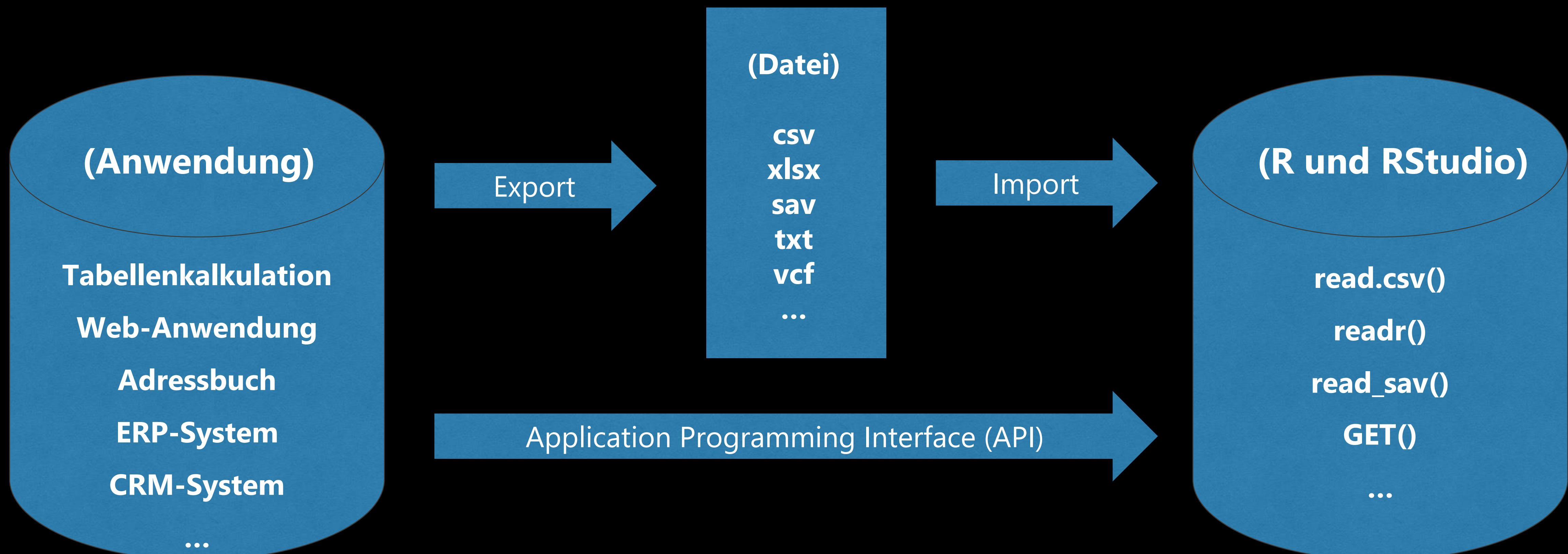
BREAKOUT

- Speichere den Datensatz `airquality` in der Variable `airQuality`.
- Berechne die Gesamtdurchschnittstemperatur.
- Berechne die Durchschnittstemperatur für den Monat Juli.
- Vergleiche, ob die Monate Juli und Mai sich signifikant in ihrer Durchschnittstemperatur unterscheiden.

