Kausalanalyse und Machinelles Lernen mit R

Ein Leitfaden für reproduzierbare Forschung

Martin C. Arnold, Christoph Hanck

2023-11-01

Inhaltsverzeichnis

1 Start

2 Statistische Programmierung mit R

Dieses Kapitel ist *nicht* als umfassende Einführung in R gedacht, sondern behandelt Kernfunktionen aus der Paketsammlung `tidyverse`. Wenngleich die Inhalte deutlich über ein Hallo-Welt-Beispiel[1] hinausgehen, betrachten wir hier grundlegene Funktionen für Datenmanipulation und Visualisierung. Diese sind Vorraussetzung für das Verständnis fortgeschrittener Code-Bausteine in späteren Kapiteln. Falls Sie bereits über Grundkenntnisse im Umgang mit `tidyverse` verfügen, können Sie dieses Kapitel überspringen. Sollten Sie nicht oder nur teilweise mit den hier gezeigten Befehlen vertraut sein oder keinerlei Erfahrung mit R haben, empfiehlt sich vorab eine Erarbeitung bzw. Wiederholung der Inhalte. Nachstehede Ressourcen finden wir hilfreich:

- Feedbackgestütze interaktive Übungsaufgaben bei DataCamp[2], bspw.
  - Einführung in R
  - Introduction to Data Visualization with ggplot2
  - Data Manipulation with dplyr
- Open-source-Literatur wie
  - der umfangreiche Leitfaden von Ellis und Mayer (2023)
  - R for Data Science
  - Hands-On Programming with R

Wir laden zunächst die Paketsammlung `tidyverse`. Für die Reproduktion mit dem R GUI oder mit RStudio muss das Paket vorab mit `install.packages()` installiert werden. In den interaktiven R-Konsolen in diesem Kapitel (und im Rest des Buchs) sind die benötigten R-Pakete bereits installiert *und* geladen, sofern nicht anders beschrieben.

```
# Paket tidyverse installieren
# install.packages("tidyverse")
```

[1] https://de.wikipedia.org/wiki/Hallo-Welt-Programm
[2] Ein Teil des hier angebotenen Katalogs (exlusive *Einführung in R*) ist kostenpflichtig.

```
# Paket 'tidyverse' laden
library(tidyverse)
```

Für das Verständnis von Code-Chunks ist es hilfreich, Zwischenergebnisse explizit zu evaluieren und in der Konsole auszugeben. Hierfür umschließen wir häufig Code-Zeilen mit runden Klammern. Der nächste Chunk illustriert dies für die Variable `x`.

```
# Variable definieren...
x <- pi
# ... und evaluieren
x

# Äquivalent:
(
  x <- pi
)
```

2.1 Lange, weite und "tidy" Datenformate

Wir betrachten den in Tabelle 2.1 dargestellten Datensatz *Klausurergebnisse*.

Tabelle 2.1: Datensatz *Klausurergebnisse*

Name	Mikro	Makro	Mathe
Tim	NA	1.3	3
Lena	1	3	NA
Ricarda	2	1.7	1.3
Simon	2.3	3.3	NA

Der Datensatz ist noch nicht in der R-Arbeitsumgebung verfügbar. Mit der Funktion `tribble()` können wir Tabelle 2.1 händisch als R-Objekt der Klasse `tibble` definieren

```
# 'klasurergebnisse' als tibble definieren
(
  klausurergebnisse <- tribble(
```

```
    ~Name,    ~Mikro, ~Makro, ~Mathe,
    "Tim",       NA,    1.3,    3.0,
    "Lena",     1.0,    3.0,     NA,
    "Ricarda",  2.0,    1.7,    1.3,
    "Simon",    2.3,    3.3,     NA
  )
)
```

`klausurergebnisse` enhält die Klausurnoten der vier Studierenden (Boebachtungen) spaltenweise *pro Modul*, d.h. die Spaltennamen `Mikro`, `Makro` und `Mathe` sind Ausprägungen der Variable *Modul*. Der Datensatz liegt also *nicht* im s.g. *Tidy-Format* vor.

> **Tidy-Format**
>
> Tidy-Format: Jede Spalte ist ***eine*** Variable, jede Reihe ist ***eine*** Beobachtung und jede Zelle enthält einen ***einen*** Wert. Datensätze im Tidy-Format sind häufig lang: Die Zeilendimension ist größer als die Spaltendimension.

Das Tidy-Format ist hilfreich für statistische Analysen mit `tidyverse`-Funktionen wie bspw. `ggplot()`. Wir nutzen die Funktion `tidyr::pivot_longer()`, um `klausurergebnisse` ein (langes) Tidy-Format zu transformieren.

```
# 'klausurergebnisse' in Tidy-Format überführen
(
  long <- pivot_longer(
    data = klausurergebnisse,
    cols = Mikro:Mathe,
    names_to = "Modul",
    values_to = "Note"
  )
)
```

Beachte, dass die Spalte `Name` die Zugehörigkeit der Ausprägungen (`Note`) jeder Variable (`Modul`) zu einer Beobachtung identifiziert. Mit dieser Information können wir den langen Datensatz wieder in das ursprüngliche (weite) Format zurückführen. Wir nutzen hierzu `tidyr::pivot_wider()`.

```
# langes Format in das Ausgangsformat transformieren
(
  wide <- pivot_wider(
    data = long,
    id_cols = "Name",
    names_from = "Modul",
    values_from = "Note"
  )
)
```

Wenn die Zuweisung von Zwischenergebnissen in Variablen nicht benötigt wird, kann eine Verkettung von Funktionsaufrufen die Verständlichkeit des Codes verbessern. Hierzu wird der Pipe-Operator `%>%` genutzt. Wir wiederholen die Transformationen mit den `tidyr::pivot_*`-Funktion bei Verwendung von `%>%`.

```
# langes Format mit %>%
(
  long <- klausurergebnisse %>%
    pivot_longer(
      cols = Mikro:Mathe,
      names_to = "Modul",
      values_to = "Note"
    )
)

# weites Format mit %>%
(
  wide <- long %>%
    pivot_wider(
      id_cols = "Name",
      names_from = "Modul",
      values_from = "Note"
    )
)
```

Ein Beispiel für den Nachteil des weiten Formats im Umgang mit `tidyverse`-Paketen ist die Funktion `tidyr::drop_na()`. Diese entfernt sämtliche *Zeilen* eines Datensatzes, die `NA`-Einträge (d.h. fehlende Werte) aufweisen. Beachte,

dass diese Operation im ursprünglichen weiten Format zum Entfernen ganzer Beobachtungen aus `wide` führt.

```
# NA-Einträge aus dem "weiten" Format entfernen
wide %>%
  drop_na()
```

Im Tidy-Format `long` hingegen bleiben die übrigen Informationen betroffener Beobachtungen erhalten.

```
# NA-Einträge aus dem "langen" Format entfernen
long %>%
  drop_na()
```

2.2 Pinguine und Pipes

In diesem Abschnitt zeigen wir die Verwendung häufig verwendeter `dplyr`-Funktionen (s.g. *Verben*) für die Transformation von Datensätzen: `mutate()`, `select()`, `filter()`,`summarise()` und `arrange()`.

Für die Illustration verwenden wir den Datensatz `penguins` aus dem R-Paket `palmerpenguins`. Dieser Datensatz wurde im Zeitraum 2007 bis 2009 von Dr. Kristen Gorman im Rahmen des *Palmer Station Long Term Ecological Research Program* zusammengetragen und enthält Größenmessungen für drei Pinguinarten, die auf den Inseln des Palmer-Archipels in der Antarktis beobachtet wurden.

```
# Paket 'palmerpenguins' installieren
# install.packages("palmerpenguins")

# Paket 'palmerpenguins' laden
library(palmerpenguins)
```

Mit `data()` wird der Datensatz in der Arbeitsumgebung verfügbar gemacht. Wir nutzen `glimpse()`, um einen Überblick zu erhalten.

```
# Datensatz in der Arbeitsumgebung verfügbar machen
data(penguins)
```

```
# Übersicht anzeigen lassen
glimpse(penguins)
```

2.2.1 `dplyr::mutate()`

Mit `mutate()` können bestehende Variablen überschrieben oder neue Variablen als Funktion bestehender Variablen definiert werden. `mutate()` operiert in der Spaltendimension des Datensatz.

Wir definieren eine neue Variable `body_mass_kg` als Transformation `body_mass_g/1000`.

```
# Neue Variable mit Gewicht in Kg definieren
penguins %>%
  mutate(
    body_mass_kg = body_mass_g/1000
  ) %>%
  glimpse()
```

Mit `across()` kann die dieselbe Operation auf mehrere Variablen angewendet werden.

Im nachstehenden Beispiel ändern wir den typ (`type`) der Variablen `species`, `island`, `sex` und `year` zu `character`.

```
# species, island, sex und year in Typ 'character'
↪  umwandeln
penguins %>%
  mutate(
    across(
      c(species, island, sex, year),
      .fns = as.character
    )
  ) %>%
  glimpse()
```

`transmute()` ist eine Variante von `mutate()`, die lediglich die transformierten Variablen beibehält.

```
# Nur transformierte Variablen behalten
penguins %>%
  transmute(
    body_mass_kg = body_mass_g/1000
  )
```

2.2.2 `dplyr::select()`

Mit `select()` werden Variablen aus dem Datensatz ausgewählt. Dies geschieht entweder über den Variablennamen...

```
# 'species' auswählen
penguins %>%
  select(species)
```

... oder über eine Indexmenge.[3]

```
# Teilmenge von Variablen per Index auswählen
penguins %>%
  select(
    c(1, 2, 3)
  )
```

Variablen können anhand eines Muster im Namen selektiert werden. Die Selektion von `ends_with("mm")` bezieht nur Variablen mit Endung `mm` im Namen ein:

```
# Nur in mm gemessene Variablen auslesen
penguins %>%
  select(
    ends_with("mm")
  )
```

Mit `where()` können wir Variablen aufgrund bestimmter Eigenschaften ihrer Ausprägungen selektieren.

