

Statistik1

Sebastian Sauer

2024-08-29

Inhaltsverzeichnis

1. Willkommen!	3
1.1. Es geht um Ihren Lernerfolg	3
1.1.1. Lernziele	3
1.1.2. Was lerne ich hier und wozu ist das gut?	4
1.1.3. Was ist hier das Erfolgsgeheimnis?	5
1.1.4. Motivieren Sie mich!	5
1.1.5. Voraussetzungen	6
1.1.6. Überblick	6
1.2. Software	7
1.2.1. Installation	7
1.2.2. Viel R (?)	7
1.3. Zum Autor	7
1.4. Nomenklatur	7
1.4.1. Griechische Buchstaben	7
1.5. Zitation	8
1.6. Reproduzierbarkeit	8
 I. Organisatorisches	 9
 II. Modellieren	 10
2. Geradenmodelle 2	11
2.1. Lernsteuerung	11
2.1.1. Standort im Lernpfad	11
2.1.2. Lernziele	11
2.1.3. Benötigte R-Pakete	11
2.1.4. Benötigte Daten	11
2.2. Forschungsbezug: Gläserne Kunden	12
2.3. Wetter in Deutschland	12
2.3.1. metrische UV	14
2.3.2. UV zentrieren	15
2.3.3. Binäre UV	18
2.3.4. Nominate UV	22
2.3.5. Binäre plus metrische UV	26
2.3.6. Interaktion	30

2.4.	Modelle mit vielen UV	32
2.4.1.	Zwei metrische UV	32
2.4.2.	Viele UV ins Modell?	33
2.5.	Fallbeispiel zur Prognose	34
2.5.1.	Modell “all-in”	34
2.5.2.	Modell “all-in”, ohne Titelspalte	36
2.6.	Vertiefung: Das Aufteilen Ihrer Daten	39
2.6.1.	Analyse- und Assessment-Sample	39
2.6.2.	Train- vs. Test-Sample	40
2.7.	Praxisbezug	41
2.8.	Wie man mit Statistik lügt	41
2.8.1.	Pinguine drehen durch	41
2.8.2.	Analyse 1: Gesamtdaten	42
2.8.3.	Analyse 2: Aufteilung in Arten (Gruppen)	43
2.8.4.	Vorsicht bei der Interpretation von Regressionskoeffizienten	44
2.9.	Fazit	45
2.10.	Fallstudien	46
2.10.1.	New Yorker Flugverspätungen 2023	46
2.10.2.	Filmerlöse	46
2.11.	Vertiefung	46
2.12.	Aufgaben	47
2.13.	Literaturhinweise	47
	Literatur	47

1. Willkommen!

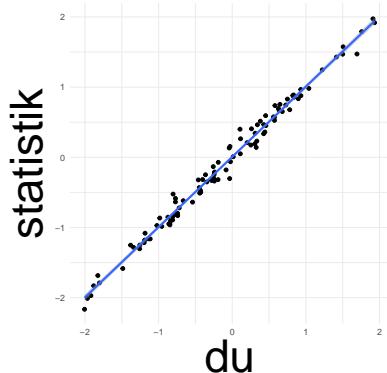


Abbildung 1.1.: Statistik und Du: Guter Fit!

1.1. Es geht um Ihren Lernerfolg

Meister Yoda rät: Lesen Sie die Hinweise (Abbildung 1.2).

Quelle: [Imgflip Memengenerator](#)

1.1.1. Lernziele

- Die Studentis sind mit wesentlichen Methoden der explorativen Datenanalyse vertraut und können diese selbstständig anwenden.
- Die Studentis können gängige Forschungsfragen in lineare Modelle übersetzen, diese auf echte Datensätze anwenden und die Ergebnisse interpretieren.

Kurz gesagt: Das ist ein Grundkurs in Daten zähmen.

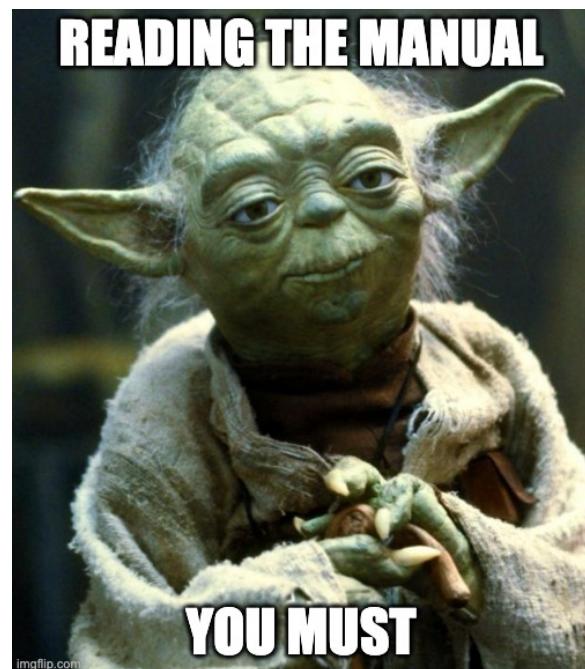


Abbildung 1.2.: Lesen Sie die folgenden Hinweise im eigenen Interesse

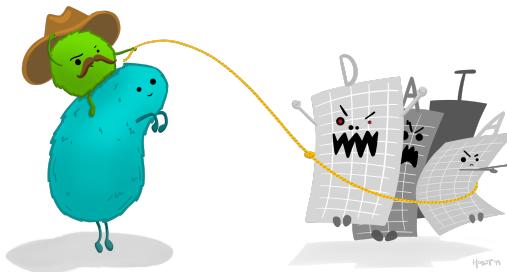


Abbildung 1.3.: Daten zähmen

Bildquelle: Allison Horst, CC-BY

1.1.2. Was lerne ich hier und wozu ist das gut?

Was lerne ich hier?

Sie lernen das *Handwerk der Datenanalyse* mit einem Schwerpunkt auf Vorhersage. Anders gesagt: Sie lernen, *Daten aufzubereiten* und aus Daten *Vorhersagen* abzuleiten. Zum Beispiel: Kommt ein Student zu Ihnen und sagt "Ich habe 42 Stunden für die Klausur gelernt, welche Note kann ich in der Klausur erwarten?". Darauf Ihre Antwort: "Auf Basis meiner Daten und meines Modells müsstest du

eine 2.7 schreiben!”¹. Außerdem lernen Sie, wie man die Güte einer Vorhersage auf Stichhaltigkeit prüft. Denn Vorhersagen kann man ja in jeder Eckkneipe oder beim Wahrsager bekommen. Wir wollen aber belastbare Vorhersagen und zumindest wissen, wie gut die Vorhersagen (von jemanden) bisher waren.

Warum ist das wichtig?

Wir wollen nicht auf Leuten vertrauen, die behaupten, sie wüssten, was für uns richtig und gut ist. Wir wollen selber die Fakten prüfen können.

Wozu brauche ich das im Job?

Datenanalyse spielt bereits heute in vielen Berufen eine Rolle. Tendenz stark zunehmend.

Wozu brauche ich das im weiterem Studium?

In Forschungsarbeiten (wie in empirischen Forschungsprojekten, etwa in der Abschlussarbeit) ist es üblich, statistische Ergebnisse hinsichtlich quantitativ zu analysieren.

Ist Statistik nicht sehr abstrakt?

Der Schwerpunkt dieses Kurses liegt auf Anwenden und Tun; ähnlich dem Erlernen eines Handwerks. Theorien und Abstraktionen stehen nur am Rand.

Gibt es auch gute Jobs, wenn man sich mit Daten auskennt?

Das Forum (2020) berichtet zu den “Top 20 job roles in increasing and decreasing demand across industries” (S. 30, Abb. 22):

1. Data Analysts und Scientists
2. AI and Machine Learning Specialists
3. Big Data Specialists

1.1.3. Was ist hier das Erfolgsgeheimnis?

! Wichtig

Dran bleiben ist der Schlüssel zum Erfolg. Üben Sie regelmäßig. Geben Sie bei Schwierigkeiten nicht auf.



1.1.4. Motivieren Sie mich!

Schauen Sie sich das Video mit einer [Ansprache zur Motivation](#) an.²

¹Darauf dis Studenti: “Hpmf.”

²<https://youtu.be/jtNlzpcPr5Y>

1.1.5. Voraussetzungen

Um von diesem Kurs am besten zu profitieren, sollten Sie Folgendes mitbringen:

- Bereitschaft, Neues zu lernen
- Bereitschaft, nicht gleich aufzugeben
- Kenntnis grundlegender Methoden wissenschaftlichen Arbeitens

Was Sie *nicht* brauchen, sind besondere Mathe-Vorkenntnisse.

1.1.6. Überblick

Abb. Abbildung 1.4 gibt einen Überblick über den Verlauf und die Inhalte des Buches. Das Diagramm hilft Ihnen zu verorten, wo welches Thema im Gesamtzusammenhang steht.

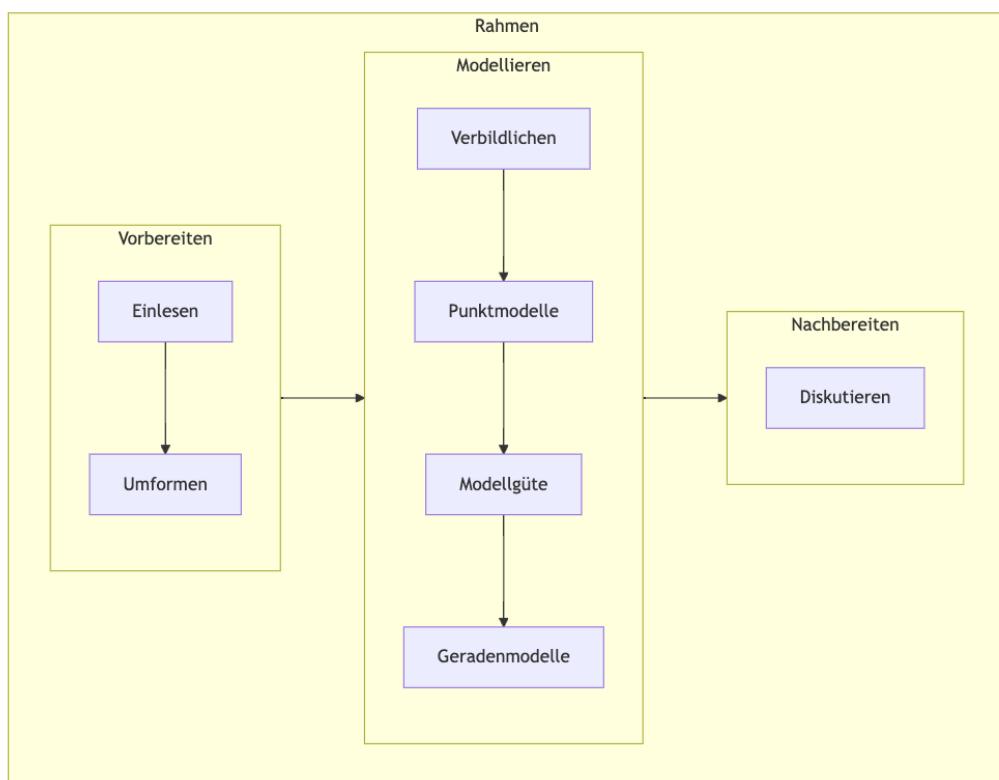


Abbildung 1.4.: Überblick über den Inhalt und Verlauf des Buches

Das Diagramm zeigt den Ablauf einer typischen Datenanalyse. Natürlich kann man sich auch andere sinnvolle Darstellungen dieses Ablaufs vorstellen.

1.2. Software

Sie benötigen R, RStudio und einige R-Pakete für diesen Kurs.

1.2.1. Installation

Hier finden Sie *Installationshinweise*.³

1.2.2. Viel R (?)

Dieses Buch enthält “mittel” viel R. Auf fortgeschrittene R-Techniken wurde aber komplett verzichtet. Dem einen oder der anderen Anfänger:in mag es dennoch “viel Code” erscheinen. Es wäre ja auch möglich gewesen, auf R zu verzichten und stattdessen eine “Klick-Software” zu verwenden. **JASP** oder **Jamovi** sind Beispiele für tolle Software aus dieser Kategorie. Ich glaube aber, der Verzicht auf eine Skriptsprache (R) wäre ein schlechter Dienst an den Studentis. Mit Blick auf eine “High-Tech-Zukunft” sollte man zumindest mit etwas Computer-Code vertraut sein. Auf Computercode zu verzichten erschien mir daher fahrlässig für die “Zukunftsfestigkeit” der Ausbildung.

1.3. Zum Autor

Nähere Hinweise zum Autor dieses Buch, Sebastian Sauer, finden Sie [hier](#).⁴ Dort gibt es auch einen Überblick über [weitere Bücher des Autors zum Themenkreis Datenanalyse](#).⁵

1.4. Nomenklatur

1.4.1. Griechische Buchstaben

In diesem Buch werden ein paar (wenige) griechische Buchstaben verwendet, die in der Statistik üblich sind. Häufig werden *griechische* Buchstaben verwendet, um eine Grundgesamtheit (Population) zu beschreiben (die meistens unbekannt ist). Lateinische (“normale”) Buchstaben werden demgegenüber verwendet, um eine Stichprobe (Datensatz, vorliegende Daten) zu beschreiben. Tabelle 1.1 stellt diese Buchstaben zusammen mit ihrer Aussprache und Bedeutung vor.

Tabelle 1.1.: Griechische Buchstaben, die in diesem Buch verwendet werden.