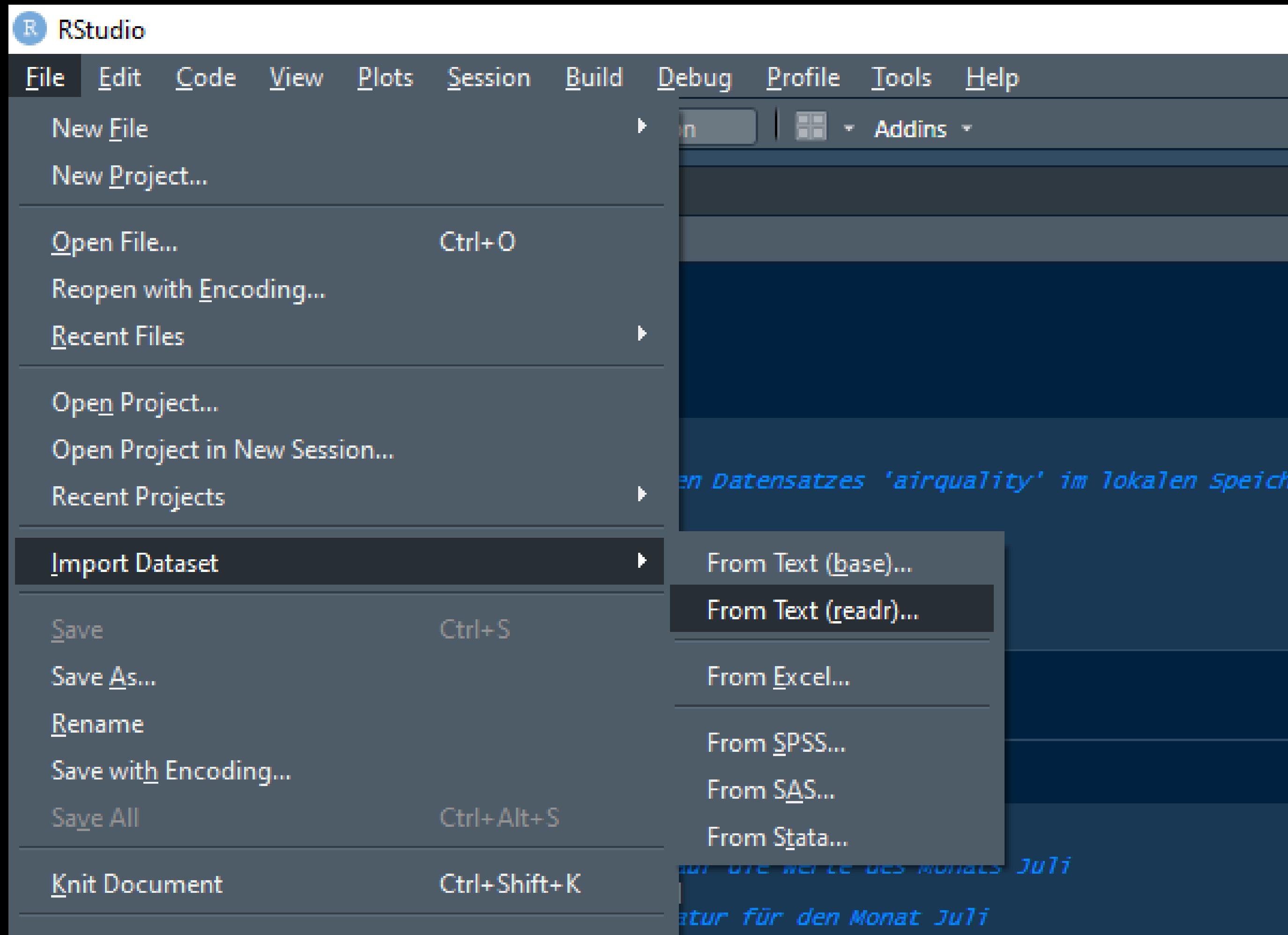
ZUWEISUNG VS. VERGLEICH

- Zuweisung von Datenobjekten:
`a <- x` (**besser nicht:** `a = x`)
- Zuweisung von Funktionsargumenten:
`mean(x, na.rm = TRUE)`
- Vergleich von Datenobjekten oder Werten:
`a == x`

IMPORT VON DATEN



IMPORT MIT HILFE VON RSTUDIO



IMPORT MIT HILFE VON RSTUDIO

Import Text Data

File/URL: [Browse...](#)

Data Preview:

Import Options:

Name: <input type="text" value="dataset"/>	<input checked="" type="checkbox"/> First Row as Names	Delimiter: <input type="button" value="Comma"/>	Escape: <input type="button" value="None"/>
Skip: <input type="text" value="0"/>	<input checked="" type="checkbox"/> Trim Spaces	Quotes: <input type="button" value="Default"/>	Comment: <input type="button" value="Default"/>
	<input checked="" type="checkbox"/> Open Data Viewer	Locale: <input type="button" value="Configure..."/>	NA: <input type="button" value="Default"/>

Code Preview:

```
library(readr)
dataset <- read_csv(NULL)
View(dataset)
```

[? Reading rectangular data using readr](#) [Import](#) [Cancel](#)

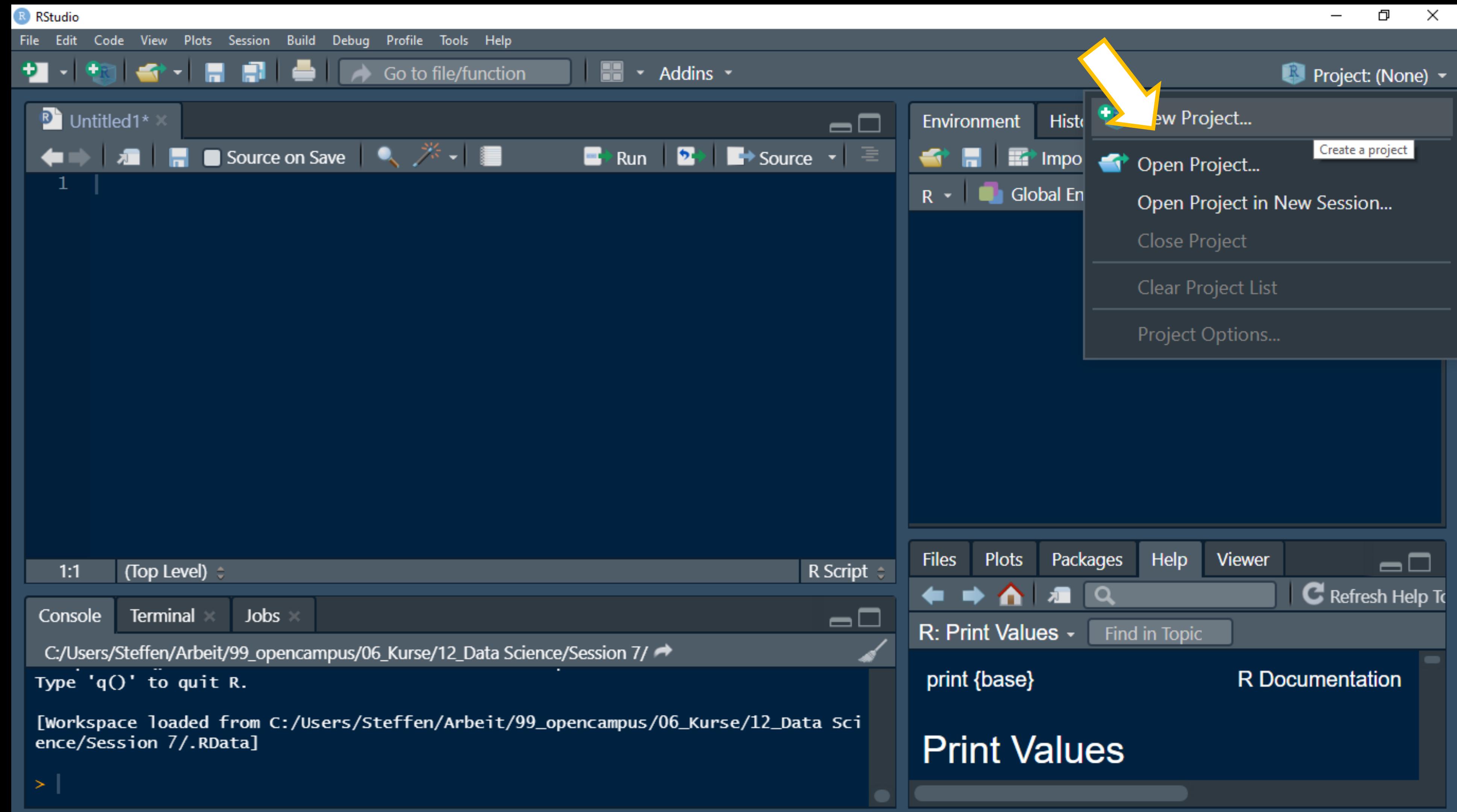
ZUSAMMENFASSUNG IMPORT

- Nutzen der Vorlage des Programmcodes aus dem RStudio Import-Aufruf

oder

- Suche im Internet:
„How to import [*Endung der Datendatei*] file into R?“

RSTUDIO-PROJEKT



BREAKOUT

- Lege mit Hilfe von RStudio ein Projektverzeichnis an.
- Lade die Dateien „kiwo.csv“, „umsatzdaten_gekuerzt.csv“ und „wetter.csv“ herunter und speichere sie in Deinem Projektverzeichnis.
Die Dateien befinden sich unter:
<https://github.com/opencampus-sh/einfuehrung-in-data-science-und-ml>
- Importiere die Datei „wetter.csv“ in RStudio.
- **Zusatzaufgabe:**
Wie importiert man eine Datei mit der Endung „.gds“?