[3] Hilfreich: `dplyr::pull()` selektiert eine Variable und wandelt diese in einen Vektor um.

```
# Nur numerische Variablen auswählen
penguins %>%
  select(
    where(is.numeric)
  )
```

2.2.3 `dplyr::filter()`

Das Verb `filter()` filtert den Datensatz in der Zeilendimension. So können Beobachtungen ausgewält werden, deren Merkmalsausprägungen bestimmte Kriterien erfüllen. Hierzu muss `filter()` ein logischer (`logical`) Ausdruck übergeben werden. Häufig erfolgt dies über Vergleichsoperatoren.

```
# Nur Pinguine mit bill_length_mm >= 39
penguins %>%
  filter(
    bill_length_mm >= 39
  )
```

```
# Nur Pinguine mit bill_length_mm <= 40
penguins %>%
  filter(
    bill_length_mm <= 40
  )
```

Oft ist es praktisch, mehrere Kriterien zu kombinieren.

```
# Kombinierter Filter -- Variante 1
penguins %>%
  filter(
    bill_length_mm >= 39 & bill_length_mm <= 40
  )
```

Analog: komma-getrennte Kriterien werden intern über den Und-Operator (`&`) verknüpft.

```
# Kombinierter Filter -- Variante 2
penguins %>%
  filter(
    bill_length_mm >= 39,
    bill_length_mm <= 40
  )
```

Ähnlich wie bei `select()` verwenden wir häufig nützliche Funktionen, welche die Interpretation des Codes erleichtern. `dplyr::between()` erlaubt filtern innerhalb eines Intervals.

```
# Filtern mit Hilfsfunktion
penguins %>%
  filter(
    between(
      bill_length_mm, left = 39, right = 40
    )
  )
```

Mit diesen Verben sind wir bereits in der Lage, den Datensatz gemäß folgender Vorschrift zu bereinigen:

1. Entfernen der Maßeinheiten aus den Variablennamen
2. Entfernen von Pinguinen mit fehlenden Werten (`NA`)
3. Entfernen von Pinguinen mit einem Gewicht *oberhalb* des 95%-Stichprobenquantils

```
# Schritt 1
(
  penguins_cleaned <- penguins %>%
    rename(
      bill_length = bill_length_mm,
      bill_depth  = bill_depth_mm,
      flipper_length = flipper_length_mm,
      body_mass = body_mass_g
    )
)
```

```
# Schritt 2
(
  penguins_cleaned <- penguins_cleaned %>%
    drop_na()
)
```

```
# Schritt 3
penguins_cleaned %>%
  filter(
    body_mass < quantile(body_mass, probs = .95)
  ) %>%
  glimpse()
```

Durch die Verkettung mit **%>%** können wir sämtliche Schritte für die Bereinigung ohne das Abspeichern von Zwischenergebnissen durchführen.

```
# Verketter Funktionsaufruf für Datensatzbereinigung
penguins_cleaned <- penguins %>%
  rename(
    bill_length = bill_length_mm,
    bill_depth  = bill_depth_mm,
    flipper_length = flipper_length_mm,
    body_mass = body_mass_g
  ) %>%
  drop_na() %>%
  filter(
    body_mass < quantile(body_mass, .95)
  )

penguins_cleaned %>%
  glimpse()
```

2.2.4 `dplyr::summarise()`

Das Verb `summarise()` fasst Variablen über Beobachtungen hinweg zusammen. Der nachstehende Code-Chunk erzeugt eine Tabelle mit Stichprobenmittelwert und -standardabweichung von `flipper_length_mm`.[4] Um zu vermeiden, dass

[4] `dplyr::summarise()` darf nicht mit `base::summary()` verwechselt werden!

die Auswertung aufgrund fehlender Werte (`NA`) in `flipper_length_mm` scheitert, lassen wir `NA`s mit `na.rm = TRUE` bei der Berechnung unberücksichtigt (wir verwenden weiterhin den unbereinigten Datensatz `penguins`).

```
# statistische Zusammenfassung mit 'summarise()'
penguins %>%
  select(flipper_length_mm) %>%
  summarise(
    mean = mean(flipper_length_mm, na.rm = TRUE),
    sd = sd(flipper_length_mm, na.rm = TRUE)
  )
```

Varianten von `summarise()` können über mehrere Variablen angewendet werden. Wir verwenden `across()` und `where()`, um lediglich numerische Variablen mit den in der liste definierten Funktionen zusammenzufassen. Beachte, dass `\(x) mean(x)` eine anonyme Funktion definiert.

```
penguins %>%
  summarise(
    across(
      where(is.numeric),
      .fns = list(
        mean = \(x) mean(x, na.rm = TRUE),
        sd = \(x) sd(x, na.rm = TRUE)
      )
    )
  ) %>%
  glimpse()
```

2.2.5 `dplyr::arrange()`

Mit `arrange()` können Datensätze in Abhängigkeit der beobachteten Ausprägungen von Variablen sortiert werden.

```
# Datensatz aufsteigend nach 'body_mass_g' sortieren
penguins %>%
  arrange(body_mass_g)
```

Die Funktion `dplyr::desc()` kehrt die Reihenfolge zu einer absteigenden Sortierung um.

```
# Absteigende Sortierung nach 'body_mass_g'
penguins %>%
  arrange(
    desc(body_mass_g)
  )
```

Komplexe Sortier-Muster werden durch Übergabe von Variablennamen in der gewünschten Reihenfolge erreicht.

```
# Erst Sortierung nach 'sex', dann gruppenweise absteigend
# nach 'body_mass_g' sortieren
penguins %>%
  arrange(
    sex, desc(body_mass_g)
  )
```

2.2.6 Operationen mit gruppierten Datensätzen

Für manche Transformationen ist eine Gruppierung der Daten hilfreich. Wir illustrieren nachfolgend die unterschiedlichen Verhaltensweisen ausgewählter Verben durch Vergleiche von gruppierten und nicht-gruppierten Anwendungen.

```
# Datensatz gruppieren
penguins_grouped <- penguins %>%
  group_by(species)

# Datensatz hat nun die Eigenschaft 'Groups'
glimpse(penguins_grouped)
```

`species` hat drei Ausprägungen. Entsprechend ist `penguins_grouped` nun in drei Gruppen eingeteilt.

Bei gruppierten Datensätzen fasst `summarise()` die Variablen pro Guppe zusammen.

```
# summarise -- ungruppiert:
penguins %>%
  summarise(
    across(
      where(is.numeric), \(x) mean(x, na.rm = T)
      )
    )
```

```
# summarise -- gruppiert:
penguins_grouped %>%
  summarise(
    across(
      where(is.numeric),
      ~ mean(., na.rm = T)
    )
  )
```

`mutate()` definiert bzw. transformiert für jede Gruppe separat. Im dies zu veranschaulichen, ziehen wir eine Zufallsstichprobe von 10 Pinguinen aus der Datensatz.

```
# Zufallsstichprobe generieren
set.seed(123)
(
  penguins_sample <- penguins %>%
    slice_sample(n = 10)
)
```

```
# mutate() -- ungruppiert:
penguins_sample %>%
  transmute(
    mean = mean(bill_length_mm)
  )
```

Für den ungruppierten Datensatz berechnet `mutate()` das Stichprobenmittel von `bill_length_mm` über *alle* zehn Datenpunkte und weißt diesen Wert jeweils in der Variable `mean` zu.

```
# mutate() -- gruppiert
penguins_sample %>%
  group_by(species) %>%
  transmute(
    mean = mean(bill_length_mm)
  )
```

Bei gruppierten Daten berechnet `mutate()` das Stichprobenmittel *pro Gruppe* und weist die Mittelwerte entsprechend zu.

2.3 Eine explorative Analyse mit `ggplot2`

Der bereinigte Datensatz `penguins_cleaned` eignet sich gut für eine graphische Auswertung mit dem R-Paket `ggplot2`, welches Bestandteil des `tidyverse` ist. Nachfolgend untersuchen wir Zusammenhänge zwischen den Körpermaßen der Pinguine.

Wir erstellen zunächst einen einfachen Punkteplot des Gewichts (`body_mass`) und der Schnabeltiefe (`bill_depth`).

```
# Punkteplot: body_mass vs. bill_depth
penguins_cleaned %>%
  ggplot(
    mapping = aes(
      x = body_mass,
      y = bill_depth
    )
  ) +
  geom_point()
```

Die Grafik zeigt einen positiven Zusammenhang zwischen dem Gewicht und der Schnabeltiefe. Als nächstes passen wir den Code so an, dass die Datenpunkte entsprechend der Art (`species`) eingefärbt sind.

```
# Punkteplot: Farbliche Darstellung verschiedener Arten
penguins_cleaned %>%
  ggplot(
    mapping = aes(
```

```
      x = body_mass,
      y = bill_depth,
      color = species
    )
  ) +
  geom_point()
```

Offenbar gibt es deutliche Unterschiede in der (gemeinsamen) Verteilung von Gewicht und Schnabeltiefe zwischen den verschiedenen Arten.