Zeichen	Aussprache	Buchstabe	Bedeutung in der Statistik
β	beta	b	Regressionskoeffizient

³<https://hinweisbuch.netlify.app/hinweise-software>

⁴<https://sebastiansauer-academic.netlify.app/>

⁵<https://sebastiansauer-academic.netlify.app/#ebooks>

Zeichen	Aussprache	Buchstabe	Bedeutung in der Statistik
μ	mü	m	Mittelwert
σ	sigma	s	Streuung
Σ	Sigma	S	Summenzeichen
ρ	rho	r	Korrelation (nach Pearson)

Mehr griechische Buchstaben finden sich [z.B. in Wikipedia](#).⁶

1.5. Zitation

Bitte zitieren Sie dieses Buch wie folgt:

Sauer, S. (2024). *Statistik1*. <https://statistik1.netlify.app/>

Hier sind die maschinenlesbaren Zitationsinfos (Bibtex-Format), die Sie in Ihre Literatursoftware importieren können:

```
@book{sauer_statistik1,
  title = {Statistik1},
  rights = {CC-BY-NC},
  url = {https://statistik1.netlify.app/},
  author = {Sauer, Sebastian},
  date = {2024},
}
```

Hier ist die DOI:

[10.5281/zenodo.10082517](https://doi.org/10.5281/zenodo.10082517)

1.6. Reproduzierbarkeit

Die verwendeten R-Pakete sind mit [renv](#) dokumentiert.⁷

Der Quellcode ist [in diesem Github-Repo](#) dokumentiert.⁸

Dieses Dokument wurde erzeugt am/um: 2024-08-29 09:33:27.

⁶https://de.wikipedia.org/wiki/Griechisches_Alphabet

⁷<https://rstudio.github.io/renv/index.html>

⁸<https://github.com/sebastiansauer/statistik1>

Teil I.

Organisatorisches

Teil II.

Modellieren

2. Geradenmodelle 2

2.1. Lernsteuerung

2.1.1. Standort im Lernpfad

Abb. Abbildung 1.4 zeigt den Standort dieses Kapitels im Lernpfad und gibt damit einen Überblick über das Thema dieses Kapitels im Kontext aller Kapitel.

2.1.2. Lernziele

- Sie können Regressionsmodelle für Forschungsfragen mit binärer, nominaler und metrischer UV erläutern und in R anwenden.
- Sie können Interaktionseffekte in Regressionsmodellen erläutern und in R anwenden.
- Sie können den Anwendungszweck von Zentrieren und z-Transformationen zur besseren Interpretation von Regressionsmodellen erläutern und in R anwenden.
- Sie können Modelle nutzen, um Vorhersagen anhand neuer Daten zu erstellen.

2.1.3. Benötigte R-Pakete

```
library(tidyverse)
library(yardstick)  # für Modellgüte im Test-Sample
library(easystats)
library(ggpubr)    # Daten visualisieren
library(openintro) # dataset mariokart
```

2.1.4. Benötigte Daten

?@lst-mario-path definiert den Pfad zum Datensatz mariokart und importiert die zugehörige CSV-Datei in R, so dass wir einen Tibble mit Namen mariokart erhalten.

```

mariokart_path <- paste0(
  "https://vincentarelbundock.github.io/Rdatasets/",
  "csv/openintro/mariokart.csv")
mariokart <- read.csv(mariokart_path)

wetter_path <- paste0(
  "https://raw.githubusercontent.com/sebastiansauer/",
  "Lehre/main/data/wetter-dwd/precip_temp_DWD.csv")
wetter <- read.csv(wetter_path)

```

Die Wetterdaten stammen vom [DWD](#).¹

2.2. Forschungsbezug: Gläserne Kunden

Lineare Modelle² sind ein altes, aber mächtiges Werkzeug. Sie gehören immernoch zum Standard-Repertoire moderner Analystis.

Beispiel 2.1 (Wie gut kann man Ihre Persönlichkeit auf Basis des Facebook-Profil vorhersagen?). In einer Studie mit viel Medienresonanz untersuchten ([Kosinski2013?](#)), wie gut Persönlichkeitszüge durch Facebook-Daten (Likes etc.) vorhergesagt werden können. Die Autoren resümieren:

We show that easily accessible digital records of behavior, Facebook Likes, can be used to automatically and accurately predict a range of highly sensitive personal attributes including: sexual orientation, ethnicity, religious and political views, personality traits, intelligence, happiness, use of addictive substances, parental separation, age, and gender.

Die Autoren berichten über hohe Modellgüte (r) zwischen den tatsächlichen persönlichen Attributen und den vorhergesagten Werten Ihres Modells, s. Abbildung 2.1. Das eingesetzte statistische Modell beruht auf einem linearen Modell, also ähnlich zu dem in diesem Kapitel vorgestellten Methoden.

Neben der analytischen Stärke der Regressionsanalyse zeigt das Beispiel auch, wie gläsern Konsument:innen im Internet sind.□

2.3. Wetter in Deutschland

Beispiel 2.2 (Wetterdaten). Nachdem Sie einige Zeit als Datenanalyst bei dem Online-Auktionshaus gearbeitet haben, stand Ihnen der Sinn nach etwas Abwechslung. Viel Geld verdienen und Ruhm und Anerkennung sind ja schon ganz nett, aber dann fiel Ihnen ein, dass Sie ja zu Generation Z gehören, und daher den schnöden Mammon nicht so hoch schätzen sollten. Sie entschließen sich,

¹Lizenzhinweis: Datenbasis: Deutscher Wetterdienst, eigene Elemente ergänzt.

²synonym: Regressionsanalysen

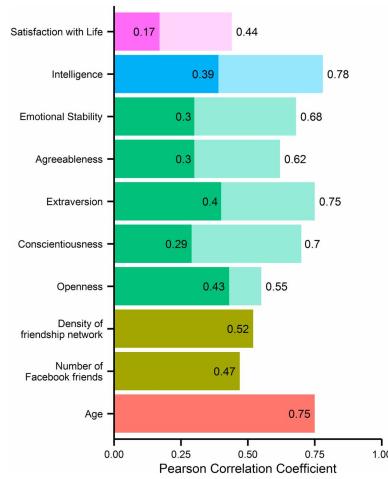


Abbildung 2.1.: Prediction accuracy of regression for numeric attributes and traits expressed by the Pearson correlation coefficient between predicted and actual attribute values

Ihre hochgeschätzten Analyse-Skills für etwas einzusetzen, das Ihnen sinnvoll erscheint: Die Analyse des Klimawandels.

Beim [Deutschen Wetterdienst, DWD](#) haben Sie sich Wetterdaten von Deutschland heruntergeladen. Nach etwas [Datenjudo, auf das wir hier nicht eingehen wollen](#) resultiert ein schöner Datensatz, den Sie jetzt analysieren wollen³:

```
wetter_path <- paste0(
  "https://raw.githubusercontent.com/sebastiansauer/",
  "Lehre/main/data/wetter-dwd/precip_temp_DWD.csv")

wetter <- read.csv(wetter_path)
```

Ein *Data-Dictionary* für den Datensatz können Sie [hier](#) herunterladen.⁴

i Hinweis

Ein *Data-Dictionary* (Codebook) erklärt einen Datensatz. Oft bedeutet das, dass für jede Spalte der Datentabelle erklärt wird, was die Spalte bedeutet.□

Hervorragend! An die Arbeit!

³Temperatur: Grad Celcius, Niederschlag (precip) mm Niederschlag pro Quadratmeter

⁴<https://raw.githubusercontent.com/sebastiansauer/Lehre/main/data/wetter-dwd/wetter-dwd-data-dict.md>

2.3.1. metrische UV

2.3.1.1. Modell Wetter1

Sie stellen sich nun folgende Forschungsfrage:

- 💡 Um wieviel ist die Temperatur in Deutschland pro Jahr gestiegen, wenn man die letzten ca. 100 Jahre betrachtet?

Die Modellparameter von `lm_wetter1` sind in Tabelle 2.1 zu sehen.

```
lm_wetter1 <- lm(temp ~ year, data = wetter)
parameters(lm_wetter1)
```

Tabelle 2.1.: Modellparameter von `lm_wetter1`

Parameter	Coefficient	SE	95% CI	t(28864)	p
(Intercept)	-14.25	1.85	(-17.87, -10.63)	-7.71	< .001
year	0.01	9.47e-04	(9.80e-03, 0.01)	12.30	< .001

Laut Ihrem Modell wurde es pro Jahr um 0.01 Grad wärmer, pro Jahrzehnt also 0.1 und pro Jahrhundert 1 Grad.

- 💡 Das ist sicherlich nicht linear! Vermutlich ist die Temperatur bis 1950 konstant geblieben und jetzt knallt sie durch die Decke!

- 💡 Mit der Ruhe, das schauen Sie sich später an.

2.3.1.2. Punkt- vs. Bereichsschätzung

In `tbl-lm-wetter1` finden sich zwei Arten von Information für den Wert des Achsenabschnitts (`b0`) und des Regressionsgewichts von `year`(`b1`):

1. *Punktschätzungen* In der Spalte `Coefficient` sehen Sie den “Best-Guess” für den entsprechenden Koeffizienten in der Population. Das ist sozusagen der Wert für den sich das Modell festlegen würde, wenn es sonst nichts sagen dürfte.
2. *Bereichsschätzungen* Cleverer als Punktschätzungen sind Bereichsschätzungen (Intervallschätzungen): Hier wird ein Bereich plausibler Werte für den entsprechenden Wert angegeben. Der “Bereich plausibler Werte” wird auch als Konfidenzintervall (engl. confidence interval, CI) bezeichnet. Entsprechend gibt `CI_low` die Untergrenze des Bereichs plausibler Werte und `CI_high` die Obergrenze aus. So können wir ablesen, dass das Regressionsgewicht von `year` irgendwo zwischen praktisch Null (0.009) und ca. 0.01 Grad geschätzt wird.

- Merke: Je schmäler das Konfidenzintervall, desto genauer wird der Effekt geschätzt.

2.3.1.3. Modell Wetter1a

Das Modell `lm_wetter1`, bzw. die Schätzungen zu den erwarteten Werten, kann mich sich so ausgeben lassen, s. Abbildung 2.2, links. Allerdings sind das zu viele Datenpunkte. Wir sollten es vielleicht anders visualisieren, s. Abbildung 2.2, rechts. Dazu aggregieren wir die Messwerte eines Jahres zu jeweils einem Mittelwert.

```
wetter_summ <-
  wider %>%
  group_by(year) %>%
  summarise(temp = mean(temp),
            precip = mean(precip)) # precipitation: engl. für
                           ↴ Niederschlag
```

Auf dieser Basis erstellen wir ein neues lineares Modell, s. Tabelle 2.2.

```
lm_wetter1a <- lm(temp ~ year, data = wider)
parameters(lm_wetter1a)
```

Tabelle 2.2.: Modellparameter von `lm_wetter1a`

Parameter	Coefficient	SE	95% CI	t(140)	p
(Intercept)	-14.14	2.70	(-19.48, -8.79)	-5.23	< .001
year	0.01	1.38e-03	(8.86e-03, 0.01)	8.38	< .001

```
plot(estimate_relation(lm_wetter1))
plot(estimate_relation(lm_wetter1a))
```

⚠️ Moment mal, der Achsenabschnitt liegt bei -15 Grad! Was soll das bitte bedeuten?

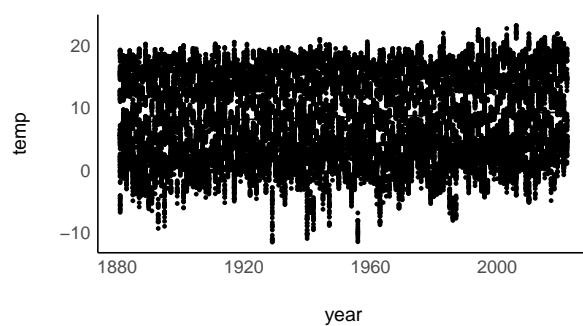
2.3.2. UV zentrieren

Zur Erinnerung: Der Achsenabschnitt (β_0 ; engl. *intercept*) ist definiert als der Y-Wert an der Stelle $X=0$, s. [?@sec-interpret-reg-mod](#).

In den Wetterdaten wäre Jahr=0 Christi Geburt. Da unsere Wetteraufzeichnung gerade mal ca. 150 Jahre in die Vergangenheit reicht, ist es vollkommen vermessen, dass Modell 2000 Jahre in die Vergangenheit zu extraplieren, ganz ohne dass wir dafür Daten haben, s. Abbildung 2.3.

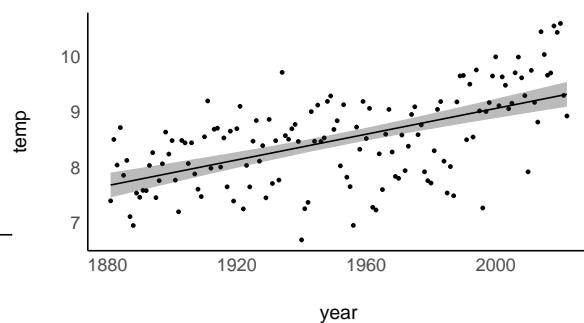
Sinnvoller ist es da, z.B. einen *Referenzwert* festzulegen, etwa 1950. Wenn wir dann von allen Jahren 1950 abziehen, wird das Jahr 1950 zum neuen Jahr Null. Damit bezöge sich der Achsenabschnitt auf das Jahr 1950, was Sinn macht, denn für dieses Jahr haben wir Daten.