DIAGRAMMTYPEN

The screenshot displays a collection of data visualization types arranged in a grid. The categories are:

- Distribution**: Violin, Density, Histogram, Boxplot, Ridgeline.
- Correlation**: Scatter, Heatmap, Correlogram, Bubble, Connected scatter, Density 2d.
- Ranking**: Barplot, Spider / Radar, Wordcloud, Parallel, Lollipop, Circular Barplot.
- Part of a whole**: (No icons shown)

At the top of the page, there is a navigation bar with the following items: a logo, a search icon, and links to CHART TYPES, QUICK, TOOLS, ALL, D3.JS, PYTHON, DATA TO VIZ, and ABOUT.

<https://www.r-graph-gallery.com/>

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 BASICS

- Eine ggplot Abbildung ist ein R-Objekt, das über eine beliebige Anzahl von „Layer“ 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, Scatterplot, ...)
- Jeder Layer wird durch ein „+“ hinzugefügt.

WEITERE GG PLOT 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
- **Data:** Definition eines grundlegenden Datensatzes

BEISPIEL SCATTERPLOT

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

Grundlegende Datentabelle wird für alle nachfolgenden Layer definiert.

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

Aesthetics werden für alle nachfolgenden Layer definiert.

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

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

WEITERE BEISPIELE

Scatterplot

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

Histogramm

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

Balkendiagramm

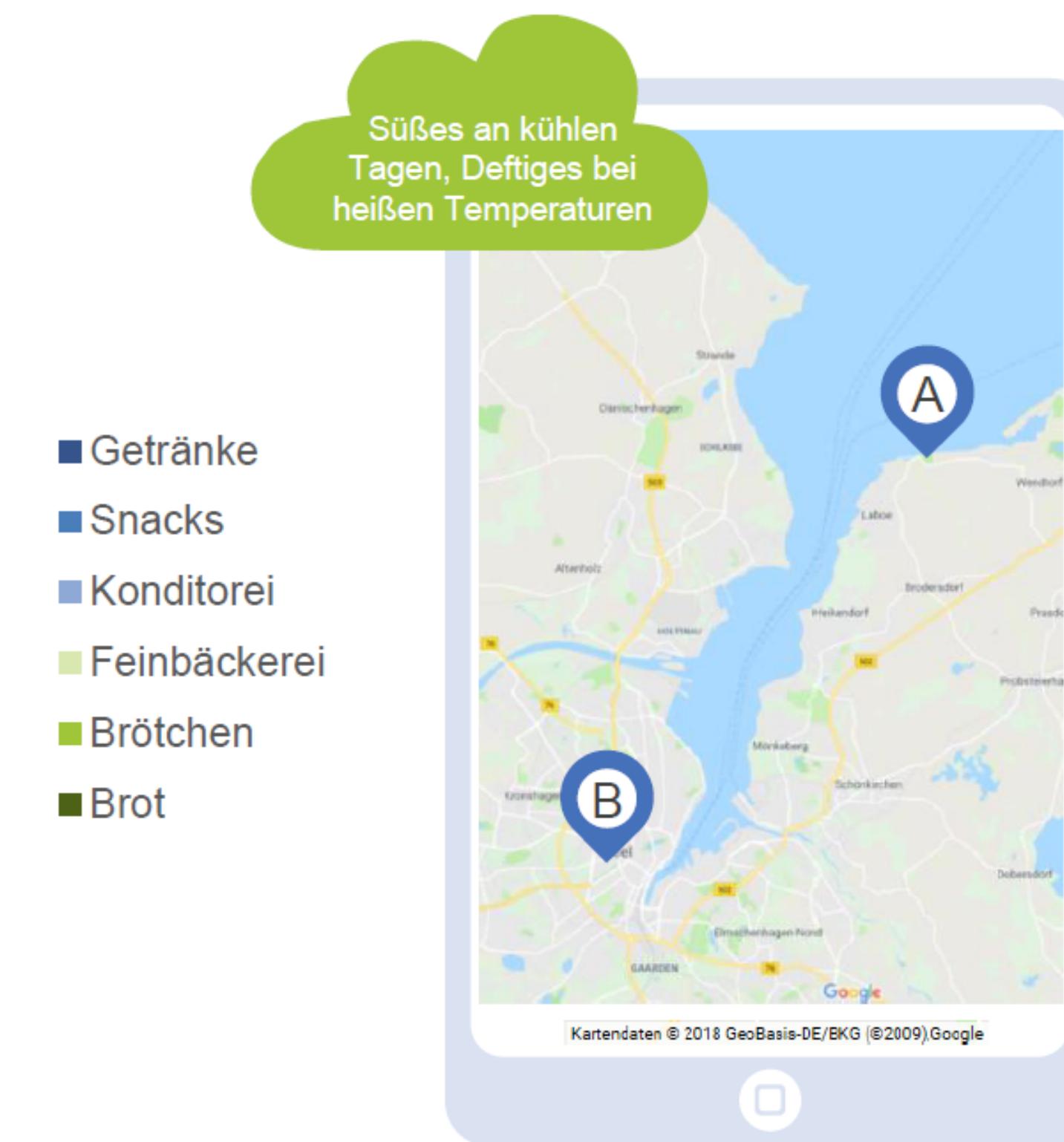
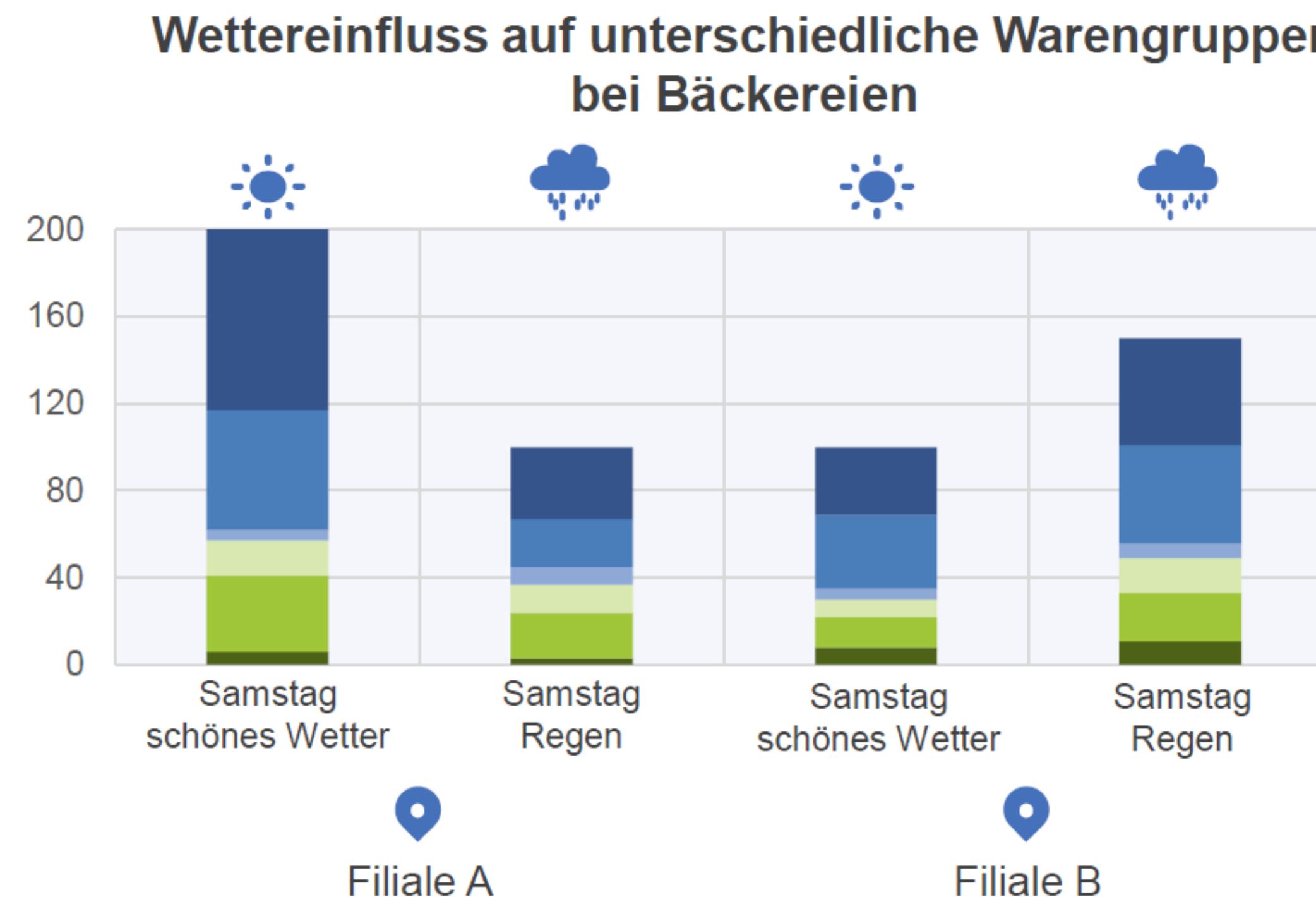
```
ggplot(mtcars)+  
  geom_bar(aes(x = as.factor(cyl), y = mpg), stat = "identity")
```