Um den Zusammenhang zwischen Gewicht und Schnabeltiefe zu untersuchen, schätzen wir lineare Regressionen

$$body\_mass = \beta_0 + \beta_1 bill\_depth + u$$

separat für jede der drei Pinguinarten mit der KQ-Methode. Anschließend zeichnen wir die geschätzten Regressionsgeraden ein.

```
# Lineare Regression per Art
penguins_cleaned %>%
  ggplot(
    aes(
      x = body_mass,
      y = bill_depth,
      color = species
    )
  ) +
  geom_point() +
  geom_smooth(method = "lm", se = F)
```

Die Schätzungen bekräftigen die Vermutung, dass der lineare Zusammenhang zwischen Gewicht und Schnabeltiefe sich nicht zwischen den verschiedenen Pinguinarten unterscheidet: Pinguine der Art *Gentoo* sind im Mittel schwerer als Pinguine der übrigen Arten, haben jedoch eine geringere Schnabeltiefe.

Der nachfolgende Code fügt der Grafik eine Regressionsline *über alle* Arten hinzu. Wir setzen hierbei das Argment `inherit_aes = FALSE` und legen damit fest, dass die Regression für `body_mass` und `bill_depth` ohne Differenzierung per `species` durchgeführt wird.

```
# Zusatz: Globale Regression
penguins_cleaned %>%
  ggplot(
    mapping = aes(
      x = body_mass,
      y = bill_depth,
      color = species
    )
  ) +
  geom_point() +
  geom_smooth(method = "lm", se = F) +
  # Regression für alle Datenpunkte
  geom_smooth(
    mapping = aes(
      x = body_mass,
      y = bill_depth
    ),
    method = "lm",
    se = F,
    inherit.aes = F
  )
```

Offenbar ist die vorherige Analyse per Spezies sinnvoller: Die Regression über alle Arten suggeriert einen negativen Zusammenhang zwischen Gewicht und Schnabeltiefe.

Facetting mit `facet_wrap()` erlaubt eine Untersuchung des Zusammenhangs je Insel (`island`), auf der die Messung erfolgt ist.

```
# Facettierung des per In
penguins_cleaned %>%
  ggplot(
    mapping = aes(
      x = body_mass,
      y = bill_depth,
      color = species)
  ) +
  geom_point() +
  geom_smooth(method = "lm", se = F) +
```

```
    facet_wrap(~ island)
```

Wir sehen, dass es hinsichtlich des Zusammenhangs von Gewicht und Schnabeltiefe keine wesentlichen Diskrepanzen zwischen den drei Inseln gibt. Darüber hinaus lässt sich anhand der Facetten leicht erkennen, wie die drei Arten über die Inseln verteilt sind.

3 Matching

```
#| context: setup
webr::install("dplyr")
webr::install("tidyr")

# create dataset directory
dir.create("datasets")
# Download the dataset
download.file(

↪  "https://raw.githubusercontent.com/mca91/kausal_data/main/darkmode.csv",
    'datasets/darkmode.csv',
)
```

```
#| context: setup
library(dplyr)
# make darkmode available using read.csv for now
# because there's some issue with readr::read_csv
# I can't fix right now
darkmode <- read.csv(
    file = "datasets/darkmode.csv",
    colClasses = c("numeric", "logical", "numeric",
↪  "numeric", "numeric")
)

options(pillar.bold = TRUE, pillar.subtle = FALSE)
```

In randomisierten kontrollierten Studien stellt eine randomisierte Behandlung sicher, dass die Individuen beider Gruppen im Mittel vergleichbar sind, dass heißt es gibt keine systematischen Unterschiede der Studiensubjekte hinsichtlich der Verteilung von Charakteristika zwischen den Gruppen. Dann ist es plausibel

eine beobachtete mittlere Differenz in der Outcome-Variable alleine auf die Behandlung zurückzuführen.

In der Praxis, insbesondere in ökonomischen Studien, sind randomisierte kontrollierte Studien aus ethischen und/oder finanziellen Gründen nicht durchführbar. Stattdessen werden nicht-experimentelle Daten genutzt, die jedoch nur sehr selten eine Vergleichbarkeit von Behandlungs- und Kontrollgruppe gewährleisten.

In diesem Kapitel betrachten wir Methoden, die in solchen Forschungsdesigns – bei hinreichenden Informationen über die Studiensubjekte – eine Schätzung kausaler Effekte ermöglichen, indem eine Vergleichbarkeit von Behandlungs- und Kontrollgruppe hergestellt wird. Dies kann durch eine gezielte Gewichtung von Beobachtungen anhand invididueller Merkmale bei der Schätzung des Behandlungseffekts erfolgen. Andere etablierte Methoden schätzen den kausalen Effekt nach Selektion von vergleichbaren Teilmengen von Subjekten beider Gruppen aus der ursprünglichen Stichprobe, sogenanntes *Matching*.

Da *Matching* ähnliche Beobachtungen basierend auf beobachtbaren Merkmalen vergleicht, kann die Wahrscheinlichkeit einer verzerrten Schätzung des kausalen Effekt durch falsche Modellspezifikationen geringer sein als für eine Schätzung des Effekts anhand multipler Regression. Weiterhin basieren Matching-Methoden nicht auf der Annahme eines linearen Zusammenhangs zwischen Kovariablen und der erklärenden Variable und können für die Schätzung unterschiedlicher Spezifikationen von Behandlungseffekten herangezogen werden.

Für die Gültigkeit eines Schätzers basierend auf *Matching* sind zwei Annahmen erforderlich.

1. ***Bedingte Unabhängigkeit.*** Seien $Y_i^{(0)}$ und $Y_i^{(1)}$ potentielle Ergebnisse der Outcome-Variable Y für ein Subjekt i mit $B_i = 0$ (keine Zuweisung zur Behandlung) beziehungsweise $B_i = 1$ (Behandlung) und X_i die beobachteten Kovariablen. Wir nehmen an, dass

$$\left\{Y_i^{(0)}, Y_i^{(1)}\right\} \perp B_i | X_i, \tag{3.1}$$

   d.h. die Behandlungszuweisung/-selektion ist unanabhängig von den potentielle Ergebnissen $Y_i^{(0)}$ und $Y_i^{(1)}$, wenn wir für die Kovariablen X kontrollieren.

2. ***Überlappung.*** Für jede mögliche Kombination von Kovariablen X_i gibt es eine positive Wahrscheinlichkeit < 1, sowohl zur Behandlungsgruppe

$(B_i = 1)$ als auch zur Kontrollgruppe $(B_i = 0)$ zugewiesen zu werden,

$$0 < \mathrm{P}(B_i = 1|X_i) < 1, \tag{3.2}$$

d.h. keine Beobachtung hat eine Behandlungswahrscheinlichkeit von exakt 0 oder 1.

Annahme 1 stellt sicher, dass die Zuweisung zur Behandlungsgruppe nach Kontrolle für die Kovariablen X als zufällig betracht werden kann. Somit ist es möglich den kausalen Effekt der Behandlung zu identifizieren, indem wir hinsichtlich der Kovariablen X ähnliche Subjekte (vgl. Annahme 2) aus Kontroll- und Behandlungsgruppe vergleichen.

Annahme 2 setzt vorraus, dass es eine ausreichende Überlappung in den Verteilungen der Kovariablen zwischen Behandlungs- und Kontrollgruppe gibt. Dann ist sichergestellt, dass für jedes Subjekt in einer Gruppe ein hinsichtlich seiner Charakteristika vergleichbares Subjekt in der anderen Gruppe geben kann, sodass ein Vergleich möglich ist.

3.1 *Balance*: Vergleichbarkeit von Behandlungs- und Kontrollgruppe

Der Lehrstuhl für Ökonometrie an der Universität Duisburg-Essen betreibt einen Ökonometrie-Blog und interessiert sich für den kausalen Effekt der Einführung eines darkmode auf die Verweildauer der User auf der Webseite. Die Webseite ist zwar nicht-kommerziell, hat sich allerdings insb. für die Aquise internationaler Studierender für den Studiengang MSc. Econometrics als wichtiges Marketing-Instrument erwiesen. Ein anprechendes Design wird daher als hoch-relevant erachtet.

Idealerweise sollte der Effekt des Design-Relaunches auf die Nutzungsintensität in einem kontrollierten randomisierten Experiment untersucht werden. Hierbei würden wir Nutzern zufällig das neue oder das alte Design zuweisen und den Effekt als Differnz des durchschnittlichen Verweildauer für die Gruppen bestimmen. Eine solche Studie ist jedoch aus technischen und finanziellen Gründen nicht realisierbar, sodass die Auswirkungen des darkmode mit vorliegenden nicht-experimenellen Nutzungsstatistiken für die Webseite geschätzt werden sollen.

Die Nutzungsstatistiken sind im Datensatz *darkmode.csv* enthalten und sollen der Analyse des Effekts des darkmode (`dark_mode`) auf die Verweildauer der

Leser auf der Webseite (`read_time`) dienen.

Tabelle 3.1 zeigt die Definitionen der Variablen in *darkmode.csv*.