Predicted response (temp ~ year)



(a) Jeder Punkt ist ein Tag (viel Overplotting, wenig nützlich)

Predicted response (temp ~ year)



(b) Jeder Punkt ist ein Jahr (wetter_summ)

Abbildung 2.2.: Die Veränderung der mittleren Temperatur in Deutschland im Zeitverlauf (Datenquelle: DWD)

MY HOBBY: EXTRAPOLATING

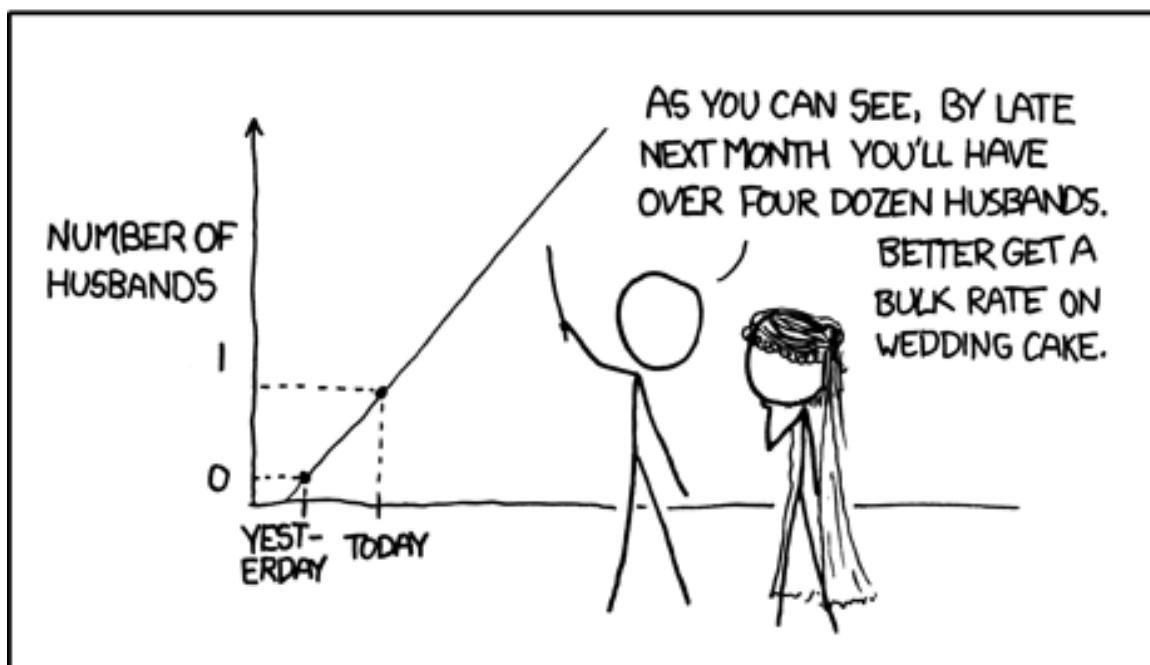


Abbildung 2.3.: Du sollst nicht ein Modell weit außerhalb seines Datenbereichs extrapoliieren

Hat man nicht einen bestimmten Wert, der sich als Referenzwert anbietet, so ist es üblich, z.B. den Mittelwert (der UV) als Referenzwert zu nehmen. Diese Transformation bezeichnet man als *Zentrierung* (engl. centering) der Daten.

So zentriert man eine Verteilung:

```
wetter <-
wetter %>%
mutate(year_c = year - mean(year)) # "c" wie centered
```

Das mittlere Jahr in unserer Messwertreihe ist übrigens 1951:

```
wetter %>%
summarise(mean(year))
```

$$\begin{array}{c} \hline \text{mean(year)} \\ \hline 1951.251 \\ \hline \end{array}$$

Die Steigung (d.h. der Regressionskoeffizient für `year_c`) bleibt unverändert, nur der Achsenabschnitt ändert sich, s. Tabelle 2.4.

```
lm_wetter1_zentriert <- lm(temp ~ year_c, data = wetter)
parameters(lm_wetter1_zentriert)
```

Tabelle 2.4.: Modellparameter von `lm_wetter1_zentriert`

Parameter	Coefficient	SE	95% CI	t(28864)	p
(Intercept)	8.49	0.04	(8.42, 8.57)	219.43	< .001
year c	0.01	9.47e-04	(9.80e-03, 0.01)	12.30	< .001

Jetzt ist die Interpretation des Achsenabschnitts komfortabel: Im Jahr 1951 ($x=0$) lag die mittlere Temperatur in Deutschland (laut DWD) bei ca. 8.5 Grad Celcius. Die Regressionsgleichung lautet: $\text{temp_pred} = 8.49 + 0.01 * \text{year_c}$. In Worten: Wir sagen eine Temperatur vorher, die sich als Summe von 8.49 Grad plus 0.01 mal das Jahr (in zentrierter Form) berechnet.

! Referenzwert entspricht Null

Der Referenzwert bzw. der Wert der Referenzgruppe entspricht dem Y-Wert bei $x=0$ im Regressionsmodell.□

Wie gut erklärt unser Modell die Daten?

```
r2(lm_wetter1_zentriert) # aus `{easystats}`  
## # R2 for Linear Regression  
##      R2: 0.005  
## adj. R2: 0.005
```

Viel Varianz des Wetters erklärt das Modell mit `year_c`⁵ aber nicht. Macht auch Sinn: Abgesehen von der Jahreszahl spielt z.B. die Jahreszeit eine große Rolle für die Temperatur. Das haben wir nicht berücksichtigt.

💡 Wie warm ist es laut unserem Modell dann im Jahr 2051?

```
predict(lm_wetter1_zentriert, newdata = tibble(year_c = 100))  
##      1  
## 9.65775
```

💡 Moment! Die Vorhersage ist doch Quatsch! Schon im Jahr 2022 lag die Durchschnittstemperatur bei 10,5° Celcius.⁶

💡 Wir brauchen ein besseres Modell! Zum Glück haben wir ambitionierte Nachwuchs-Wissenschaftler:innen.

2.3.3. Binäre UV

Definition 2.1 (Binäre Variable). Eine *binäre* UV, auch *Indikatorvariable* oder *Dummyvariable* genannt, hat nur zwei Ausprägungen: 0 und 1.□

Beispiel 2.3 (Binäre Variablen). Das sind zum Beispiel *weiblich* mit den Ausprägungen 0 (nein) und 1 (ja) oder *before_1950* mit 1 für Jahre früher als 1950 und 0 ansonsten.□

Beispiel 2.4. Hier interessiert Sie folgende Forschungsfrage:

💡 Ob es in der zweiten Hälfte des 20. Jahrhunderts wohl wärmer warm, im Durchschnitt, als vorher?□

Aber wie erstellen Sie eine Variable `after_1950`, um die zweite Hälfte des 20. Jahrhunderts (und danach) zu fassen? Nach einem Überlegen kommen Sie auf die Idee, das vektorisierte Rechnen von R (s. [?@sec-veccalc](#)) auszunutzen:

⁵`year` und `year_c` sind gleich stark mit `temp` korreliert, daher wird sich die Modellgüte nicht unterscheiden.

⁶Quelle: [Umweltbundesamt](#)

```

year <- c(1940, 1950, 1960)
after_1950 <- year > 1950 # prüfe ob es Jahr größer als 1950
  ↵ ist
after_1950
## [1] FALSE FALSE TRUE

```

Die ersten zwei Jahre von `year` sind nicht größer als 1950, das dritte schon.

Ja, so könnte das klappen! Diese Syntax übertragen Sie auf Ihre `wetter`-Daten:

```

wetter <-
  wetter %>%
  mutate(after_1950 = year > 1950) %>%
  filter(region != "Deutschland") # ohne Daten für
  ↵ Gesamt-Deutschland

```

Scheint zu klappen!

Jetzt ein lineares Modell dazu berechnen:

```
lm_wetter_bin_uv <- lm(temp ~ after_1950, data = wetter)
```

Die Parameter des Modells lassen darauf schließen, dass es tatsächlich wärmer war nach 1950, und zwar im Schnitt offenbar ein gutes halbes Grad, s. Abbildung 2.4.

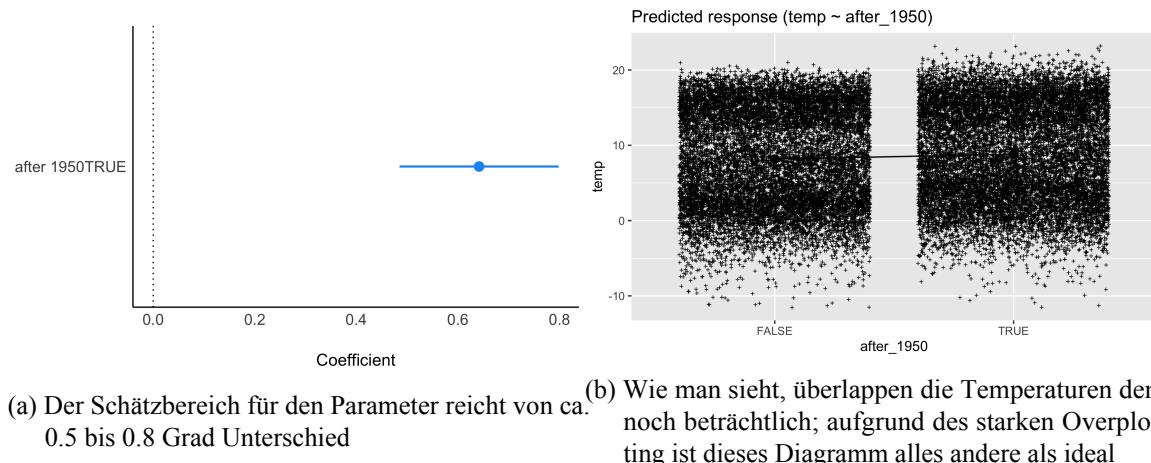


Abbildung 2.4.: Modell `temp ~ after_1950`

Leider zeigt ein Blick zum `r2`, dass die Vorhersagegüte des Modells zu wünschen übrig lässt⁷. □

⁷`r2(lm_wetter_bin_uv)`

! Lineare Modelle verkraften nur metrische Variablen

Um die Koeffizienten eines linearen Modells auszurechnen, benötigt man eine metrische X- und eine metrische Y-Variable. Hier haben wir aber keine richtige metrische X-Variable⁸, sondern eine *logische* Variable mit den Werten TRUE und FALSE.□

Um die X-Variable in eine metrische Variable umzuwandeln, gibt es einen einfachen Trick, den R für uns ohne viel Ankündigung durchführt: Umwandlung in mehrere binäre Variablen.

Hat ein nominaler Prädiktor zwei Stufen, so überführt⁹ `lm()` diese Variable in eine binäre Variable. Da eine binäre Variable metrisch ist, kann die Regression in gewohnter Weise durchgeführt werden. Wenn Sie die Ausgabe der Parameter betrachten, so sehen Sie die neu erstellte binäre Variable. Man beachte, dass der ursprüngliche Datensatz nicht geändert wird, nur während der Analyse von `lm` wird die Umwandlung der Variable¹⁰ durchgeführt.

⌚ Eine 1 kannst du als “Ja! Richtig!” verstehen und eine 0 als “Nein! Falsch!”

`after_1950` wird in eine Indikatorvariable umgewandelt:

id	after_1950
1	TRUE
2	FALSE

→

id	after_1950TRUE
1	1
2	0

Beispiel 2.5 (Beispiel: ‘Geschlecht’ in eine binäre Variable umwandeln.). Angenommen wir haben eine Variable `geschlecht` mit den zwei Stufen Frau und Mann und wollen diese in eine Indikatorvariable umwandeln. Da “Frau” alphabetisch vor “Mann” kommt, nimmt R “Frau” als *erste* Stufe bzw. als *Referenzgruppe*. “Mann” ist dann die zweite Stufe, die in der Regression dann in Bezug zur Referenzgruppe gesetzt wird. `lm` wandelt uns diese Variable in `geschlechtMann` um mit den zwei Stufen 0 (kein Mann, also Frau) und 1 (Mann).□

id	geschlecht
1	Mann
2	Frau

→

id	geschlechtMann
1	1
2	0

Ein lineares Modell mit binärer UV ist nichts anderes die Differenz der Gruppenmittelwerte zu berechnen:

```
wetter %>%
  group_by(after_1950) %>%
  summarise(temp_mean = mean(temp))
```

⁸UV

⁹synonym: transformiert

¹⁰Transformation

	after_1950	temp_mean
FALSE		8.175287
TRUE		8.816761

Die Interpretation eines linearen Modells mit binärer UV veranschaulicht Abbildung 2.5: Der Achsenabschnitt (b_0) entspricht dem Mittelwert der 1. Gruppe. Der Mittelwert der 2. Gruppe entspricht der Summe aus Achsenabschnitt und dem Koeffizienten der zweiten Gruppe. (Abbildung 2.5 zeigt nur die Daten für den Monat Juli im Bundesland Bayern, der Einfachheit und Übersichtlichkeit halber.)