PROJEKTDATENSATZ

- Zur Verfügung gestellt von Meteolytix
- Umsatzdaten von verschiedenen Warengruppen einer Bäckereifiliale für den Zeitraum vom 01.07.2013 bis zum 06.06.2019
- Wetterdaten für den zugehörigen Zeitraum (und darüber hinaus)

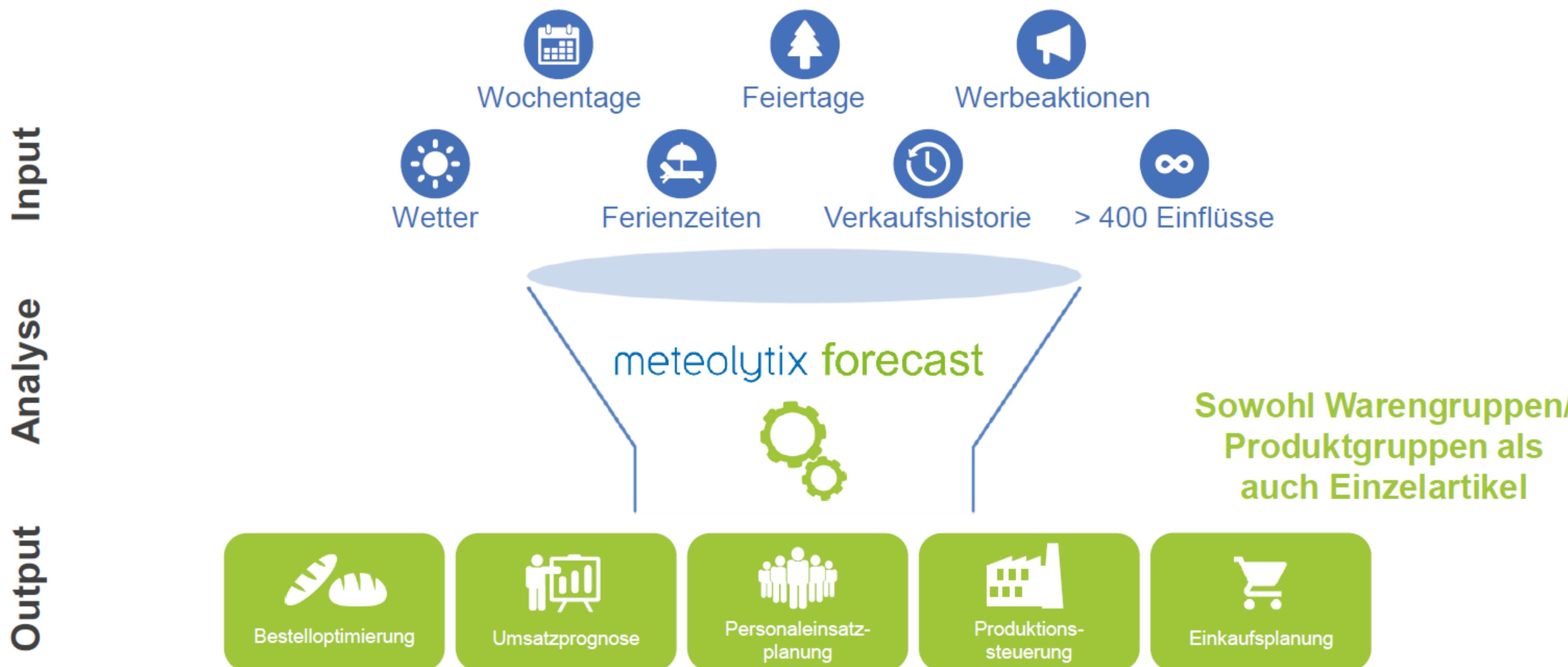
Die Stärke des Wettereffekts variiert von Ort zu Ort und wird jeweils filialindividuell berücksichtigt.