Tabelle 3.1: Variablen im Datensatz *darkmode*

Variable	Beschreibung
read_time	Lesezeit (Minuten/Woche)
dark_mode	Indikator: Beobachtung nach Einführung darkmode
male	Indikator: Individuum männlich
age	Alter (in Jahren)
hours	Bisherige Verweildauer auf der Seite

Für die Analyse lesen wir zunächst den Datensatz *darkmode.csv* mit `readr::read_csv()` ein und verschaffen uns einen Überblick über die verfügbaren Variablen.

```
# Paket `tidyverse` laden
library(tidyverse)

# Datensatz 'darkmode' einlesen
darkmode <- read_csv(
  file = "datasets/darkmode.csv"
)
```

`dark_mode` hat den Typ `logical`. Mit `dplyr::mutate_all()` können wir komfortabel alle Spalten in den Typ `numeric` transformieren.

```
# Alle Variablen zu typ 'numeric' formatieren...
darkmode <- darkmode %>%
  mutate_all(.funs = as.numeric)

# ... und überprüfen
glimpse(darkmode)
```

```
Rows: 300
Columns: 5
$ read_time <dbl> 14.4, 15.4, 20.9, 20.0, 21.5, 19.5, 22.0,
17.4, 23.6, 15.7, ~
```

```
$ dark_mode <dbl> 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1,
1, 1, 0, 0, 0, ~
$ male      <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
1, 0, 1, 0, 0, ~
$ age       <dbl> 43, 55, 23, 41, 29, 64, 18, 53, 59, 53, 43,
38, 42, 23, 39, ~
$ hours     <dbl> 65.6, 125.4, 642.6, 129.1, 190.2, 185.3,
333.5, 279.3, 1302.~
```

Eine naive Schätzung des durchschnittlichen Behandlungseffekts (ATE) $\widehat{\tau}^{\text{naiv}}$ erhalten wir als Mittelwertdifferenz von `read_time` für die Behandlungsgruppe (`dark_mode == 1`) und die Kontrollgruppe (`dark_mode == 0`)

$$\widehat{\tau}^{\text{naiv}} = \overline{\text{read\_time}}_{\text{Behandlung}} - \overline{\text{read\_time}}_{\text{Kontrolle}}. \tag{3.3}$$

Diese Berechnung ist schnell mit R durchgeführt.

```r
# Naiver Schätzer für ATE:
# Differenz der Gruppen-Durchschnitte

# Outcome in Behandlungsgruppe
read_time_mTG <- darkmode %>%
  filter(dark_mode == 1) %>%
  pull("read_time")

# Outcome in Kontrollgruppe
read_time_mKG <- darkmode %>%
  filter(dark_mode == 0) %>%
  pull("read_time")

# Mittelwert-Differenz
mean(read_time_mTG) - mean(read_time_mKG)
```

```
[1] -0.4446331
```

Die Schätzung ergibt einen negativen Behandlungseffekt, mit der Interpreation, dass das neue Design zu einer Reduktion der Lesezeit um etwa 0.44 Minuten pro Woche führt. Dieses Ergebnis ist allerdings zweifelhaft, weil eine Isolierung des Behandlungseffekts aufgrund von Backdoor-Pfaden im DGP vermutlich nicht

gewähleistet ist. Ein Indikator hierfür sind systematische Unterschiede hinsichtlich von (möglicherweise unbeobachtbaren) Charakteristika von Kontrollgruppe und Behandlungsgruppe.

Da die User sich beim Aufrufen der Seite aktiv für oder gegen den das neue Design entscheiden müssen (und somit selektieren, ob Sie in der Behandlungs- oder Kontrollgruppe landen), liegt wahrscheinlich *Confounding* vor: Unsere Hypothese ist zunächst, dass männliche User eine durchschnittlich längere Lesezeit aufweisen *und* mit größerer Wahrscheinlichkeit auf das neue Design wechseln als nicht-männliche Leser. Dann ist `male` eine Backdoor-Variable. Diese Situation ist unter der Annahme, dass nur diese Faktoren den DGP bestimmen, in Abbildung 3.1 dargestellt.

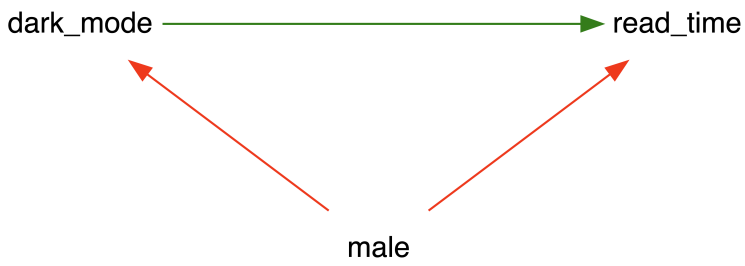


Abbildung 3.1: Backdoor durch 'male' im Website-Design-Bespiel

Der DGP in Abbildung 3.1 führt zu einer verzerrten Schätzung des kausalen Effekts von `dark_mode` auf `read_time` mit (3.3), wenn das Verhältnis von männlichen und nicht-männlichen Usern in Bahandlungs- und Kontrollgruppe nicht ausgeglichen ist. Wir überprüfen dies mit R.

```
# Anteile männlicher und nicht-männlicher User
(
  anteile <- darkmode %>%
  group_by(dark_mode) %>%
  summarise(
    gesamt = n(),
    ant_m = mean(male),
    ant_nm = 1 - ant_m,
    anz_m = sum(male),
    anz_nm = gesamt - anz_m
    )
```

```
)
```

```
# A tibble: 2 x 6
  dark_mode gesamt ant_m ant_nm anz_m anz_nm
      <dbl>  <int> <dbl>  <dbl> <dbl>  <dbl>
1         0    151 0.338  0.662    51    100
2         1    149 0.658  0.342    98     51
```

Die Zusammenfassung `anteile_m` zeigt, dass der Anteil männlicher User in der Behandlungsgruppe deutlich höher ist als in der Kontrollgruppe.

Matching eliminiert die Variation von `male` zwischen den Gruppen. Eine Möglichkeit hierfür ist die Gewichtung der Beobachtungen in der Kontrollgruppe entsprechend der Anteile von Männern und Nicht-Männern in der Behandlungsgruppe, sodass die Vergleichbarkeit mit der Behandlungsgruppe hinsichtlich des Geschlechts gewährleistet ist. Dies wird in der Literatur als *Balance* bezeichnet. Der Behandlungseffekt wird dann analog zu (3.3) geschätzt.

Die Gewichte für Beobachtungen in der Kontrollgruppe w_i werden berechnet als

$$w_i = \begin{cases} \text{ant\_m}_B/\text{anz\_m}_K, & \text{falls male}_i = 1 \\ \text{ant\_nm}_B/\text{anz\_nm}_K, & \text{sonst.} \end{cases} \tag{3.4}$$

Anhand der Formel für einen gewichteten Durchschnitt,

$$\overline{X}_w = \frac{\sum_i w_i \cdot X_i}{\sum_i w_i}, \tag{3.5}$$

berechnen wir die gewichteten Mittelwerte für `male` und `read_time` in der Kontrollgruppe.

```
# Anteile und Anzahlen aus `anteile` auslesen
anz_m_K <- anteile %>%
  filter(dark_mode == 0) %>% pull(anz_m)

anz_nm_K <- anteile %>%
  filter(dark_mode == 0) %>% pull(anz_nm)

ant_m_B <- anteile %>%
  filter(dark_mode == 1) %>% pull(ant_m)
```

```r
ant_nm_B <- anteile %>%
  filter(dark_mode == 1) %>% pull(ant_nm)
```

```r
# Gewichtete Mittel für Kontrollgruppe berechnen
(
gew_K <- darkmode %>%
  filter(dark_mode == 0) %>%
  select(read_time, male) %>%
  mutate(w = ifelse(
    male == 1,
    ant_m_B/anz_m_K,
    ant_nm_B/anz_nm_K)
    ) %>%
  summarise(
    male_k = sum(male * w) / sum(w),
    mean_read_time_wK = sum(read_time * w) / sum(w)
  )
)
```

```
# A tibble: 1 x 2
  male_k mean_read_time_wK
   <dbl>             <dbl>
1  0.658              18.1
```

Ein Vergleich des gewichteten Mittelwertes von `male` in der Kontrollgruppe mit dem Mittelwert in der Behandlungsgruppe (`male_k`) zeigt, dass die Gewichte die Variation in `male` zwischen beiden Gruppen eliminieren, sodass die Backdoor durch `male` geschlossen ist. Mit `wmean_read_time_K` haben wir einen entsprechend gewichteten Mittelwert der Verweildauer für die Kontrollgruppe berechnet. Wir schätzen den Behandlungseffekt nun als

$$\widehat{\tau}^{\mathrm{w}} = \overline{\mathrm{read\_time}}_{B} - \overline{\mathrm{read\_time}}_{w,K}. \tag{3.6}$$

```r
mean(read_time_mTG)  - gew_K$mean_read_time_wK
```

```
[1] 0.6383579
```

Entgegen der naiven Schätzung andhand von (3.3) erhalten wir nach Matching für `male` eine positive Schätzung des Behandlungseffekts von etwa 0.64.

Die Schätzung des Behandlungseffekts anhand von (3.6) entspricht dem geschätzten Koeffizienten $\widehat{\beta}_1$ aus einer gewichteten KQ-Regression im Modell

$$\text{read\_time} = \beta_0 + \beta_1 \text{dark\_mode} + u,$$

wobei die Beobachtungen der Kontrollgruppe wie in (3.4) gewichtet werden und $w_i = 1$ für Beobachtungen der Behandlungsgruppe ist. Wir überprüfen dies mit R.

```
darkmode_w <- darkmode %>%
  mutate(
    w = case_when(
      male == 1 & dark_mode == 0 ~ ant_m_B/anz_m_K,
      male == 0 & dark_mode == 0 ~ ant_nm_B/anz_nm_K,
      T ~ 1
    )
  )

lm(read_time ~ dark_mode, weights = w, data = darkmode_w)
↪  %>%
  summary()
```

```
Call:
lm(formula = read_time ~ dark_mode, data = darkmode_w, weights =
w)

Weighted Residuals:
     Min       1Q   Median       3Q      Max
-11.4302  -0.6929   0.0814   0.7230  12.8698

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  18.0918     3.4796   5.199 3.72e-07 ***
dark_mode     0.6384     3.4912   0.183    0.855
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 3.48 on 298 degrees of freedom
Multiple R-squared:  0.0001122, Adjusted R-squared:  -0.003243
F-statistic: 0.03343 on 1 and 298 DF,  p-value: 0.855
```

Der geschätzte Koeffizient von `dark_mode` entspricht $\widehat{\tau}^w$.