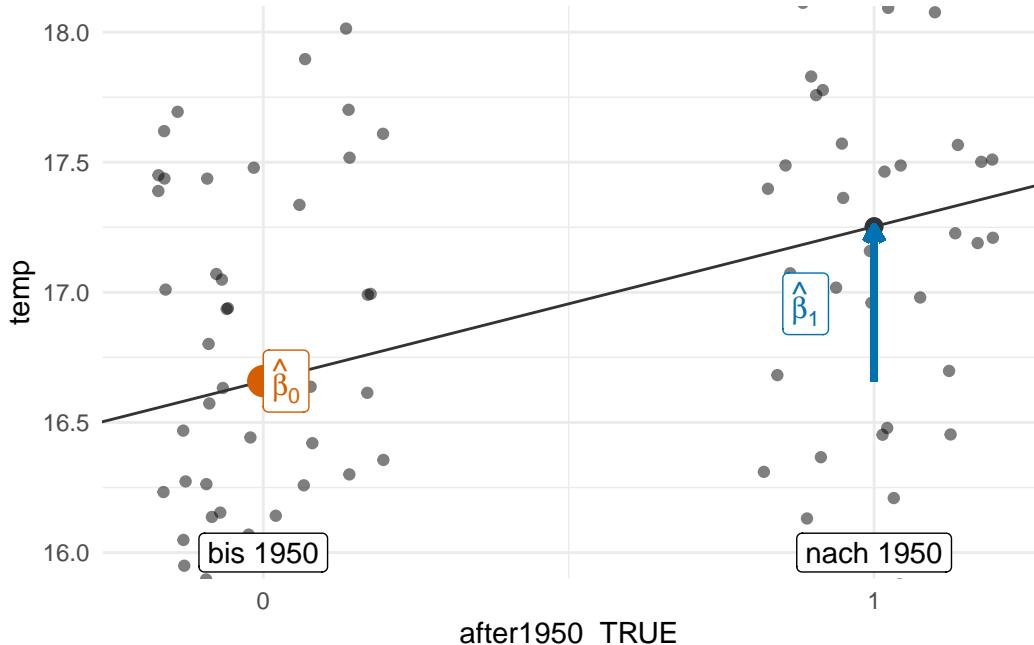


Abbildung 2.5.: Sinnbild zur Interpretation eines linearen Modells mit binärer UV (reingezoomt, um den Mittelwertsunterschied hervorzuheben)

Fassen wir die Interpretation der Koeffizienten für das Modell mit binärer UV zusammen:

1. Mittelwert der 1. Gruppe (bis 1950): Achsenabschnitt (b_0)
2. Mittelwert der 2. Gruppe (nach 1950): Achsenabschnitt (b_0) + Steigung der Regressionsgeraden (b_1)

Für die Modellwerte \hat{y} gilt also:

- Temperatur laut Modell bis 1950: $\hat{y} = \beta_0 = 17.7$
- Temperatur laut Modell bis 1950: $\hat{y} = \beta_0 + \beta_1 = 17.7 + 0.6 = 18.3$

i Hinweis

Bei *nominalen* (und auch bei *binären*) Variablen ist β_1 ein *Schalter*; bei *metrischen* Variablen ein *Dimmer*.¹¹ □

2.3.4. Nominale UV

In diesem Abschnitt betrachten wir ein lineare Modell¹² mit einer mehrstufigen¹³ (nominalskalierten) UV.¹⁴

Beispiel 2.6. Ob es wohl substanzielle¹⁵ Temperaturunterschiede zwischen den Bundesländern gibt?

Befragen wir dazu ein lineares Modell, s. Tabelle 2.10.

```
lm_wetter_region <- lm(temp ~ region, data = wetter)
parameters(lm_wetter_region)
```

Tabelle 2.10.: Modellparameter für lm_wetter_region

Parameter	Coefficient	SE	95% CI	t(27152)	p
(Intercept)	8.25	0.16	(7.93, 8.56)	51.62	< .001
region (Bayern)	-0.63	0.23	(-1.07, -0.19)	-2.79	0.005
region (Brandenburg)	0.57	0.23	(0.13, 1.02)	2.53	0.011
region (Brandenburg/Berlin)	0.58	0.23	(0.14, 1.03)	2.59	0.010
region (Hessen)	0.11	0.23	(-0.33, 0.56)	0.51	0.612
region (Mecklenburg-Vorpommern)	0.08	0.23	(-0.37, 0.52)	0.34	0.732
region (Niedersachsen)	0.52	0.23	(0.07, 0.96)	2.29	0.022
region	0.52	0.23	(0.08, 0.96)	2.31	0.021
(Niedersachsen/Hamburg/Bremen)					
region (Nordrhein-Westfalen)	0.80	0.23	(0.35, 1.24)	3.53	< .001
region (Rheinland-Pfalz)	0.46	0.23	(0.02, 0.90)	2.03	0.042
region (Saarland)	0.71	0.23	(0.27, 1.16)	3.16	0.002
region (Sachsen)	-0.04	0.23	(-0.48, 0.40)	-0.18	0.853
region (Sachsen-Anhalt)	0.55	0.23	(0.11, 1.00)	2.45	0.014
region (Schleswig-Holstein)	0.17	0.23	(-0.27, 0.62)	0.76	0.446
region (Thueringen)	-0.48	0.23	(-0.92, -0.03)	-2.11	0.035
region	0.10	0.23	(-0.34, 0.54)	0.43	0.664
(Thueringen/Sachsen-Anhalt)					

¹¹Ich danke Karsten Lübke für diese Idee.

¹²für uns synonym: Regressionsmodell

¹³drei oder mehr Stufen bzw. Ausprägungen

¹⁴So ein Modell ist von den Ergebnissen her praktisch identisch zu einer einfachen *Varianzanalyse*.

¹⁵wie könnte man dieses Wort eigentlich definieren?

Hat die nominalskalierte UV mehr als zwei Stufen, so transformiert `lm` sie in mehr als eine Indikatorvariablen um. Genauer gesagt ist es immer eine Indikatorvariablen weniger als es Stufen in der nominalskalierten Variablen gibt.

Betrachten wir ein einfaches Beispiel, eine Tabelle mit der Spalte `Bundesland` – aus Gründen der Einfachheit hier nur mit *drei* Bundesländern. Damit `lm` arbeiten kann, wird `Bundesland` in *zwei* Indikatorvariablen umgewandelt:

id	Bundesland		→	id	BL_Bayern	BL_Bra
1	BaWü			1	0	0
2	Bayern			2	1	0
3	Brandenburg			3	0	1

Auch im Fall mehrerer Ausprägungen einer nominalen Variablen gilt die gleiche Logik der Interpretation wie bei binären Variablen:

1. Mittelwert der 1. Gruppe: Achsenabschnitt (b_0)
2. Mittelwert der 2. Gruppe: Achsenabschnitt (b_0) + Steigung der 1. Regressionsgeraden (b_1)
3. Mittelwert der 2. Gruppe: Achsenabschnitt (b_0) + Steigung der 2. Regressionsgeraden (b_2)
4. usw.

Es kann nervig sein, dass das Bundesland, welches als *Referenzgruppe* (sprich als Gruppe des Achsenabschnitts ausgewählt wurde) nicht explizit in der Ausgabe angegeben ist. Der Wert der Referenzgruppe findet seinen Niederschlag im Achsenabschnitt.

i Hinweis

Bei einer Variable vom Typ `character` wählt R den alphabetisch ersten Wert als Referenzgruppe für ein lineares Modell aus. Bei einer Variable vom Typ `factor` ist die Reihenfolge bereits festgelegt, vgl. Kapitel 2.3.5. Der Mittelwert dieser Gruppe entspricht dem Achsenabschnitt. □

Beispiel 2.7 (Achsenabschnitt in `wetter_lm2`). Da Baden-Württemberg das alphabetisch erste Bundesland ist, wird es von R als Referenzgruppe ausgewählt, dessen Mittelwert als Achsenabschnitt im linearen Modell hergenommen wird. □

Am einfachsten verdeutlicht sich `lm_wetter_region` vielleicht mit einem Diagramm, s. Abbildung 2.6.

Beispiel 2.8 (Niederschlagsmenge im Vergleich der Monate). Eine weitere Forschungsfrage, die Sie nicht außer acht lassen wollen, ist die Frage nach den jahreszeitlichen Unterschieden im Niederschlag (engl. `precipitation`). Los R, rechnen!

Endlich geht's weiter! Ergebnisse in Tabelle 2.13! □

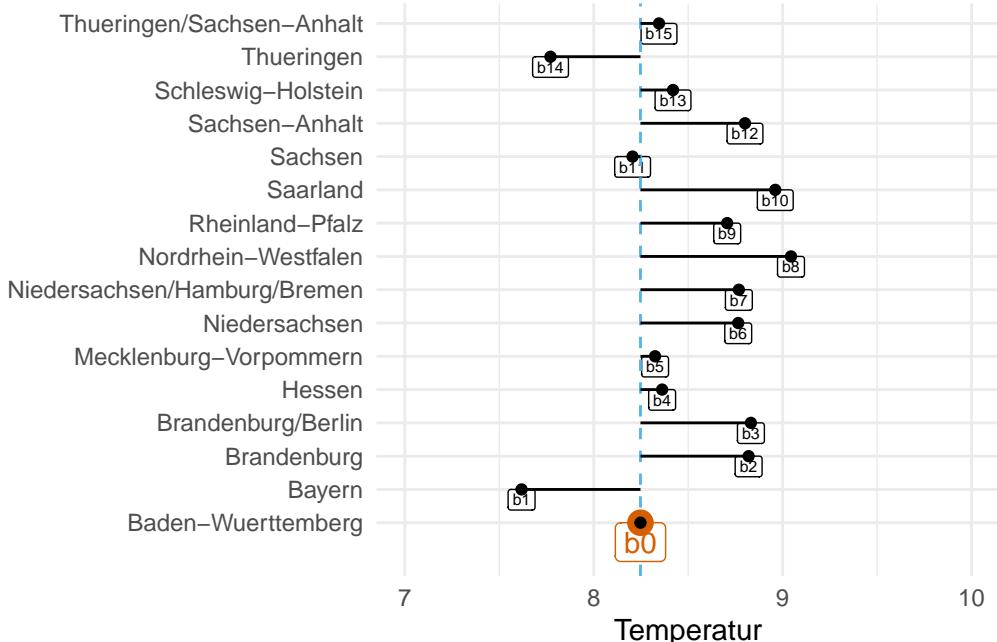


Abbildung 2.6.: Sinnbild zur Interpretation eines linearen Modells mit nominaler UV (reingezoomt, um den Mittelwertsunterschied hervorzuheben). Die Achsen wurden um 90° gedreht, damit man die Namen der Bundesländer besser lesen kann.

```
lm_wetter_month <- lm(precip ~ month, data = wetter)
parameters(lm_wetter_month)
```

Tabelle 2.13.: Modellparameter für lm_wetter_month

Parameter	Coefficient	SE	95% CI	t(27166)	p
(Intercept)	53.27	0.41	(52.46, 54.08)	128.76	< .001
month	1.14	0.06	(1.03, 1.25)	20.29	< .001

Ja, da scheint es deutliche Unterschiede im Niederschlag zu geben. Wir brauchen ein Diagramm zur Verdeutlichung, s. Abbildung 2.7, links.¹⁶ Oh nein: R betrachtet month als numerische Variable! Aber ‘‘Monat’’ bzw. ‘‘Jahreszeit’’ sollte nominal sein.

⌚ Aber month ist als Zahl in der Tabelle hinterlegt. Jede ehrliche Maschine verarbeitet eine Zahl als Zahl, ist doch klar!

🐱 Okay, R, wir müssen month in eine nominale Zahl transformieren. Wie geht das?

¹⁶plot(estimate_expectation(lm_wetter_month))

💡 Dazu kannst du den Befehl `factor` nehmen. Damit wandelst du eine numerische Variable in eine nominalskalierte Variable (Faktorvariable) um. Faktisch heißt das, dass dann eine Zahl als Text gesehen wird.

Beispiel 2.9. Transformiert man 42 mit `factor`, so wird aus 42 "42". Aus der Zahl wird ein Text. Alle metrischen Eigenschaften gehen verloren; die Variable ist jetzt auf nominalen Niveau. □

```
wetter <-
wetter %>%
  mutate(month_factor = factor(month))
```

Jetzt berechnen wir mit der faktorisierten Variablen ein lineares Modell, s. Tabelle 2.14.

```
lm_wetter_month_factor <- lm(precip ~ month_factor, data =
  ↴ wetter)
parameters(lm_wetter_month_factor)
```

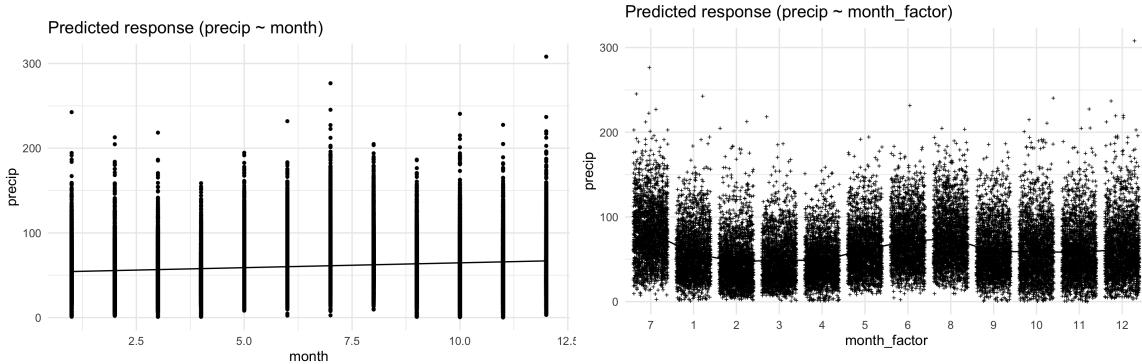
Tabelle 2.14.: Modellparameter von `lm_wetter_month_factor`

Parameter	Coefficient	SE	95% CI	t(27156)	p
(Intercept)	56.95	0.64	(55.68, 58.21)	88.56	< .001
month factor (2)	-9.95	0.91	(-11.73, -8.17)	-10.94	< .001
month factor (3)	-7.78	0.91	(-9.56, -6.00)	-8.56	< .001
month factor (4)	-8.49	0.91	(-10.27, -6.71)	-9.34	< .001
month factor (5)	4.74	0.91	(2.96, 6.53)	5.22	< .001
month factor (6)	14.34	0.91	(12.56, 16.12)	15.77	< .001
month factor (7)	24.36	0.91	(22.57, 26.14)	26.74	< .001
month factor (8)	17.52	0.91	(15.74, 19.31)	19.24	< .001
month factor (9)	1.93	0.91	(0.15, 3.72)	2.12	0.034
month factor (10)	2.29	0.91	(0.51, 4.08)	2.52	0.012
month factor (11)	0.89	0.91	(-0.89, 2.68)	0.98	0.327
month factor (12)	5.20	0.91	(3.42, 6.99)	5.71	< .001

Sehr schön! Jetzt haben wir eine Referenzgruppe (Monat 1, d.h. Januar) und 11 Unterschiede zum Januar, s. Abbildung 2.7, rechts.