WAS WIR MACHEN



meteolytix forecast analysiert die Datenzusammenhänge von mehr als 400 Einflussfaktoren und liefert Absatzprognosen für viele Einsatzfelder.

WAS WIR MACHEN



WARENGRUPPEN

- 1 **Brot**
- 2 **Brötchen**
- 3 **Croissant**
- 4 **Konditorei**
- 5 **Kuchen**
- 6 **Saisonbrot**

WETTERDATEN

- mittlerer Bewölkungsgrad am Tag (0: min bis 8: max)
- mittlere Temperatur in Celsius
- mittlere Windgeschwindigkeit in m/s
- Wettercode
(eine Liste mit Beschreibungen gibt es z.B. hier: http://www.seewetter-kiel.de/seewetter/daten_symbole.htm)

BREAKOUT

Erstellt jeweils einmal eines der folgenden Diagrammtypen und nutzt dazu den Datensatz „wetter.csv“:

- **Scatterplot**
- **Histogramm**
- **Balkendiagramm**

ERSTELLUNG VON GGPLOTS

- 1) Auswahl eines Diagramms aus R Graph Gallery**
- 2) Ausführen des Beispielcodes**
- 3) Ersetzen der gegeben Beispieldaten durch eigene**
- 4) Anpassen mit weiteren detaillierteren Hilfe-Seiten**



Reference

Plot basics

All ggplot2 plots begin with a call to `ggplot()`, supplying default data and aesthetic mappings, specified by `aes()`. You then add layers, scales, coords and facets with `+`. To save a plot to disk, use `ggsave()`.

`ggplot()`

Create a new ggplot

`aes()`

Construct aesthetic mappings

`^+` (<gg>) `%^%``

Add components to a plot

`ggsave()`

Save a ggplot (or other grid object) with sensible defaults

`qplot() quickplot()`

Quick plot

Layers

Geoms

A layer combines data, aesthetic mapping, a geom (geometric object), a stat (statistical transformation), and a position adjustment. Typically, you will create layers using a `geom_` function, overriding the default position and stat if needed.



`geom_abline()` `geom_hline()` Reference lines: horizontal, vertical, and diagonal

`geom_vline()`

Contents

[Plot basics](#)

[Layers](#)

[Aesthetics](#)

[Scales](#)

[Guides: axes and legends](#)

[Facetting](#)

[Coordinate systems](#)

[Themes](#)

[Programming with ggplot2](#)

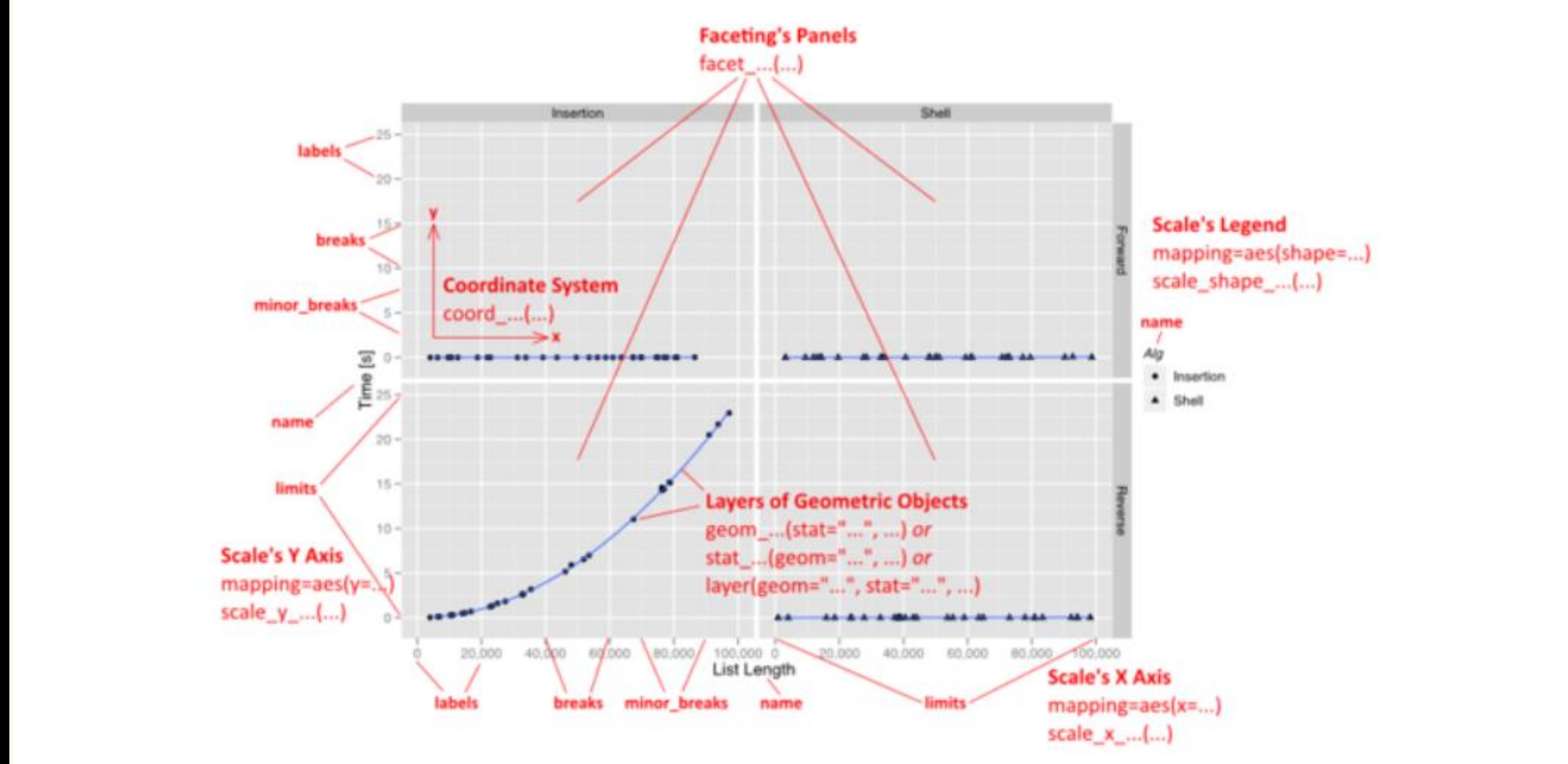
[Extending ggplot2](#)