Da `male` eine binäre Variable ist, reduziert sich eine Beurteilung der Vergleichbarkeit der Verteilungen von `male` in Behandlungs- und Kontrollgruppe auf einen simplen Vergleich des Männeranteils beider Gruppen. In der Praxis gibt es meist eine Vielzahl potentieller Backdoor-Variablen, die zudem kontinuierlich verteilt sind. Es scheint plausibel, dass das Alter der Nutzer sowohl die Akzeptanz des Design-Updates als auch die Lesezeit beeinflusst. Die bisherige Verweildauer ist mindestens eine plausible Determinante der Lesezeit.

Der erweiterte DGP ist in Abbildung Abbildung 3.2 dargestellt, wobei der zusätzliche Backdoor-Pfad durch `age` ebenfalls mit roten Pfeilen gekennzeichnet sind.

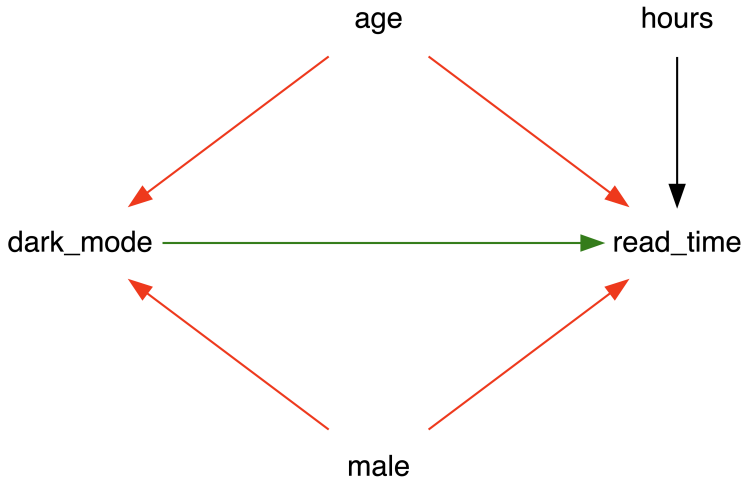


Abbildung 3.2: Erweiterter DGP im Website-Design-Beispiel

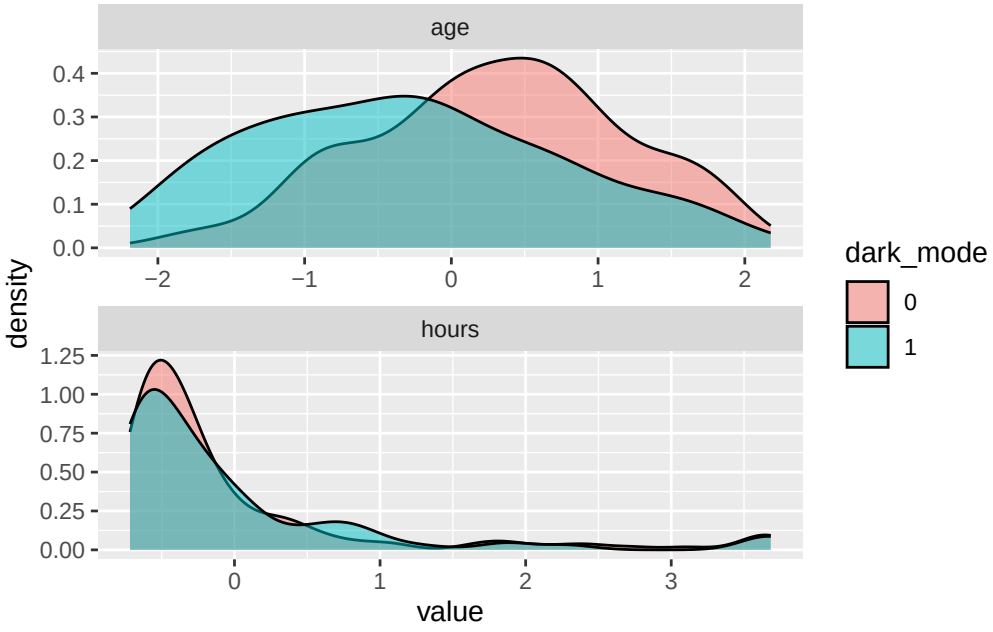
Die Beurteilung der *Balance* von Kontrollgruppe und Behandlungsgruppe kann durch eine grafische Gegenüberstellung der empirischen Verteilungen der Kovariablen beider Gruppen erfolgen. Wir visualisieren die empirischen Verteilungen mit `ggplot2`. Hierzu standardisieren wir `age` und `hours` zunächst mit `scale()`.

```r
# Datensatz für graphische Darstellung formatieren
darkmode_p <- darkmode %>%
  # Standardisierung mit 'scale()'
  mutate(
    dark_mode = as_factor(dark_mode),
    age = scale(age),
    hours = scale(hours)
  )

head(darkmode_p)
```

```
# A tibble: 6 x 5
  read_time dark_mode  male age[,1] hours[,1]
      <dbl> <fct>     <dbl>   <dbl>     <dbl>
1      14.4 0             0  0.0377    -0.591
2      15.4 0             1  1.11      -0.459
3      20.9 1             0 -1.74       0.684
4      20   0             0 -0.141     -0.451
5      21.5 1             0 -1.21      -0.316
6      19.5 0             0  1.91      -0.327
```

Für **age** und **hours** eignen sich die geschätzten Dichtefunktionen für einen Vergleich der Verteilungen in Behandlungs- und Kontrollgruppe.

```r
# Vergleich mit Dichteschätzungen
darkmode_p %>%
  select(dark_mode, hours, age) %>%
  # in langes Format überführen
  pivot_longer(cols = c(-dark_mode)) %>%

  ggplot(
    aes(x = value, fill = dark_mode)
    ) +
  geom_density(alpha = .5) +
  facet_wrap(
    facets = ~ name,
    scales = "free",
    nrow = 2
    )
```



Die graphische Analyse zeigt deutliche Unterschiede in den Verteilungen von `age` zwischen Kontroll- und Behandlungsgruppe. Für einen Beurteilung mit deskriptiven Statistiken wird häufig eine sogenannte *Balance Table* herangezogen. Wir berechnen diese für `age`, `hours` und `male` mit `cobalt::bal.tab()`

```
library(cobalt)

# Balance table mit 'cobalt::bal.tab()'
bal.tab(
  x = darkmode %>%
    select(age, hours, male),
  treat = darkmode$dark_mode,
   # berechne SMD für KG und TG:
  disp = "m",
  s.d.denom = "pooled"
)
```

```
Balance Measures
         Type   M.0.Un   M.1.Un Diff.Un
age   Contin.  46.0132  39.0940 -0.6469
hours Contin. 337.7775 328.5738 -0.0203
male   Binary   0.3377   0.6577  0.3200

Sample sizes
```

```
    Control Treated
All     151     149
```

Die Einträge `M.0.Un` und `M.1.Un` zeigen die jeweiligen Stichprobenmittelwerte der Variablen für Kontroll- und Behandlungsgruppe. `Diff.Un` gibt eine standardisierte Mittelwertdifferenz SMD an, wobei

$$SMD_j := \left(\overline{X}_{j,B} - \overline{X}_{j,K}\right) \Big/ \sqrt{\frac{1}{2}\left(\widehat{\text{Var}}(X_{j,B}) + \widehat{\text{Var}}(X_{j,K})\right)},$$

mit Stichprobenmitteln $\overline{X}_{j,B}$ und $\overline{X}_{j,K}$ und Stichprobenvarianzen $\widehat{\text{Var}}(X_{j,B})$ und $\widehat{\text{Var}}(X_{j,K})$ für eine kontinuierliche Kovariable j.[1] Obwohl es keinen einheitlichen Schwellenwert für die standardisierte Differenz gibt, der ein erhebliches Ungleichgewicht anzeigt, gilt für kontinuierliche Variablen eine standardisierte (absolute) Differenz von weniger als 0.1 als Hinweis auf einen vernachlässigbaren Unterschied zwischen den Gruppen.

Die Balance Table weist also auf einen vernachlässigbaren Unterschied für `hours` hin und bestätigt den aus den Grafiken abgeleiteten Eindruck einer relevanten Differenzen für `age`.

3.1.1 Mehrere Matching-Variablen und der Propensity Score

Bei mehreren Backdoor-Variablen kann eine Gewichtung anhand der Behandlungswahrscheinlichkeit (*Treatment Propensity*) erfolgen. Die Idee hierbei ist, dass der DGP wie in Abbildung 3.3 dargestellt werden kann.

Hierbei beeinflussen die Backdoor-Variablen *age* und *male* die Behandlungsvariable *dark_mode* lediglich durch die Behandlungswahrscheinlichkeit *Treatment Propensity*. Diese Darstellung zeigt, das die mehrdimensionale Information bzgl. der Ähnlichkeit von Subjekten hinsichtlich der beobachteten Kovariablen in einer einzigen Variable zusammengefasst werden kann. Die Backdoor-Pfade können daher geschlossen werden, indem wir Subjekte anhand von *Treatment Propensity* derart gewichten, dass beide Gruppen hinsichtlich der Verteilung der Behandlungswahrscheinlichkeit vergleichbar sind. Betrachte erneut (3.1) und beachte, dass

$$Y_i = Y_i^{(1)} D_i + Y_i^{(0)}(1 - D_i). \tag{3.7}$$

Rosenbaum und Rubin (1983) zeigen, dass es hinsichtlich (3.1) äquivalent ist für die *Treatment Propensity* $P_i(X_i) := P(B_i = 1 | X_i = x)$ zu kontrollieren,

[1] Siehe Austin (2011) für einen Überblick zu Balance-Statistiken.