Möchte man die Referenzgruppe eines Faktors ändern, kann man dies mit `relevel` tun:

```
wetter <-
wetter %>%
  mutate(month_factor = relevel(month_factor, ref = "7"))
```



(a) `lm_wetter_month`, Monat fälschlich als metrische Variable
(b) `lm_wetter_month_text`, Monat korrekt als nominale Variable (aber mit viel Overplotting, das müsste man besser machen)

Abbildung 2.7.: Niederschlagsunterschiede pro Monat (ein Punkt ist ein Jahr); aufgrund der vielen Datenpunkte ist das Diagramm wenig übersichtlich (Overplotting).

So sieht dann die geänderte Reihenfolge aus:¹⁷

```
levels(wetter$month_factor)
## [1] "7"   "1"   "2"   "3"   "4"   "5"   "6"   "8"   "9"   "10"  "11"
## [12] "12"
```

2.3.5. Binäre plus metrische UV

In diesem Abschnitt untersuchen wir ein lineares Modell mit zwei UV: einer *zweistufigen* (binären) UV plus einer *metrischen* UV.¹⁸

Beispiel 2.10. Ob sich die Niederschlagsmenge wohl unterschiedlich zwischen den Monaten entwickelt hat in den letzten gut 100 Jahren? Der Einfachheit halber greifen Sie sich nur zwei Monate heraus (Januar und Juli).

```
wetter_month_1_7 <-
wetter %>%
filter(month == 1 | month == 7)
```

💡 Ich muss mal kurz auf eine Sache hinweisen...

¹⁷Zum Dollar-Operator s. [?@sec-dollar-op](#)

¹⁸So ein Modell kann auch als *Kovarianzanalyse* (engl. analysis of covariance, ancova) bezeichnet werden.

i Faktorvariable

Eine Faktorvariable ist einer der beiden Datentypen in R, die sich für nominalskalierte Variablen anbieten: Textvariablen (`character`) und Faktor-Variablen (`factor`). Ein wichtiger Unterschied ist, dass die erlaubten Ausprägungen (“Faktorstufen”) bei einer Faktor-Variable mitgespeichert werden, bei der Text-Variable nicht.

Das kann praktisch sein, denn bei einer Faktorvariable ist immer klar, welche Ausprägungen in Ihrer Variable möglich sind.□

Beispiel 2.11 (Beispiel für eine Faktorvariable).

```
geschlecht <- c("f", "f", "m")
geschlecht_factor <- factor(geschlecht)
geschlecht_factor
## [1] f f m
## Levels: f m
```

Beispiel 2.12 (Filtern verändert die Faktorstufen nicht). Wenn Sie von der Faktorvariablen¹⁹ `geschlecht` das 3. Element ("m") herausfiltern, so dass z.B. nur die ersten beiden Elemente übrig bleiben mit allein der Ausprägung "f", merkt sich R trotzdem, dass es *zwei* Faktorstufen gibt ("f" und "m").

Genaus so ist es, wenn Sie aus `wetter` nur die Monate "1" und "7" herausfiltern: R merkt sich, dass es 12 Faktorstufen gibt. Möchten Sie die herausgefilterten Faktorstufen “löschen”, so können Sie einfach die Faktorvariable neu berechnen (mit `factor`).□

```
wetter_month_1_7 <-
wetter %>%
filter(month == 1 | month == 7) %>%
# Faktor (und damit die Faktorstufen) neu berechnen:
mutate(month_factor = factor(month))
```

Okay. Wie spezifiziert man jetzt das lineare Modell?□

Hat man mehrere (“multiple”) X-Variablen²⁰, so trennt man sich mit einem Plus-Zeichen in der Regressionsformel, z.B. `temp ~ year_c + month`.

! Multiple Regression

Eine multiple Regression beinhaltet mehr als eine X-Variable. Die Modellformel spezifiziert man so:

¹⁹synonym: nominalskalierte Variable

²⁰Prädiktoren, unabhängige Variablen, X-Variablen

$$y x_1 + x_2 + \dots + x_n \quad \square$$

i Modellgleichung

Das Pluszeichen hat in der Modellgleichung²¹ *keine* arithmetische Funktion. Es wird nichts addiert. In der Modellgleichung sagt das Pluszeichen nur “und noch folgende UV...”. \square

Die obige Modellgleichung liest sich also so:

Temperatur ist eine Funktion von der (zentrierten) Jahreszahl und des Monats

```
lm_year_month <- lm(precip ~ year_c + month_factor, data =
  ↴ wetter_month_1_7)
```

Die Modellparameter sind in Tabelle 2.15 zu sehen.

Tabelle 2.15.: Modellparameter von lm_year_month

Parameter	Coefficient	SE	95% CI	t(4525)	p
(Intercept)	56.94	0.68	(55.60, 58.27)	83.57	< .001
year c	0.03	0.01	(5.59e-03, 0.05)	2.43	0.015
month factor (7)	24.37	0.97	(22.48, 26.27)	25.25	< .001

Die Modellkoeffizienten sind so zu interpretieren:

1. Achsenabschnitt (b_0 , (Intercept)): Im Referenzjahr (1951) im *Referenzmonat Januar* lag die Niederschlagsmenge bei 57 mm pro Quadratmeter.
2. Regressionskoeffizient für Jahr (b_1 , year_c): Pro Jahr ist die Niederschlagsmenge im Schnitt um 0.02 mm an (im Referenzmonat).
3. Regressionskoeffizient für Monat (b_2 , month [7]) Im Monat 7 (Juli) lag die mittlere Niederschlagsmenge (im Referenzjahr) knapp 25 mm über dem mittleren Wert des Referenzmonats (Januar).

Die Regressiongleichung von lm_year_month lautet: precip_pred = 56.94 + 0.03*year_c + 24.37*month_factor_7.

Im Monat Juli ist month_factor_7 = 1, ansonsten (Januar) ist month_factor = 0.

💡 Puh, kompliziert!

💡 Es gibt einen Trick, man kann sich von R einfach einen beliebigen Y-Wert berechnen lassen, s. Beispiel 2.13.

²¹synonym: Regressionsformel

Beispiel 2.13 (Niederschlag laut Modell Im Juli 2020?). Hey R, berechne uns anhand neuer Daten den laut Modell zu erwartenden Niederschlag für Januar im Jahr 2020!

```
neue_daten <- tibble(year_c = 2020-1951,
                      month_factor = factor("1"))
predict(lm_year_month, newdata = neue_daten)
##      1
## 58.92171
```

i Hinweis

Alle Regressionskoeffizienten beziehen sich auf den Y-Wert *unter der Annahme, dass alle übrigen Prädiktoren den Wert Null (bzw. Referenzwert) aufweisen.* □

Visualisieren wir uns die geschätzten Erwartungswert pro Prädiktorwert, s. Abbildung 2.8:
`plot(estimate_expectation(lm_year_month))`

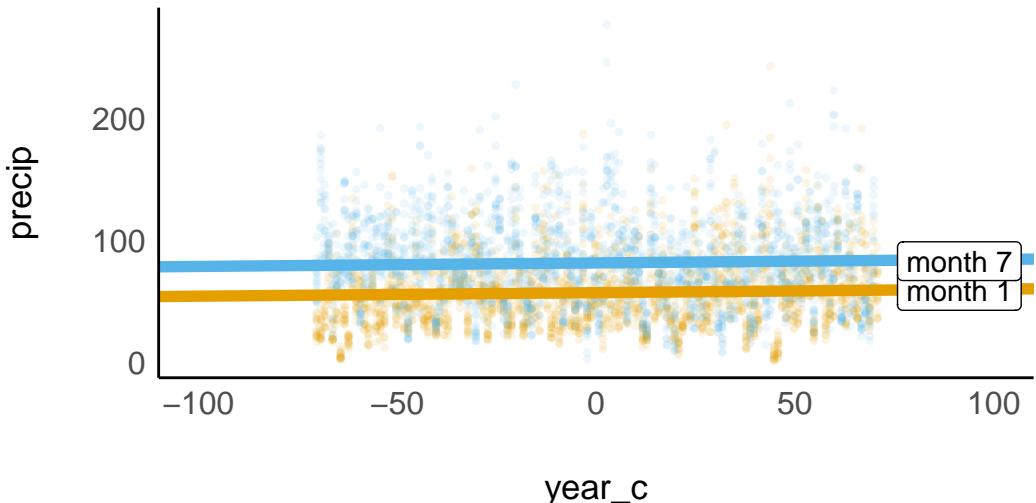


Abbildung 2.8.: Temperaturverlauf über die Jahre für zwei Monate. Man beachte, dass die Regressionsgeraden *parallel* sind.

Mit `scale_color_okabeito` haben wir die Standard-Farbpalette durch die von (Okabe und Ito 2023) ersetzt²². Das ist nicht unbedingt nötig, aber robuster bei Schwarz-Weiß-Druck und bei Sehschwächen, vgl. `?@sec-farbwahl`.

Die erklärte Varianz von `lm_year_month` liegt bei:

```
r2(lm_year_month)
## # R2 for Linear Regression
##      R2: 0.124
## adj. R2: 0.124
```

²²s. Hinweise hier: https://malcolmbarrett.github.io/ggokabeito/reference/palette_okabe_ito.html

2.3.6. Interaktion

Eine Modellgleichung der Form $\text{temp} \sim \text{year} + \text{month}$ zwingt die Regressionsgeraden dazu, parallel zu verlaufen. Aber vielleicht würden sie besser in die Punktwolken passen, wenn wir ihnen erlauben, auch *nicht* parallel verlaufen zu dürfen?

Nicht-parallele Regressionsgeraden erlauben wir, indem wir das Regressionsmodell wie folgt spezifizieren und visualisieren, s. Listing 2.1.

Listing 2.1 Ein Interaktionsmodell spezifiziert man in dieser Art: $y \sim x_1 + x_2 + x_1:x_2$

```
lm_year_month_interaktion <- lm(  
  precip ~ year_c + month_factor + year_c:month_factor,  
  data = wetter_month_1_7)
```

Visualisiert ist das Modell in Abbildung 2.9.

```
plot(estimate_expectation(lm_year_month_interaktion)) +  
  scale_color_okabeito() # schönes Farbschema
```

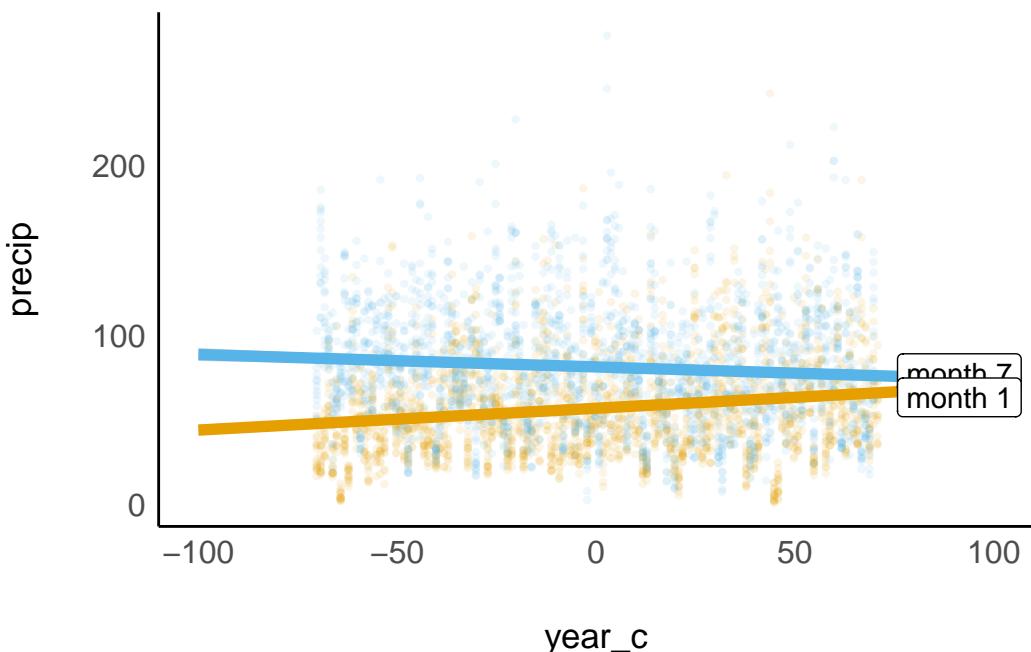


Abbildung 2.9.: Niederschlag im Jahresverlauf und Monatsvergleich mit Interaktionseffekt: Die Veränderung im Verlauf der Jahre ist unterschiedlich für die Monate (Janur vs. Juli). Die beiden Regressionsgeraden sind *nicht* parallel.

Der *Doppelpunkt-Operator* (`:`) fügt der Regressionsgleichung einen *Interaktionseffekt* hinzu, in diesem Fall die Interaktion von Jahr (`year_c`) und Monat (`month_factor`):

`precip ~ year_c + month_factor + year_c:month_factor`

! Wichtig

Einen Interaktionseffekt von x_1 und x_2 kennzeichnet man in R mit dem Doppelpunkt-Operator,
 $x_1:x_2$:

$$y \sim x_1 + x_2 + x_1:x_2 \square$$

In Worten:

y wird modelliert als eine Funktion von x_1 und x_2 und dem Interaktionseffekt von x_1 mit x_2 .

Wie man in Abbildung 2.9 sieht, sind die beiden Regressionsgeraden *nicht parallel*.

i Hinweis

Sind die Regressionsgeraden von zwei (oder mehr) Gruppen nicht parallel, so liegt ein Interaktionseffekt vor.□

Beispiel 2.14 (Interaktionseffekt von Niederschlag und Monat). Wie ist die Veränderung der Niederschlagsmenge (Y-Achse) im Verlauf der Jahre (X-Achse)? *Das kommt darauf an, welchen Monat man betrachtet*. Der Effekt der Zeit ist *unterschiedlich* für die Monate: Im Juli nahm der Niederschlag ab, im Januar zu.□

Liegt ein Interaktionseffekt vor, kann man nicht mehr von “dem” (statistischen) Effekt eines Prädiktors (afu die Y-Variable) sprechen. Vielmehr muss man unterscheiden: Je nach Gruppe (z.B. Monat) unterscheidet der Effekt.²³

Betrachten wir die Parameterwerte des Interaktionsmodells (`parameters(lm_year_month_interaktion)`) s. Tabelle 2.16.