[Vector helpers](#)

[Data](#)

[Autoplot and fortify](#)

Übersicht über existierende Layer und den Funktionen, die existieren.
<https://ggplot2.tidyverse.org/reference/>



Gute bildliche Darstellung der Elemente einer Abbildung:
<http://sape.inf.usi.ch/quick-reference/ggplot2>

Articles - R Graphics Essentials

GGPlot Cheat Sheet for Great Customization

 [kassambara](#) |  [17/11/2017](#) |  [36049](#) |  [Comments \(2\)](#) |  [R Graphics Essentials](#)

This chapter provides a cheat sheet to change the global appearance of a ggplot.

You will learn how to:

- Add title, subtitle, caption and change axis labels
- Change the appearance - color, size and face - of titles
- Set the axis limits
- Set a logarithmic axis scale
- Rotate axis text labels
- Change the legend title and position, as well, as the color and the size
- Change a ggplot theme and modify the background color
- Add a background image to a ggplot
- Use different color palettes: custom color palettes, color-blind friendly palettes, RColorBrewer palettes, viridis color palettes and scientific journal color palettes.
- Change point shapes (plotting symbols) and line types
- Rotate a ggplot
- Annotate a ggplot by adding straight lines, arrows, rectangles and text.

Contents:

- [Prerequisites](#)
- [Titles and axis labels](#)

How-To's mit verschiedenen Beispielen

www.sthda.com/english/articles/32-r-graphics-essentials/125-ggplot-cheat-sheet-for-great-customization/

Data Visualization

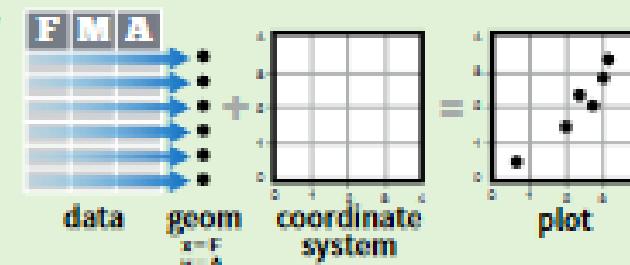
with ggplot2

Cheat Sheet

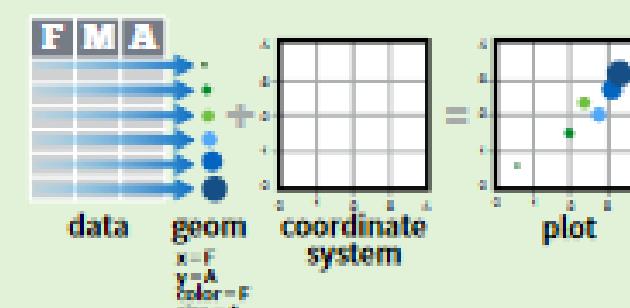


Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data** set, a set of **geoms**—visual marks that represent data points, and a **coordinate system**.



To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.



Build a graph with **qplot()** or **ggplot()**

aesthetic mappings **data** **geom**

qplot(x = cty, y = hwy, color = cyl, data = mpg, geom = "point")

Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

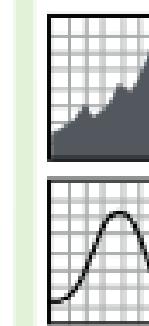
For more details see the [ggplot2 documentation](#)

Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

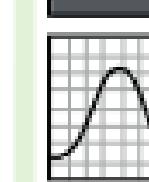
One Variable

Continuous

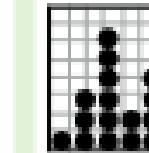
a + geom_area(stat = "bin")



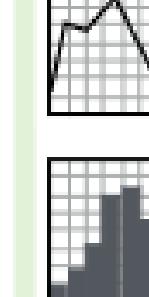
x, y, alpha, color, fill, linetype, size
b + geom_area(aes(y = ..density..), stat = "bin")



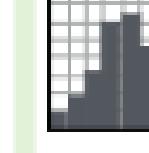
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))



x, y, alpha, color, fill
a + geom_dotplot()



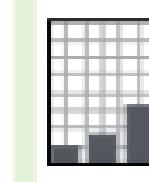
x, y, alpha, color, linetype, size
a + geom_freqpoly()



x, y, alpha, color, fill, linetype, size, weight
b + geom_histogram(binwidth = 5)