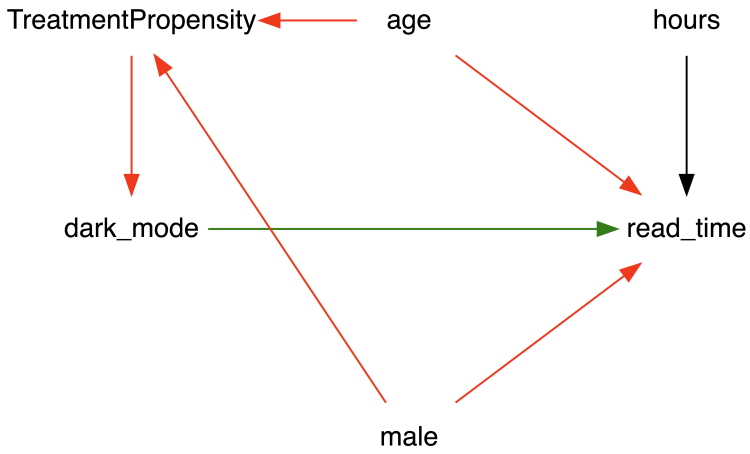


Abbildung 3.3: Propensity im Website-Design-Beispiel

d.h.

$$\left\{Y_i^{(1)}, Y_i^{(0)}\right\} \perp B_i | X_i \quad \Leftrightarrow \quad \left\{Y_i^{(1)}, Y_i^{(0)}\right\} \perp B_i | P_i(X_i). \tag{3.8}$$

Der Behandlungseffekt kann so als Differenz von gewichteten Gruppenmittelwerten berechnet werden, mit inversem Wahrscheinlichkeitsgewicht (IPW) $w_{i,B} = 1/P_i(X_i)$ für Beobachtungen in der Behandlungsgruppe und $w_{i,K} = 1/(1 - P_i(X_i))$ für Beobachtungen in der Kontrollgruppe,

$$\tau^{\text{IPW}} = \frac{1}{n} \sum_{i=1}^{n} \left[\frac{B_i Y_i}{P_i(X_i)} - \frac{(1 - B_i) Y_i}{1 - P_i(X_i)} \right]. \tag{3.9}$$

Grundsätzlich ist *TreatmentPropensity* eine nicht beobachtbare Variable und muss daher aus den Daten geschätzt werden. Eine geschätzte Behandlungswahrscheinlichkeiten $\widehat{P}_i(X_i)$ wird als *Propensity Score* bezeichnet. In der Praxis erfolgt die Schätzung von *Propensity Scores* meist mit logistischer Regression. Ein erwartungstreuer Schätzer des ATE ist

$$\widehat{\tau}^{\text{IPW}} = \frac{1}{n} \sum_{i=1}^{n} \left[\frac{B_i Y_i}{\widehat{P}_i(X_i)} - \frac{(1 - B_i) Y_i}{1 - \widehat{P}_i(X_i)} \right]. \tag{3.10}$$

Hirano, Imbens, und Ridder (2003) diskutieren Alternativen zu (3.10) für die Schätzung anderer Typen von Behandlungseffekten.

Wir schätzen nachfolgend die *Propensity Scores* für unser Anwendungsbeispiel, erläutern die Berechnung der Gewichte sowie die Schätzung von Be-

handlungseffekten mit gewichteter Regression. Hierbei betrachten wir eine Variante von (3.10) mit normalisierten Gewichten $\tilde{w}_{i,B} = w_{i,B} / \sum_i w_{i,B}$ und $\tilde{w}_{i,K} = w_{i,K} / \sum_i w_{i,K}$ die sich jeweils zu 1 summieren.[2] Dies ergibt den Hájek-Schätzer[3]

$$\widehat{\tau}_N^{\text{IPW}} = \frac{\sum_i \tilde{w}_{i,B} Y_i}{\sum_i \tilde{w}_{i,B}} - \frac{\sum_i \tilde{w}_{i,K} Y_i}{\sum_i \tilde{w}_{i,K}}. \tag{3.11}$$

Zunächst Schätzen wir ein logistisches Regressionsmodell mit `age`, `male` und `hours` als erklärende Variablen für `dark_mode`.

```
# Logit-Modell mit 'glm()' schätzen
(
  darkmode_ps_logit <- glm(
    formula = dark_mode ~ age + male + hours,
    data = darkmode,
    family = binomial
  )
)
```

```
Call:  glm(formula = dark_mode ~ age + male + hours, family =
binomial,
    data = darkmode)

Coefficients:
(Intercept)          age         male        hours
  2.330e+00   -7.330e-02    1.623e+00   -9.293e-05

Degrees of Freedom: 299 Total (i.e. Null);  296 Residual
Null Deviance:      415.9
Residual Deviance: 346.4    AIC: 354.4
```

Die *Propensity Scores* erhalten wir als angepasste Werte aus der Regression `darkmode_ps_logit` mit `fitted()`. Wir erweitern den Datensatz mit den Ergebnissen.

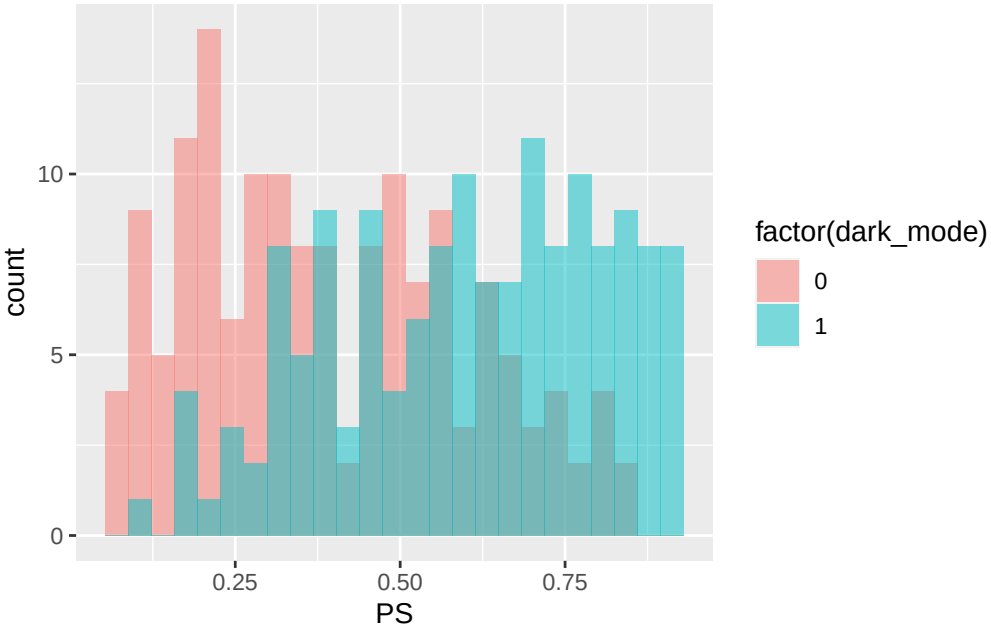
[2] Eine Normalisierung der Gewichte reduziert die Varianz des Schätzers, vgl. Hirano, Imbens, und Ridder (2003)
[3] Siehe Hájek (1971).

```r
# Datensatz um Propensity Scores erweitern
(
  darkmode_probs <-
    darkmode %>%
    mutate(
      PS = fitted(darkmode_ps_logit)
    )
)
```

```
# A tibble: 300 x 6
   read_time dark_mode  male   age  hours     PS
       <dbl>     <dbl> <dbl> <dbl>  <dbl>  <dbl>
 1      14.4         0     0    43   65.6 0.304
 2      15.4         0     1    55  125.  0.478
 3      20.9         1     0    23  643.  0.642
 4      20           0     0    41  129.  0.335
 5      21.5         1     0    29  190.  0.547
 6      19.5         0     0    64  185.  0.0849
 7      22           1     0    18  334.  0.727
 8      17.4         0     0    53  279.  0.171
 9      23.6         0     0    59 1303.  0.108
10      15.7         0     0    53   16.1 0.174
# i 290 more rows
```

Zur Beurteilung der Überlappung (vgl. Annahme (3.2) können wir die Verteilung der *Propensity Scores* nach Behandlungs-Indikator mit Histogrammen visualisieren.

```r
# Überlappung prüfen:
# Histogramme der PS nach Treatment-Indikator
darkmode_probs %>%
  ggplot(
    mapping = aes(
      x = PS,
      fill = factor(dark_mode)
    )
  ) +
  geom_histogram(
    alpha = .5,
```