Tabelle 2.16.: Modellparameter von `lm_year_month_interaktion`

Parameter	Coefficient	SE	95% CI	t(4524)	p
(Intercept)	56.91	0.68	(55.59, 58.24)	84.21	< .001
year c	0.13	0.02	(0.10, 0.16)	7.80	< .001
month factor (7)	24.37	0.96	(22.50, 26.25)	25.45	< .001
year c × month factor (7)	-0.20	0.02	(-0.25, -0.16)	-8.62	< .001

Neu bei der Ausgabe zu diesem Modell ist die Zeile `year c × month factor [7]`. Sie gibt die Stärke des Interaktionseffekts an. Die Zeile zeigt, wie unterschiedlich sich die die Niederschlagsmenge zwischen den beiden Monaten im Verlauf der Jahre ändert: Im Monat "7" ist der Effekt von `year_c` um 0.20 mm geringer: Die Regressionsgerade neigt sich mehr nach “unten” im Monat Juli, da der Koeffizient kleiner als Null ist.

²³Effekt ist hier immer statistisch, nie kausal gemeint.

Die Regressionsgleichung lautet: $\text{precip_pred} = 56.91 + 0.13*\text{year_c} + 24.37*\text{month_factor_7} - 0.20*\text{year_c}:\text{month_factor_7}$.

! Wichtig

Der Achsenabschnitt gibt den Wert für Y an unter der Annahme, dass alle Prädiktoren den Wert Null aufweisen. In diesem Fall gibt der Achsenabschnitt also den Niederschlag für den Janur des Jahres 1951 an. Die Regressionskoeffizienten geben die Zunahme in Y an, wenn der jeweilige Prädiktorwert um 1 steigt, die übrigen Prädiktoren aber den Wert 0 aufweisen. \square

Das R-Quadrat von `lm_year_month_interaktion` beträgt übrigens nur geringfügig mehr als im Modell ohne Interaktion:

```
r2(lm_year_month_interaktion) # aus `easystats`  
## # R2 for Linear Regression  
##      R2: 0.139  
## adj. R2: 0.138
```

2.4. Modelle mit vielen UV

2.4.1. Zwei metrische UV

Ein Modell mit zwei metrischen UV kann man sich im 3D-Raum visualisieren, s. Abbildung 2.12, oder im 2D-Raum, s. Abbildung 2.11. Im 3D-Raum wird die Regressionsgerade zu einer *Regressionsfläche*.

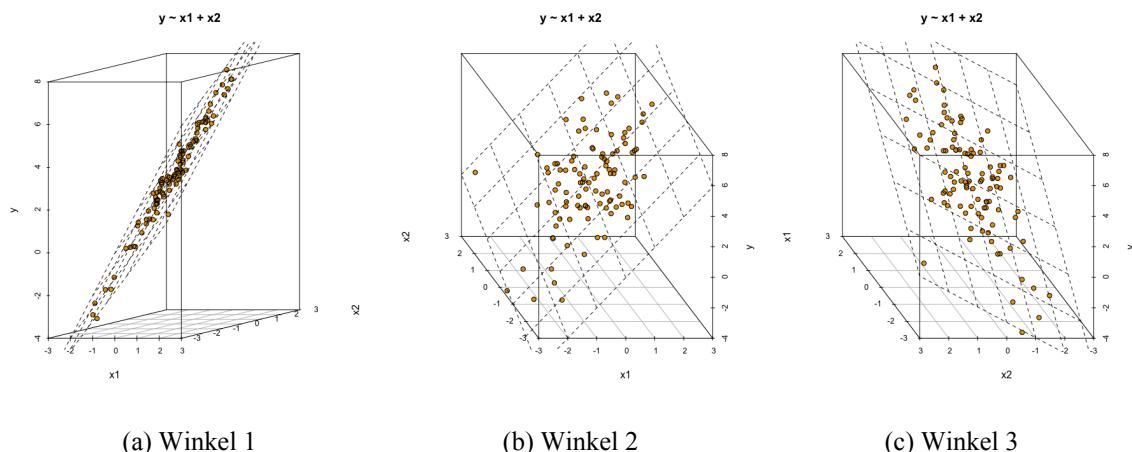


Abbildung 2.10.: Ein lineares Modell, $y \sim x_1 + x_2$ mit zwei Prädiktoren im 3D-Raum.

Grundsätzlich kann man viele Prädiktoren in ein (lineares) Modell aufnehmen. Betrachten wir z. B. folgendes lineares Modell mit zwei metrischen UV.

Predicted response ($y \sim x_1 + x_2$)

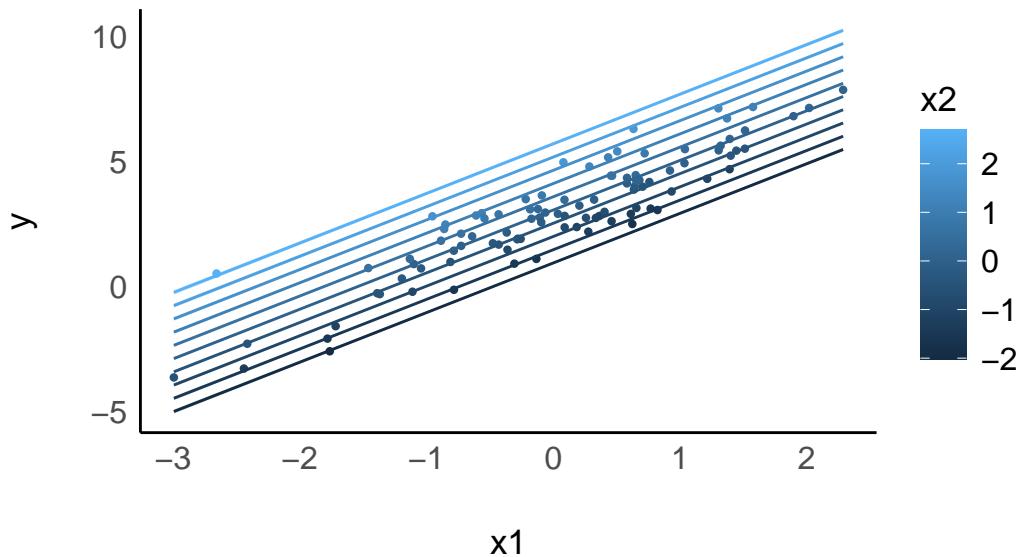


Abbildung 2.11.: 2D-Diagramm für 3D-Modell

```
lm_mario_2uv <- lm(total_pr ~ start_pr + ship_pr, data =
  ↵ mariokart %>% filter(total_pr < 100))
```

?@fig-mario-2uv visualisiert das Modell lm_mario2v in einem 3D-Diagramm (betrachtet aus verschiedenen Winkeln).

2.4.2. Viele UV ins Modell?

Wir könnten im Prinzip alle Variablen unserer Datentabelle als Prädiktoren in das Regressionsmodell aufnehmen. Die Frage ist nur: Macht das Sinn?

Hier sind einige Richtlinien, die helfen, welche Prädiktoren (und wie viele) man in ein Modell aufnehmen sollte (Gelman, Hill, und Vehtari 2021), s. S. 199:

1. Man sollte alle Prädiktoren aufnehmen, von denen anzunehmen ist, dass Sie Ursachen für die Zielvariablen sind
2. Bei Prädiktoren mit starken (absoluten) Effekten kann es Sinn machen, ihre Interaktionseffekte auch mit in das Modell aufzunehmen
3. Prädiktoren mit kleinem Schäzbereich (95% CI) sollten tendenziell im Modell belassen werden, da sie die Modellgüte verbessern

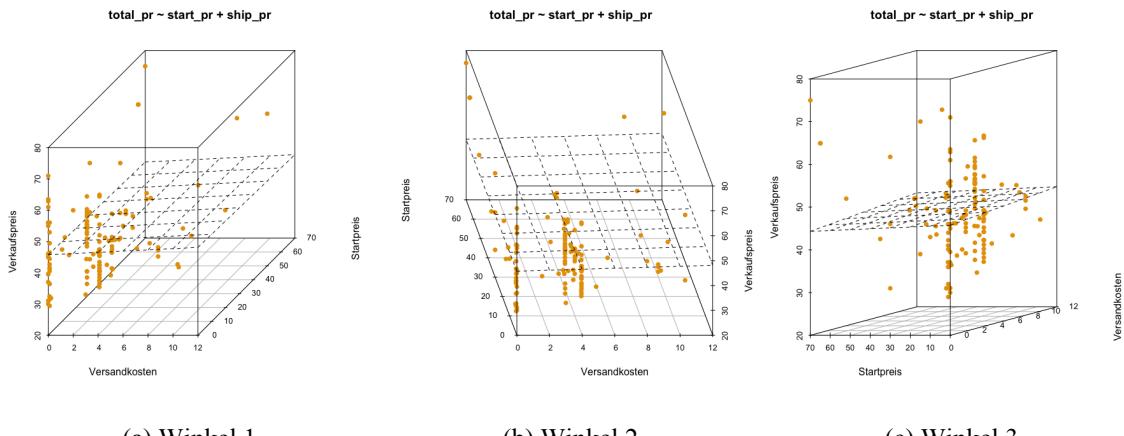


Abbildung 2.12.: Das Modell lm_mario2v mit 2 metrischen UV (und 1 metrische AV) als 3D-Diagramm

2.5. Fallbeispiel zur Prognose

Beispiel 2.15 (Prognose des Verkaufspreis). Ganz können Sie von Business-Welt und ihren Gratifikationen nicht lassen, trotz Ihrer wissenschaftlichen Ambitionen. Sie haben den Auftrag bekommen, den Verkaufspreis von MarioKart-Spielen möglichst exakt vorherzusagen. Also gut, das Honorar ist phantastisch, Sie sind jung und brauchen das Geld.□

2.5.1. Modell “all-in”

Um die Güte Ihrer Vorhersagen zu prüfen, teilt Ihr Chef den Datensatz in zwei zufällige Teile.

□ ↗ Ich teile den Datensatz mariokart zufällig in zwei Teile. Den ersten Teil kannst du nutzeln, um Modelle zu berechnen (“trainieren”) und ihre Güte zu prüfen. Den Teil nenne ich “Trainingssample”, hört sich cool an, oder? Im Train-Sample ist ein Anteil (`fraction`) von 70% der Daten, okay? Die restlichen Daten behalte ich. Wenn du ein gutes Modell hast, kommst du und wir berechnen die Güte deiner Vorhersagen in dem verbleibenden Teil, die übrigen 30% der Daten. Diesen Teil nennen wir Test-Sample, alles klar?

Wenn die Daten auf Ihrer Festplatte liegen, z.B. im Unterordner `daten`, dann können Sie sie von dort importieren:

```
mariokart_train <- read.csv("daten/mariokart_train.csv")
```

Alternativ können Sie sie auch von diesem Pfad von einem Rechner in der Cloud herunterladen:

```

mariokart_train_path <- paste0(
  "https://raw.githubusercontent.com/sebastiansauer/",
  "statistik1/main/daten/mariokart_train.csv")

mariokart_train <- read.csv(mariokart_train_path)

```

Dann importieren wir auf gleiche Weise Test-Sample in R:

```

mariokart_test_path <- paste0(
  "https://raw.githubusercontent.com/sebastiansauer/",
  "statistik1/main/daten/mariokart_test.csv")

mariokart_test <- read.csv(mariokart_test_path)

```

Also los. Sie probieren mal die “All-in-Strategie”: Alle Variablen rein in das Modell. Viel hilft viel, oder nicht?

```

lm_allin <- lm(total_pr ~ ., data = mariokart_train)
r2(lm_allin) # aus easystats
## # R2 for Linear Regression
##          R2: 0.994
## adj. R2: 0.979

```

Der Punkt in `total_pr ~ .` heißt “alle Variablen in der Tabelle (außer `total_pr`)”.

👉 Hey! Das ist ja fast perfekte Modellgüte!

⚠️ Vorsicht: Wenn ein Angebot aussieht wie “too good to be true”, dann ist es meist auch too good to be true.

💡 Overfitting

Der Grund für den fast perfekten Modellfit ist die Spalte `Title`. Unser Modell hat einfach den Titel jeder Auktion auswendig gelernt. Weiß man, welcher Titel zu welcher Auktion gehört, kann man perfekt die Auktion aufsagen bzw. das Verkaufsgebot perfekt vorhersagen. Leider nützen die Titel der Auktionen im Train-Sample *nichts* für andere Auktionen. Im Test-Sample werden unsere Vorhersagen also grottenschlecht sein, wenn wir uns auf die Titel der Auktionen im Test-Sample stützen. Merke: Höchst idiografische Informationen wie Namen, Titel etc. sind nicht nützlich, um allgemeine Muster zu erkennen und damit exakte Prognosen zu erstellen.□

Probieren wir also die Vorhersage im Test-Sample:

```
predict(lm_allin, newdata = mariokart_test)
## Error in eval(predvars, data, env): object 'V1' not found
```

Oh nein! Was ist los!? Eine Fehlermeldung!