Discrete

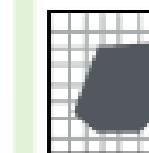
b + geom_bar()



x, alpha, color, fill, linetype, size, weight

Graphical Primitives

c + geom_polygon(aes(group = group))



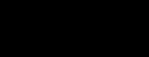
x, y, alpha, color, fill, linetype, size

d + geom_boxplot()



lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight

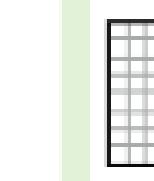
g + geom_dotplot(binaxis = "y",



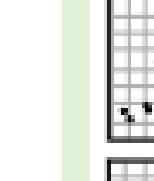
Two Variables

Continuous X, Continuous Y

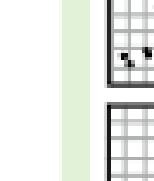
f < ggplot(mpg, aes(cty, hwy))



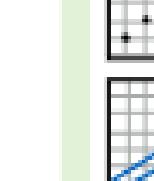
f + geom_blank()



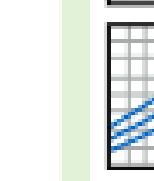
f + geom_jitter()



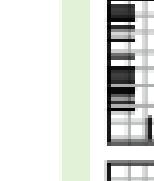
x, y, alpha, color, fill, shape, size



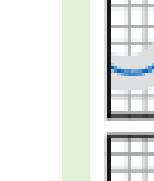
f + geom_point()



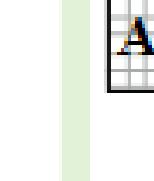
x, y, alpha, color, linetype, size, weight
f + geom_quantile()



f + geom_rug(sides = "bl")



alpha, color, linetype, size
f + geom_smooth(model = lm)

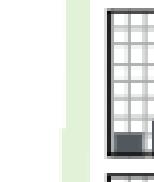


x, y, label, alpha, angle, color, family, fontface,

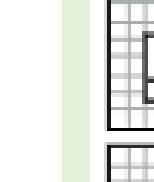
hjust, lineheight, size, vjust
f + geom_text(aes(label = cty))



C + geom_bar(stat = "identity")



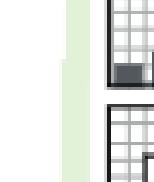
x, y, alpha, color, fill, linetype, size, weight
g + geom_boxplot()



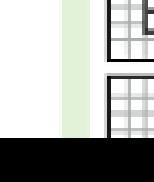
lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight
g + geom_dotplot(binaxis = "y",



g + geom_crossbar(fatten = 2)



x, y, ymax, ymin, alpha, color, fill, linetype, size
g + geom_errorbar()



x, ymin, ymax, alpha, color, linetype, size
g + geom_linerange()

Continuous Bivariate Distribution

i < ggplot(movies, aes(year, rating))



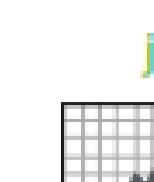
i + geom_bin2d(binwidth = c(5, 0.5))



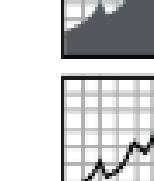
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight



i + geom_density2d()



x, y, alpha, colour, linetype, size



j + geom_step(direction = "hv")



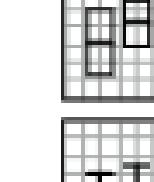
x, y, alpha, color, linetype, size

Continuous Function

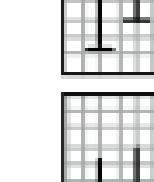
j < ggplot(economics, aes(date, unemploy))



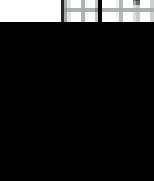
j + geom_area()



x, y, alpha, color, fill, linetype, size



j + geom_line()



x, y, alpha, color, linetype, size

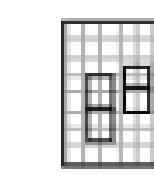
j + geom_step(direction = "hv")

x, y, alpha, color, linetype, size

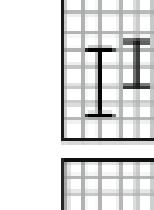
Visualizing error

df < data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)

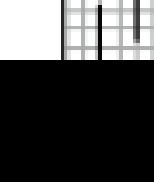
k < ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))



g + geom_crossbar(fatten = 2)



x, y, ymax, ymin, alpha, color, fill, linetype, size
g + geom_errorbar()