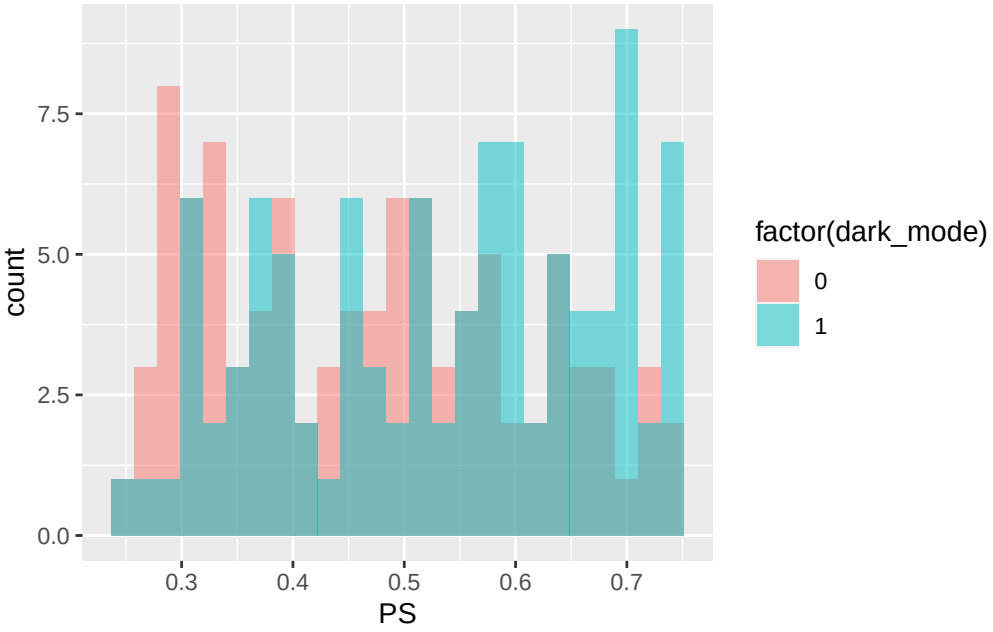



```
    bins = 25,
    position = "identity"
  )
```

Ein Vergleich der Histogramme zeigt, dass die Überlappung der *Propensity Scores* in der linken Flanken der Verteilungen der Kontrollgruppe und in der rechten Flanke der Behandlungsgruppe schlechter wird. Wir entfernen zunächst Beobachtungen aus der Stichprobe deren *Propensity Scores* wenig bzw. keine Überlappung aufweisen.

```
# Datensatz nach PS trimmen
darkmode_probs <- darkmode_probs %>%
  filter(
    between(
      x = PS,
      left = .25,
      right = .75
    )
  )
```

```
# Überlappung nach trimming prüfen:
# Dichteschätzung der PS nach Treatment-Indikator
```




```r
darkmode_probs %>%
ggplot(
  mapping = aes(
    x = PS,
    fill = factor(dark_mode))
  ) +
  geom_histogram(
    alpha = .5,
    bins = 25,
    position = "identity"
  )
```

IPWs anhand der *Propensity Scores* können schnell mit der Vorschrift

$$\text{IPW} = \frac{\text{dark\_mode}}{\text{PS}} + \frac{1 - \text{dark\_mode}}{1 - \text{PS}}, \tag{3.12}$$

berechnet werden.

```r
# Datensatz um IPWs erweitern
darkmode_IPW <- darkmode_probs %>%
  mutate(
    IPW = dark_mode / PS + (1 - dark_mode) / (1 - PS)
  )
```

```
  darkmode_IPW %>%
    select(IPW)
```

```
# A tibble: 194 x 1
     IPW
   <dbl>
 1  1.44
 2  1.91
 3  1.56
 4  1.50
 5  1.83
 6  1.38
 7  1.43
 8  1.47
 9  2.71
10  1.42
# i 184 more rows
```

Eine Schätzung des durchschnittlichen Behandlungseffekts gemäß (3.11) implementieren wir mit `dplyr`.

```
  darkmode_IPW %>%
    group_by(dark_mode) %>%
    mutate(w = IPW / sum(IPW)) %>%
    summarise(weighted_mean = sum(read_time * w)) %>%
    summarise(diff = diff(weighted_mean))
```

```
# A tibble: 1 x 1
   diff
  <dbl>
1  1.90
```

Diese Schätzung des Behandlungseffekts ist äquivalent zur gewichteten KQ-Schätzung anhand eines einfachen linearen Regressionsmodells.

```
  # Mit IPWs gewichteter KQ-Schaetzer berechnet den ATE
  model_ipw <- lm(
    formula = read_time ~ dark_mode,
```

```
  data = darkmode_IPW,
  weights = IPW
)

summary(model_ipw)
```

```
Call:
lm(formula = read_time ~ dark_mode, data = darkmode_IPW, weights
= IPW)

Weighted Residuals:
     Min       1Q   Median       3Q      Max
-18.5086  -4.4579   0.6096   4.1345  20.4566

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  17.9709     0.4942  36.362  < 2e-16 ***
dark_mode     1.9011     0.6952   2.735  0.00683 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.913 on 192 degrees of freedom
Multiple R-squared:  0.03749,   Adjusted R-squared:  0.03248
F-statistic: 7.478 on 1 and 192 DF,  p-value: 0.006829
```

Unsere Schätzung des ATE ist der geschätzte Koeffizient von `dark_mode`. Die ausgegebenen Standardfehler und Inferenzstatistiken sind jedoch *ungültig* aufgrund der Gewichtung mit IPWs, den inversen *geschätzten* Wahrscheinlichkeiten für eine Behandlung. Der Grund hierfür ist, dass die Berechnung der Standardfehler in `summary()` die zusätzliche Unsicherheit durch die geschätzen *Propensity Scores* nicht berücksichtigt! Später im Kapitel erläutern wir die Berechnung gültiger Standardfehler für IPW-Schätzer basierend auf *Propensity Scores* mit dem Bootstrap.

3.2 Matching-Verfahren

Das grundsätzliche Konzept von Matching wird in der nachstehenden interaktiven Grafik veranschaulicht. Hier betrachten wir beobachtete Ausprägungen von

zwei (unabhängig und identisch verteilten) Matching-Variablen für Subjekte in der Behandlungsgruppe (blau) sowie Kontrollgruppe (rot). Per Klick auf eine Beobachtung werden Matches aus der anderen Gruppe in grün kenntlich gemacht. Als Matches zählen sämtliche Beobachtungen der anderen Gruppe, deren Euklidische Distanz zu dem ausgewählten Punkt das über den Slider eingestellte Maximum *Caliper* nicht überschreitet.[4] Diese Region wird durch den gestrichelten Kreis gekennzeichnet. Die Grafik illustriert inbesondere, dass Beobachtungen mehrfach (s.g. Matching mit zurücklegen) oder gar nicht gematcht werden können.

`MatchIt::matchit()` nutzt standardmäßig Eins-zu-Eins-Matching (ohne Zurücklegen) von Beobachtungen der Treatment-Gruppe mit Beobachtungen der Kontrollgruppe. Die für das Matching zu verwendenden Variablen werden über das Argument `formula` als Funktion des Behandlungsindikators definiert. `matchit()` bereitet das Objekt für eine Schätzung des ATT mit einer geeigneten Funktionen, s. `?matchit` und hier insb. die Erläuterungen der Argumente `replace = F`, `ratio = 1` und `estimand = "ATT"` für Details. Mit `cobalt::balt.tab()` erhalten wir eine *balance table* für den gematchten Datensatz.

Wir zeigen als nächstes, wie `MatchIt::matchit()` für Matching anhand der Regressoren `age`, `hours`, und `male` in unserem Website-Beispiel für unterschiedliche Varianten durchgeführt werden kann.

Exaktes Matching

Exaktes Matching ordnet einem Subjekt aus der Behandlungsgruppe ein oder mehrere Subjekte aus der Kontrollgruppe zu, wenn die boebachteten Ausprägung der Matching-Variablen *exakt* übereinstimmen. Hierbei muss die 'Distanz' zwischen den Ausprägung der Matching-Variablen folglich 0 sein. Dieses Verfahren findet meist bei ausschließlich diskret verteilten Merkmalen Anwendung. Bei kontinuierlich verteilten Merkmalen (vgl. die obige interaktive Grafik) sind exakte Matches zwar theoretisch unmöglich, ergeben sich jedoch in der Praxis aus der Datenerfassung, bspw. durch Rundungsfehler. In `matchit()` erhalten wir exaktes Ein-zu-eins-Matching mit `method = "exact"`.

```
library(MatchIt)

# Exaktes Eins-zu-Eins-Matching durchführen
res_em <- matchit(
```

[4] Es handelt sich hierbei um einen Spezialfall von Matching anhand der Mahalanobis-Distanz.

```
  formula = dark_mode ~ age + male + hours,
  data = darkmode,
  estimand = "ATT",
  method = "exact"
)
```

```
Error in `matchit()`:
! No exact matches were found.
```

```
res_em
```

```
Error in eval(expr, envir, enclos): object 'res_em' not found
```

Aufgrund der kontinulierliche Verteilten Variable `hours` gibt es in unserem Website-Beispiel keine exakten Matches. Dieses Verfahren ist hier folglich ungeeignet.