🔥 Vorsicht

Nominalskalierte Prädiktorvariablen mit vielen Ausprägungen, wie `title` sind problematisch. Kommt eine Ausprägung von `title` im Test-Sample vor, die es *nicht* im Train-Sample gab, so resultiert ein Fehler beim `predict`. Häufig ist es sinnvoll, auf diese Variable zu verzichten, da diese Variablen oft zu Overfitting führen.□

2.5.2. Modell “all-in”, ohne Titelspalte

Okay, also auf die Titelspalte sollten wir vielleicht besser verzichten. Nächster Versuch.

```
mariokart_train2 <-
  mariokart_train %>%
  select(-c(title, V1, id))
```

Wir entfernen auch die Spalte `V1` und `id`, da sie ebenfalls keine Informationen bergen.

```
lm_allin_no_title <- lm(total_pr ~ ., data = mariokart_train2)
r2(lm_allin_no_title)
## # R2 for Linear Regression
##          R2: 0.521
## adj. R2: 0.441
```

Das R-Quadrat ist ja durchaus ordentlich. Schauen wir uns noch den `rmse` (die SD der Vorhersagefehler) an²⁴:

🏆 Gut gemacht!

```
performance::rmse(lm_allin_no_title)
## [1] 20.22998
```

🔥 Name Clash

Im Paket `yardstick` gibt es eine Funktion namens `rmse` und im Paket `performance`, Teil des Meta-Pakets `easystats` ebenfalls. Da sind Probleme vorprogrammiert. Das ist so

²⁴der Befehl wohnt im Paket `performance`, Teil des Metapakets `easystats`

als würde die Lehrerin rufen: “Schorsch, komm her!”. Dabei gibt es zwei Schorsche in der Klasse: Den Müllers Schorsch und den Meiers Schorsch. Sonst kommen beide, was die Lehrerin nicht will. Die Lehrerin müsste also rufen: “Müller Schorsch, komm her!”. Genau dasselbe machen wir, wenn wir das R-Paket eines Befehls mitschreiben, sozusagen den “Nachnamen” des Befehls: `paketname::funktion` ist wie `Müller::Schorsch`. In unserem Fall also: `performance::rmse` Endlich weiß R wieder, was zu tun ist!□

Sie rennen zu Ihrem Chef, der jetzt die Güte Ihrer Vorhersagen in den *restlichen* Daten bestimmen soll.

👉 Da wir dein Modell in diesem Teil des Komplett-Datensatzes *testen*, nennen wir diesen Teil das “Test-Sample”.

Ihr Chef schaut sich die Verkaufspreise im Test-Sample an:

```
mariokart_test %>%
  select(id, total_pr) %>%
  head()
```

	id	total_pr
1	120477729093	37.02
2	290355805215	54.99
3	180415462166	56.01
4	180415244903	56.00
5	350261958546	64.95
6	110443013258	46.50

👉 Okay, hier sind die ersten paar echten Verkaufspreise. Jetzt mach mal deine Vorhersagen auf Basis deines besten Modells!

Hier sind Ihre Vorhersagen²⁵:

```
lm_allin_predictions <- predict(lm_allin_no_title, newdata =
  mariokart_test)
```

Hier sind Ihre ersten paar Vorhersagen:

```
head(lm_allin_predictions)
##      1      2      3      4      5      6
## 28.62826 53.85885 53.28035 54.03619 41.75512 46.57713
```

Dies Vorhersagen fügen wir noch der Ordnung halber in die Tabelle mit den Test-Daten:

²⁵engl. predictions; to predict: vorhersagen

```
mariokart_test <-
  mariokart_test %>%
  mutate(lm_allin_predictions = predict(lm_allin_no_title,
    ↴ newdata = mariokart_test))
```

Okay, was ist jetzt der mittlere Vorhersagefehler?

Um die Vorhersagegüte im Test-Sample auszurechnen²⁶, nutzen wir die Funktionen des R-Paketes `yardstick`²⁷:

```
library(yardstick)

yardstick::mae(data = mariokart_test,
               truth = total_pr, # echter Verkaufspreis
               estimate = lm_allin_predictions) # Ihre
               ↴ Vorhersage
yardstick::rmse(data = mariokart_test,
                 truth = total_pr, # echter Verkaufspreis
                 estimate = lm_allin_predictions) # Ihre
                 ↴ Vorhersage
```

```
library(yardstick)

yardstick::mae(data = mariokart_test,
               truth = total_pr, # echter Verkaufspreis
               estimate = lm_allin_predictions) |> # Ihre
               ↴ Vorhersage
knitr::kable()
yardstick::rmse(data = mariokart_test,
                 truth = total_pr, # echter Verkaufspreis
                 estimate = lm_allin_predictions) |> # Ihre
                 ↴ Vorhersage
knitr::kable()
```

Ihr mittlerer Vorhersagefehler (RMSE) liegt bei ca. 13 Euro.²⁸

👉 Ganz okay.

²⁶wir verwenden dazu die Funktionen `mae` und `rsq`

²⁷welches Sie vielleicht noch installieren müssen.

²⁸Wir haben hier `yardstick::rmse` geschrieben und nicht nur `rmse`, da es sowohl im Paket `performance` (Teil des Metapakets `easystats`) als auch im Paket `yardstick` (Teil des Metapakets `tidymodels`) einen Befehl des Namens `rmse` gibt. Name-Clash-Alarm! R könnte daher den anderen `rmse`“ meinen als Sie, was garantiert zu Verwirrung führt. Entweder bei R oder bei Ihnen.

Wie ist es um das R-Quadrat Ihrer Vorhersagen bestellt?

```
# `rsq` ist auch aus dem Paket yardstick:  
rsq(data = mariokart_test,  
     truth = total_pr, # echter Verkaufspreis  
     estimate = lm_allin_predictions) # Ihre Vorhersage
```

.metric	.estimator	.estimate
rsq	standard	0.1741705

🤔 17%, nicht berauschend, aber immerhin!

i Modellgüte im Test-Sample meist geringer als im Train-Sample

Wie das Beispiel zeigt, ist die Modellgüte im Test-Sample (leider) oft *geringer* als im Train-Sample. Die Modellgüte im Train-Sample ist mitunter übermäßig optimistisch. Dieses Phänomen bezeichnet man als *Overfitting*.□

💡 Tipp

Bevor man Vorhersagen eines Modells einreicht, bietet es sich, die Modellgüte in einem neuen Datensatz, als einem Test-Sample, zu überprüfen.□

2.6. Vertiefung: Das Aufteilen Ihrer Daten

2.6.1. Analyse- und Assessment-Sample

Wenn Sie eine robuste Schätzung der Güte Ihres Modells erfahren möchten, bietet sich folgendes Vorgehen an (vgl. Abbildung 2.13):

1. Teilen Sie Ihren Datensatz (das Train-Sample) in zwei Teile: Das sog. Validation-Sample und das sog. Assessment-Sample.
2. Berechnen Sie Ihr Modell im ersten Teil Ihres Datensatzes (dem *Validation-Sample*).
3. Prüfen Sie die Modellgüte im zweiten Teil Ihres Datensatzes (dem *Assessment-Sample*)

Diese Aufteilung Ihres Datensatzes in diese zwei Teile nennt man auch *Validierungsaufteilung* (validation split); Sie können sie z.B. so bewerkstelligen:

```
library(rsample)  
mariokart <- read_csv("daten/mariokart.csv") # Wenn die  
# CSV-Datei in einem Unterordner mit Namen "daten" liegt  
  
meine_aufteilung <- initial_split(mariokart, strata =  
# total_pr)
```

`initial_split` bestimmt für jede Zeile (Beobachtung) zufällig aus, ob diese Zeile in das Analyse- oder in das Assessment-Sample kommen soll. Im Standard werden 75% der Daten in das Analyse- und 25% in das Assessment-Sample eingeteilt²⁹; das ist eine sinnvolle Aufteilung. Das Argument `strata` sorgt dafür, dass die Verteilung der AV in beiden Stichproben gleich ist. Es wäre nämlich blöd für Ihr Modell, wenn im Train-Sample z.B. nur die teuren, und im Test-Sample nur die günstigen Spiele landen würde.³⁰ In so einem Fall würde sich Ihr Modell unnötig schwer tun.

Im nächsten Schritt können Sie anhand anhand der von `initial_split` bestimmten Aufteilung die Daten tatsächlich aufteilen.³¹

```
mariokart_train <- training(meine_aufteilung)    #
  ↳ Analyse-Sample
mariokart_test <- testing(meine_aufteilung)    #
  ↳ Assessment-Sample
```

Ich persönliche nenne die Tabelle mit den Daten gerne `d_analysis` bzw. `d_assess`, das ist kürzer zu tippen und einheitlich. Sie können aber auch ein eigenes Namens-Schema nutzen; was aber hilfreich ist, ist Konsistenz in der Benamung, außerdem Kürze und aussagekräftige Namen.

2.6.2. Train- vs. Test-Sample

Definition 2.2 (Train-Sample). Den Datensatz, für die Sie sowohl UV als auch AV vorliegen haben, nennt man Train-Sample. □

Das Train-Sample stellt die bekannten Daten dar; aus denen können wir lernen, d.h. unser Modell berechnen.

Definition 2.3 (Test-Sample). Den Datensatz, für den Sie nur Daten der UV, aber nicht zu der AV vorliegen haben, nennt man Test-Sample. □

Das Test-Sample stellt das Problem der wirklichen Welt dar: Neue Beobachtungen, von denen man (noch) nicht weiß, was der Wert der AV ist.

Der Zusammenhang dieser verschiedenen, aber zusammengehörigen Arten von Stichproben ist in Abbildung 2.13 dargestellt.

²⁹vgl. `help(initial_split)`

³⁰Anderes Beispiel: In den ersten Zeilen stehen nur Kunden aus Land A und in den unteren Zeilen nur aus Land B.

³¹`initial_split` sagt nur, welche Zeile in welche der beiden Stichproben kommen soll. Die eigentliche Aufteilung wird aber noch nicht durchgeführt.

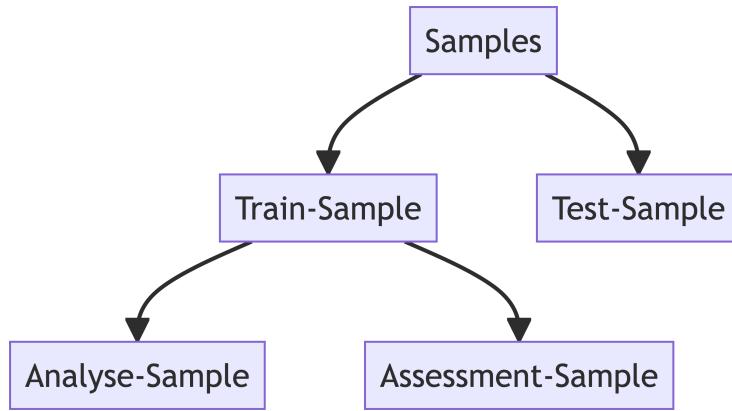


Abbildung 2.13.: Verschiedene Arten von zusammengehörigen Stichprobenarten im Rahmen einer Prognosemodellierung

2.7. Praxisbezug

Ein Anwendungsbezug von moderner Datenanalyse ist es vorherzusagen, welche Kunden “abwanderungsgefährdet” sind, also vielleicht in Zukunft bald nicht mehr unsere Kunden sind (“customer churn”). Es gibt eine ganze Reihe von Untersuchungen dazu, z.B. die von ([lalwani_customer_2022?](#)). Die Forschis versuchen anhand von Daten und u.a. auch der linearen Regression vorherzusagen, welche Kunden abgewandert sein werden. Die Autoren berichten von einer Genauigkeit von über 80% in Ihrem (besten) Vorhersagemodell.

2.8. Wie man mit Statistik lügt

2.8.1. Pinguine drehen durch

Ein Forscher-Team untersucht Pinguine von der [Palmer Station, Antarktis](#). Das Team ist am Zusammenhang von Schnabellänge (*bill length*) und Schnabeltiefe (*bill depth*) interessiert, s. Abbildung [2.14](#).

Das Team hat in schweißtreibender eiszapfentreibender Arbeit $n = 344$ Tiere vermessen bei antarktischen Temperaturen. Hier sind die Daten:

```

penguins_path <- paste0(
  "https://vincentarelbundock.github.io/",
  "Rdatasets/csv/palmerpenguins/penguins.csv")

penguins <- read.csv(penguins_path)
  
```

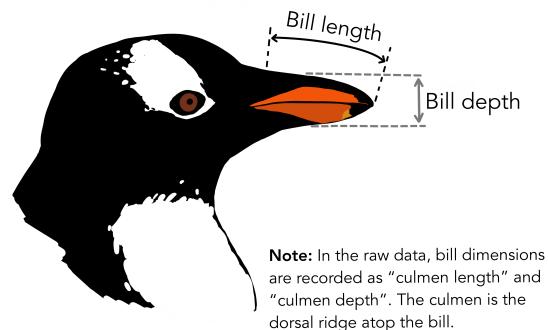
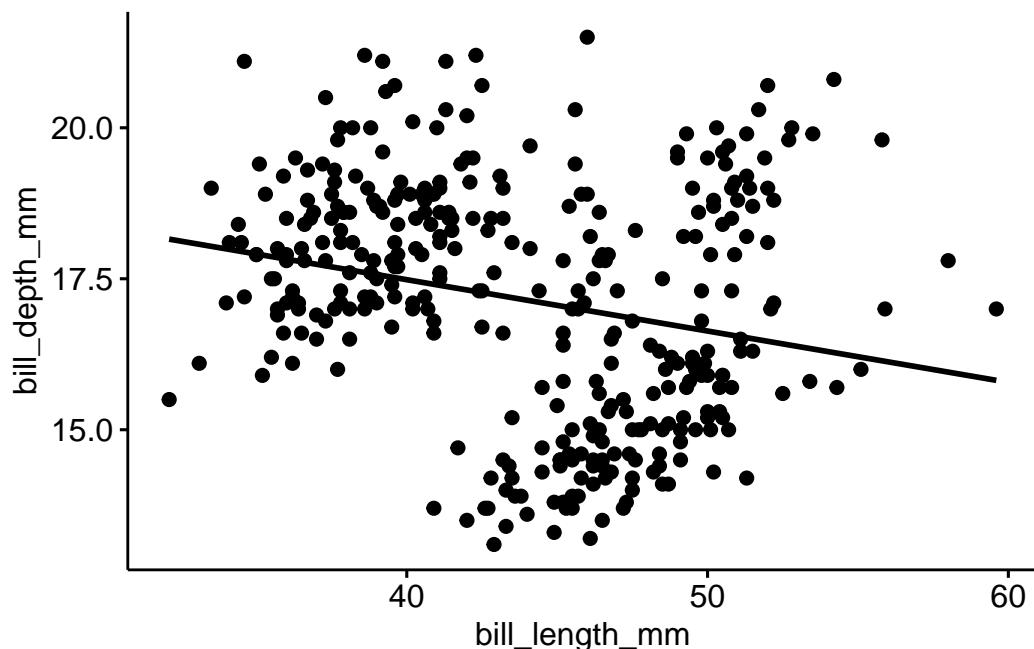