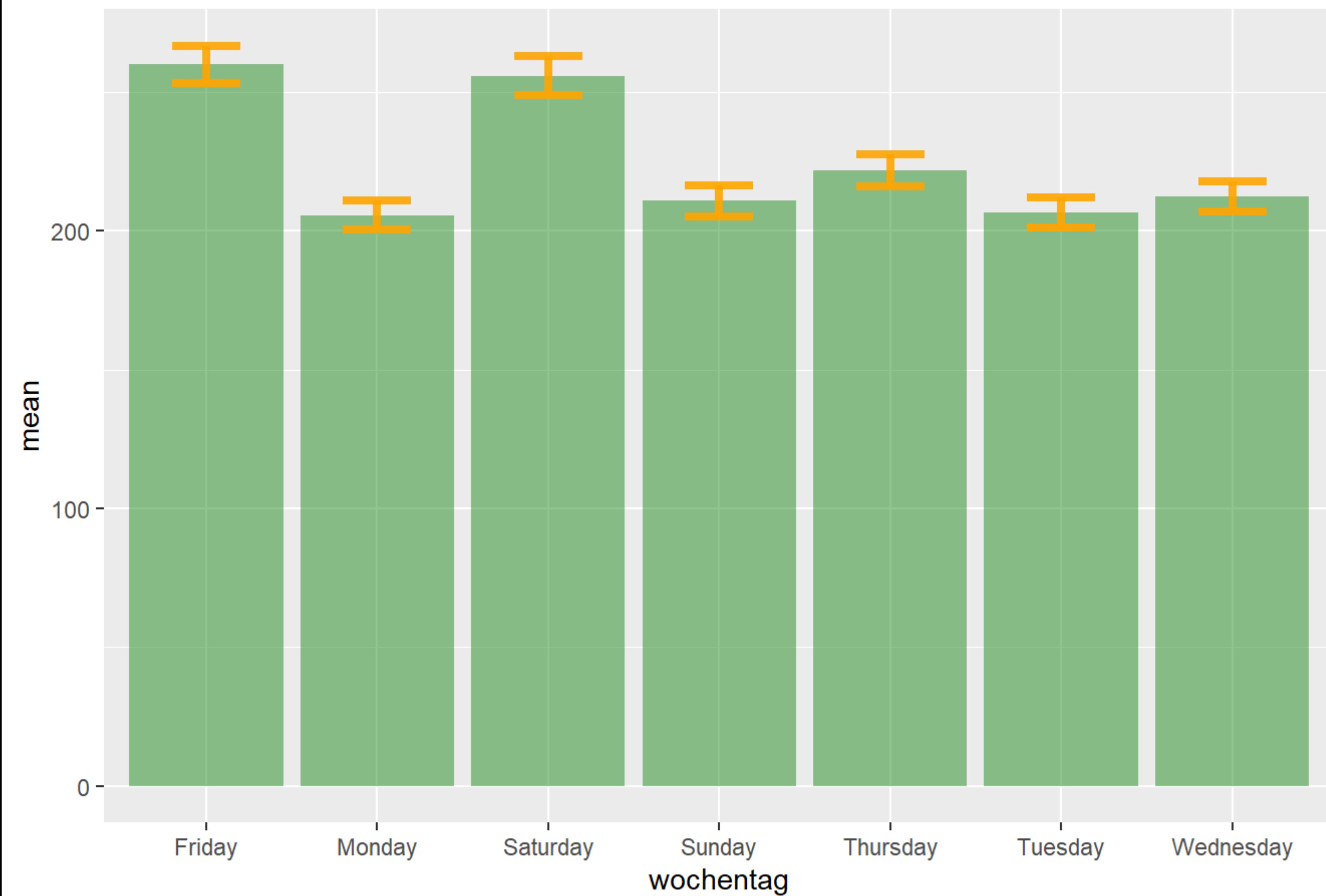
x, ymin, ymax, alpha, color, linetype, size
g + geom_linerange()

d < ggplot(economics, aes(date, unemploy))

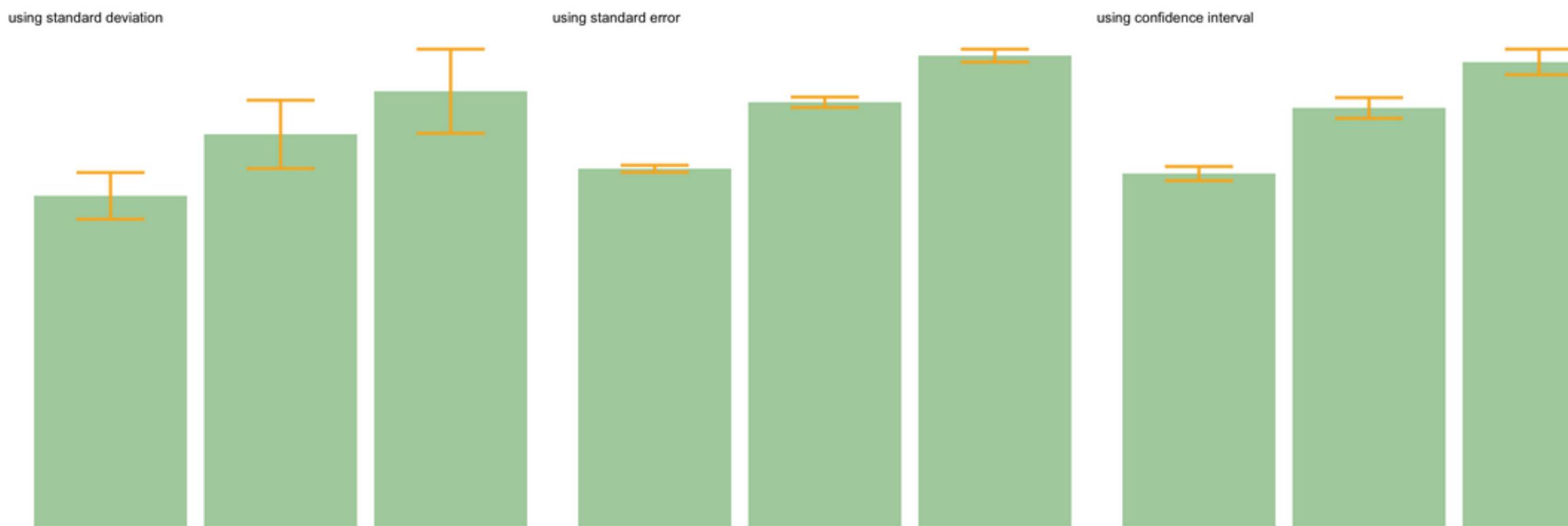
Cheat-Sheet von RStudio

<https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf>

using confidence interval



Standard deviation, Standard error or Confidence Interval?



Three different types of values are commonly used for error bars, sometimes without even specifying which one is used. It is important to understand how they are calculated, since they give very different results (see above). Let's compute them on a simple vector:

```
vec=c(1,3,5,9,38,7,2,4,9,19,19)
```

→ Standard Deviation (SD). [wiki](#)

It represents the amount of dispersion of the variable. Calculated as the root square of the variance:

```
sd <- sd(vec)
sd <- sqrt(var(vec))
```

→ Standard Error (SE). [wiki](#)

It is the standard deviation of the vector sampling distribution. Calculated as the SD divided by the square root of the sample size. By construction, SE is

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)

STARTHILFE

Import needed Libraries

```
library(readr)  
library(lubridate)  
library(ggplot2)  
library(dplyr)
```

Import turnover data

```
umsatzdaten <- read_csv("https://raw.githubusercontent.com/opencampus-sh/wise20-datascience/main/umsatzdaten_gekuerzt.csv")
```

Create variable weekday

```
umsatzdaten$wochentag <- weekdays(umsatzdaten$Datum)
```