Abbildung 2.14.: Schnabellänge und Schnabeltiefe

2.8.2. Analyse 1: Gesamtdaten

Man untersucht, rechnet und überlegt. Ah! Jetzt haben wir es! Klarer Fall: Ein *negativer* Zusammenhang von Schnabellänge und Schnabeltiefe. Das könnte einen Nobelpreis wert sein. Schnell publizieren!

```
ggscatter(penguins, x = "bill_length_mm", y = "bill_depth_mm",
          add = "reg.line") # aus `ggpubr`
```



Hier sind die statistischen Details, s. Tabelle 2.19.

```
lm1 <- lm(bill_depth_mm ~ bill_length_mm, data = penguins)
```

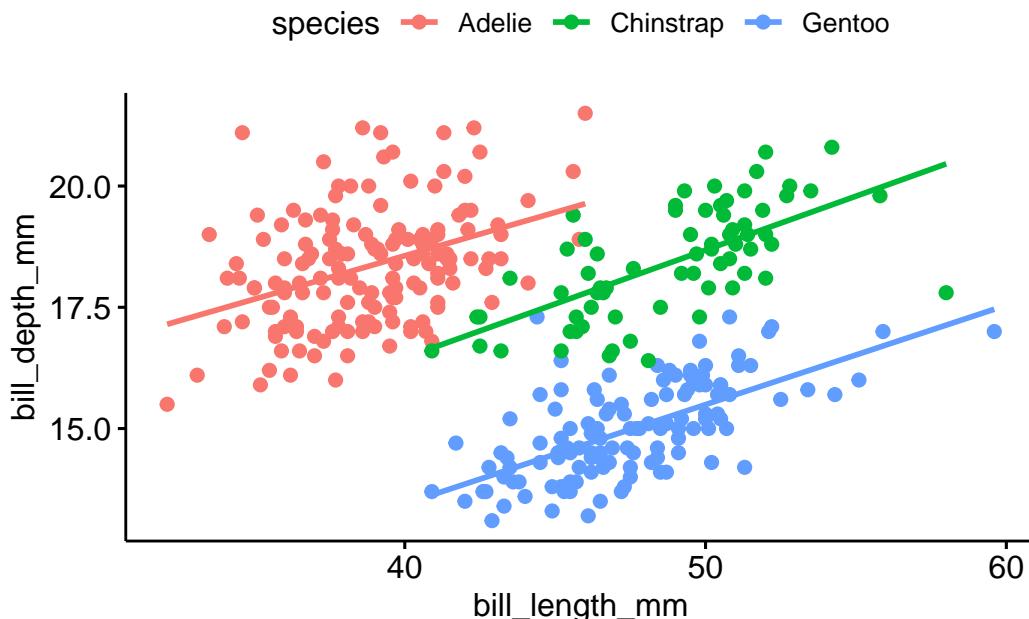
Tabelle 2.19.: Koeffizienten des Modells 1: Negativer Effekt von bill_length_mm

Parameter	Coefficient	SE	95% CI	t(340)	p
(Intercept)	20.89	0.84	(19.23, 22.55)	24.75	< .001
bill length mm	-0.09	0.02	(-0.12, -0.05)	-4.46	< .001

2.8.3. Analyse 2: Aufteilung in Arten (Gruppen)

Kurz darauf veröffentlicht eine verfeindete Forscherin auch einen Aufsatz zum gleichen Thema. Gleiche Daten. Aber mit *gegenteiligem* Ergebnis: Bei *jeder Rasse* von (untersuchten) Pinguinen gilt: Es gibt einen *positiven* Zusammenhang von Schnabellänge und Schnabeltiefe.

```
ggscatter(penguins, x = "bill_length_mm", y = "bill_depth_mm",
          add = "reg.line", color = "species")
```



Oh nein! Was ist hier nur los? Daten lügen nicht, oder doch?

Hier sind die statistischen Details der zweiten Analyse, s. Tabelle 2.20. Im zweiten Modell kam `species` als zweiter Prädiktor neu ins Modell (zusätzlich zur Schnabellänge).

```
lm2 <- lm(bill_depth_mm ~ bill_length_mm + species, data =
  ↵ penguins)
```

Tabelle 2.20.: Koeffizienten des Modells 2: Positiver Effekt von bill_length_mm

Parameter	Coefficient	SE	95% CI	t(338)	p
(Intercept)	10.59	0.68	(9.25, 11.94)	15.51	< .001
bill length mm	0.20	0.02	(0.17, 0.23)	11.43	< .001
species (Chinstrap)	-1.93	0.22	(-2.37, -1.49)	-8.62	< .001
species (Gentoo)	-5.11	0.19	(-5.48, -4.73)	-26.67	< .001

🔥 Daten alleine reichen nicht

Ohne Hintergrundwissen oder ohne weitere Analysen kann *nicht* entschieden werden, welche Analyse - Gesamtdaten oder Subgruppen - die richtige ist. Nicht-experimentelle Studien können zu grundverschiedenen Ergebnissen führen, je nachdem ob Prädiktoren dem Modell hinzugefügt oder weggelassen werden. □

2.8.4. Vorsicht bei der Interpretation von Regressionskoeffizienten

❗ Wichtig

Interpretiere nie Modellkoeffizienten kausal ohne ein Kausalmodell.□

Nur wenn man die Ursache-Wirkungs-Beziehungen in einem System kennt, macht es Sinn, die Modellkoeffizienten kausal zu interpretieren. Andernfalls lässt man besser die Finger von der Interpretation der Modellkoeffizienten und begnügt sich mit der Beschreibung der Modellgüte und mit Vorhersage³². Wer das nicht glaubt, der betrachte Abbildung 2.15, links.³³ Ei Forschi stellt das Modell m1: $y \sim x$ auf und interpretiert dann b1: "Ist ja klar, X hat einen starken positiven Effekt auf Y!".

In der nächsten Studie nimmt das Forschi dann eine zweite Variable, group (z.B. Geschlecht) in das Modell auf: m2: $y \sim x + g$. Oh Schreck! Jetzt ist b1 auf einmal nicht mehr stark positiv, sondern praktisch Null, und zwar in jeder Gruppe, s. Abbildung 2.15, rechts!

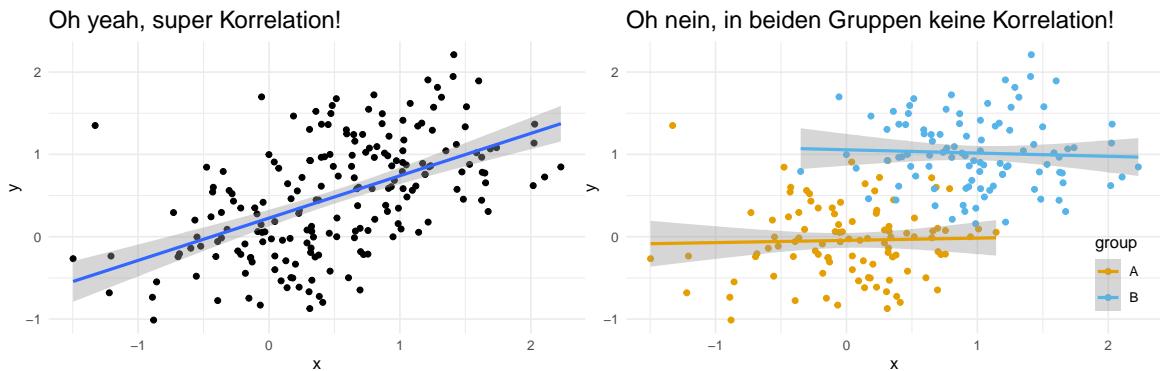
Dieses Umschwenken der Regressionskoeffizienten kann *nicht* passieren, wenn der Effekt "echt", also kausal, ist. Handelt es sich aber um "nicht echte", also nicht-kausale Zusammenhänge, um Scheinzusammenhänge also, so können sich die Modellkoeffizienten dramatisch verändern (sogar das Vorzeichen kann wechseln³⁴), wenn man das Modell verändert, also Variablen hinzufügt oder aus dem Modell entfernt.

³²synonym: Prognose

³³Quelle

³⁴das nennt man dann *Simpsons Paradox*

Wenn man die kausalen Abhängigkeiten nicht kennt, weiß man also nicht, ob die Zusammenhänge kausal oder nicht-kausal sind. Man weiß also nicht, ob die Modellkoeffizienten belastbar, robust, stichhaltig sind oder nicht.



- (a) Modell: $y \sim x$, starker Zusammenhang; b1 ist stark positiv
(b) Modell: $y \sim x + g$, in jeder der beiden Gruppen ist der Zusammenhang praktisch Null, b1 = 0

Abbildung 2.15.: Fügt man in ein Modell eine Variable hinzu, können sich die Koeffizienten massiv ändern. In beiden Diagrammen wurden die gleichen Daten verwendet.

Man könnte höchstens sagen, dass man (wenn man die Kausalstruktur nicht kennt) die Modellkoeffizienten nur *deskriptiv* interpretiert, z.B. "Dort wo es viele Störche gibt, gibt es auch viele Babies".³⁵ Leider ist unser Gehirn auf kausale Zusammenhänge geprägt: Es fällt uns schwer, Zusammenhänge nicht kausal zu interpretieren. Daher werden deskriptive Befunde immer wieder unzulässig kausal interpretiert – von Laien und Wissenschaftlern auch.

2.9. Fazit

In diesem Kapitel haben Sie lineare Modelle gelernt, die über einfache Modelle der Art $y \sim x$ hinausgehen. Dazu gehören multiple Modelle, das sind Modelle mit mehr als einer UV (Prädiktor) und auch Interaktionsmodelle. Außerdem haben Sie sich mit einem Datensatz von gesamtgesellschaftlichen Nutzen beschäftigt - sehr schön. Das Fallbeispiel zum Schluss war vielleicht erhellend insofern, als dass ein gutes Modell im Train-Sample nicht (notwendig) zu guten Vorhersagen im Test-Sample führt.

! Wichtig

Wenn Sie dran bleiben an der Statistik, wird der Erfolg sich einstellen, s. Abbildung 2.16. □

³⁵Das Störche-Babys-Beispiel passt auch zu Abbildung 2.15.

³⁶Quelle: imgflip, <https://imgflip.com/memegenerator/Distracted-Boyfriend>



(a) So ging es Ihnen gestern

(b) So wird es Ihnen morgen ergehen, wenn Sie dran bleiben

Abbildung 2.16.: Statistik, Sie und Party: Gestern und (vielleicht) morgen.³⁶

2.10. Fallstudien

Die folgenden Fallstudien zeigen auf recht anspruchsvollem Niveau (bezogen auf diesen Kurs) beispielhaft zwei ausführlichere Entwicklungen eines Prognosemodells.

Nutzen Sie diese Fallstudien, um sich intensiver mit der Entwicklung eines Prognosemodells auseinander zu setzen.

2.10.1. New Yorker Flugverspätungen 2023

[Vorhersage von Flugverspätungen](#)

2.10.2. Filmerlöse

[Vorhersagen von Filmerlösen](#)

2.11. Vertiefung

[Allison Horst](#) erklärt die lineare Regression mit Hilfe von Drachen. Sehenswert.

2.12. Aufgaben

Die Webseite datenwerk.netlify.app³⁷ stellt eine Reihe von einschlägigen Übungsaufgaben bereit. Sie können die Suchfunktion der Webseite nutzen, um die Aufgaben mit den folgenden Namen zu suchen:

- [interpret-koeff-lm](#)
- [Aussagen-einfache-Regr](#)
- [interpret-koeff](#)
- [regression1b](#)
- [mtcars-regr01](#)
- [regression1a](#)
- [lm1](#)
- [Regression5](#)
- [Regression6](#)
- [lm-mario1](#)
- [lm-mario2](#)
- [lm-mario3](#)
- [ausreisser1](#)
- [mario-compare-models](#)

2.13. Literaturhinweise

Wenn es ein Standardwerk für Regressionsanalyse geben könnte, dann vielleicht das neueste Buch von Andrew Gelman, ein bekannter Statistiker (Gelman, Hill, und Vehtari 2021). Sein Buch ist für Sozialwissenschaftler geschrieben, also nicht für typische Nerds, hat aber deutlich mehr Anspruch als dieses Kapitel.

Literatur

Forum, World Economic. 2020. „The Future of Jobs Report 2020“. CH-1223 Cologny/Geneva Switzerland: World Economic Forum. https://www3.weforum.org/docs/WEF_Future_of_Jobs_2020.pdf.

Gelman, Andrew, Jennifer Hill, und Aki Vehtari. 2021. *Regression and Other Stories*. Analytical Methods for Social Research. Cambridge: Cambridge University Press.

Okabe, Masataka, und Kei Ito. 2023. „Color Universal Design (CUD) / Colorblind Barrier Free“. 2023. <https://jfly.uni-koeln.de/color/>.

³⁷<https://datenwerk.netlify.app>