Coarsened Exact Matching

Bei dieser Methode werden kontinuierliche Matching-Variablen grob (Engl. *coarse*) klassiert, ähnlich wie bei einem Histogram. Diese Diskretisierung ermöglicht es exakte Übereinstimmungen zwischen Behandlungs- und Kontrollgruppenbeobachtungen hinsichtlich ihrer klassierten Ausprägungen zu finden. Sowohl Behandlungs- als auch Kontrollbeobachtungen die mindestents einen exakten Match haben, werden Teil des gematchten Datensatzes. In `matchit()` wird Coarsened Exact Matching mit `method = "cem"` durchgeführt. Über das Argument `cutpoints` geben wir an, dass `hours` in 6 Klassen und `age` in 4 Klassen eingeteilt werden soll.[5] Mit `k1k = TRUE` erfolgt Eins-zu-eins-Matching: Bei mehreren exakten Matches wird die Beobachtung mit der geringsten Mahalanobis-Distanz (für die unklassierten Matching-Variablen) gewählt.

```
# Coarsened Exact Matching
res_CEM <- matchit(
  formula = dark_mode ~ age + male + hours,
  data = darkmode,
  estimand = "ATT",
  method = "cem",
  k2k = TRUE,
  cutpoints = list(
```

[5] Diese Werte wurden ad-hoc gewählt da sie zu einem guten Ergebnis führen.

```
    "hours" = 6,
    "age" = 4
  )
)
res_CEM
```

```
A matchit object
 - method: Coarsened exact matching
 - number of obs.: 300 (original), 164 (matched)
 - target estimand: ATT
 - covariates: age, male, hours
```

```
# Balance-Table Coarsened Exact Matching
bal.tab(res_CEM)
```

```
Balance Measures
         Type Diff.Adj
age   Contin.   0.0106
male   Binary   0.0000
hours Contin.  -0.0135

Sample sizes
          Control Treated
All           151     149
Matched        82      82
Unmatched      69      67
```

Mit Coarsened Exact Matching erhalten wir einen Datensatz mit 82 Beobachtungen und guter Balance.

Matching anhand der Mahalanobis-Distanz

Die Euklidische Distanz misst den direkten Abstand zwischen zwei Punkten und ist nicht invariant gegenüber Transformationen, insbesondere bei unterschiedlichen Skalierungen und bei Korrelation der Matching-Variablen. Die Mahalanobis-Distanz hingegen ist ein standardisiertes Distanzmaß, das unter Berücksichtigung der Varianz-Kovarianz-Struktur der Daten angibt, wie viele Standardabweichungen zwei Datenpunkte voneinander entfernt sind. Die Mahalanobis-Distanz ist invariant gegenüber linearen Transformationen (Skalierung, Translation und Rotation) der Daten und bietet ein genaueres Maß

für die Unähnlichkeit zweier Beobachtungen hinsichtlich ihrer Ausprägungen der Matching-Variablen.

Betrachte die Datenpunkte $P_1 = (X_1, Y_1)'$ und $P_2 = (X_2, Y_2)'$ für die Matching-Variablen X und Y. Die Mahalanobis-Distanz zwischen P_1 und P_2 ist definiert als

$$d_M(P_1,\, P_2) = \sqrt{(P_1 - P_2)' \boldsymbol{S}^{-1} (P_1 - P_2)},$$

wobei $\boldsymbol{S}$ die Varianz-Kovarianz-Matrix von X und Y ist. Die Mahalanobis-Distanz $d_M(\cdot, \cdot)$ ist also die Euklidische Distanz zwischen den standardisierten Datenpunkten.

Für beobachtete Daten ersetzen wir die Komponenten der Varianz-Kovarianz-Matrix durch Stichprobenmaße. Dies ergibt die Formel

$$\widehat{d}_M(P_1,\, P_2) = \sqrt{\begin{pmatrix} X_1 - X_2 \\ \tilde{Y}_1 - \tilde{Y}_2 \end{pmatrix}' \begin{pmatrix} \widehat{\mathrm{Var}}(X^2) & \widehat{\mathrm{Cov}}(X, Y) \\ \widehat{\mathrm{Cov}}(X, Y) & \widehat{\mathrm{Var}}(X^2) \end{pmatrix}^{-1} \begin{pmatrix} X_1 - X_2 \\ \tilde{Y}_1 - \tilde{Y}_2 \end{pmatrix}}.$$

Die nachstehende interaktive Grafik zeigt Beobachtungen zweier Matching-Variablen, die aus einer bivariaten Normalverteilung mit positiver Korrelation generiert wurden. Diese bivariate Verteilung ist identisch für Beobachtungen aus der Kontrollgruppe (rot) und Beobachtungen aus der Behandlungsgruppe (blau). Für die ausgewählte Beobachtung aus der Behandlungsgruppe (schwarzer Rand) werden potentielle Matches in der Kontrollgruppe innerhalb der vorgegebenen Mahalanobis-Distanz in Cyan kenntlich gemacht. Beachte, dass die Mahalanobis-Distanz Varianzen und Kovarianzen der Daten berücksichtigt, sodass die gematchten Beobachtungen in einem elliptischen Bereich um die betrachtete behandelte Beobachtung liegen. Eine Euklidische Distanz hingegen (gestrichelte Linie) ignoriert die Skalierung der Daten.

Für Eins-zu-Eins-Matching im Website-Beispiel anhand der Mahalanobis-Distanz mit `matchit()` setzen wir `distance = "mahalanobis"` und wählen `method = "nearest"`. Mit diesen Parametern wird jeder Behandlung aus der Behandlungsgruppe die gemäß d_M am ehesten vergleichbarste Beobachtung aus der Kontrollgruppe zugewiesen, wobei keine mehrfachen Matches zulässig sind.

```r
# 1:1 Mahalanobis-Distanz-Matching
res_maha <- matchit(
  formula = dark_mode ~ age + male + hours,
  data = darkmode,
  estimand = "ATT",
  distance = "mahalanobis",
  method = "nearest"
)
res_maha
```

```
A matchit object
 - method: 1:1 nearest neighbor matching without replacement
 - distance: Mahalanobis
 - number of obs.: 300 (original), 298 (matched)
 - target estimand: ATT
 - covariates: age, male, hours
```

```r
# Balance-Table für 1:1 Mahalanobis-Matching
bal.tab(res_maha)
```

```
Balance Measures
         Type Diff.Adj
age   Contin.  -0.5826
male   Binary   0.3154
hours Contin.   0.0106

Sample sizes
          Control Treated
All           151     149
Matched       149     149
Unmatched       2       0
```

Die Ergebnisse zeigen, dass für sämtliche 149 Beobachtungen aus der Behandlungsgruppe ein individueller Match in der Kontrollgruppe gefunden werden konnte. Es werden lediglich 2 Beobachtungen der 151 Beobachtungen in der Kontrollgruppe nicht gematcht.

Entsprechend zeigt die Balance-Table eine ähnliche Diskrepanz beider Gruppen hinsichtlich der Matching-Variablen an.

Mahalanobis-Distanz mit Caliper .25 für Propensity Scores basierend auf logistischer Regression

Für eine strengeres Matching-Kriterium kann ein *Caliper*, d.h. eine maximal zulässige Distanz, herangezogen werden. Die Mahalanobis-Distanz hat jedoch keine einheitliche Skala: Ob eine Distanz als groß oder klein betrachten werden kann, hängt von der Anzahl der Matching-Variablen und dem Überlappungsgrad zwischen den Gruppen ab. Daher wird die Beschränkung durch einen Caliper nicht auf $\widehat{d}_M$ sondern auf Propensity Scores angewendet.

Im nächsten Code-Beispiel spezifizieren wir mit `distance = "glm"`, dass Propensity Scores gemäß der Vorschrift in `formula` geschätzt werden. Mit `mahvars = ~ age + male + hours` legen wir die Matching-Variablen für die Berechnung von $\widehat{d}_M$ fest. `caliper = .25` legt fest, dass lediglich Beobachtungen der Kontrollgruppe bei einer absoluten Differenz der Propensity Scores von höchstens 0.25 Standardabweichungen als Match für eine Beobachtung in der Behandlungsgruppe qualifiziert sind.

```
# Mahalanobis-Matchig mit PS-Caliper
res_mahaC <- matchit(
  formula = dark_mode ~ age + male + hours,
  data = darkmode,
  distance = "glm",
  estimand = "ATT",
  method = "nearest",
  mahvars = ~ age + male + hours,
  caliper = .25
)
res_mahaC
```

```
A matchit object
 - method: 1:1 nearest neighbor matching without replacement
 - distance: Mahalanobis [matching]
             Propensity score [caliper]
             - estimated with logistic regression
 - caliper: <distance> (0.058)
 - number of obs.: 300 (original), 208 (matched)
 - target estimand: ATT
 - covariates: age, male, hours
```

```r
# Balance Table
bal.tab(res_mahaC)
```

```
Balance Measures
             Type Diff.Adj
distance Distance   0.1812
age       Contin.  -0.1176
male       Binary   0.0481
hours     Contin.  -0.0001

Sample sizes
          Control Treated
All           151     149
Matched       104     104
Unmatched      47      45
```

Die Balance-Table zeigt einen deutlichen Effekt der Beschränkung qualifizierter Beobachtungen durch `caliper = .25`: Aufgrund der oberen Grenze für die Propensity-Score-Differenz von 0.058 wird für lediglich 104 Beobachtungen aus der Behandlungsgruppe ein individueller Match in der Kontrollgruppe gefunden.[6] Weiterhin finden wir eine verbesserte Balance für den gematchten Datensatz.

Matching mit Propensity Scores und Caliper

Eine gängige Variante ist Matching ausschließlich anhand von Propensity Scores innerhalb eines Calipers.

```r
# 1:1 Matching mit PS und Caliper
res_PSC <- matchit(
  formula = dark_mode ~ age + male + hours,
  data = darkmode,
  estimand = "ATT",
  distance = "glm",
  method = "nearest",
  caliper = .25
)
res_PSC
```

[6] Die durch `caliper` implizierte Obergrenze ergibt sich als `.25 * sd(fitted(darkmode_ps_logit))`.

```
A matchit object
 - method: 1:1 nearest neighbor matching without replacement
 - distance: Propensity score [caliper]
             - estimated with logistic regression
 - caliper: <distance> (0.058)
 - number of obs.: 300 (original), 208 (matched)
 - target estimand: ATT
 - covariates: age, male, hours
```

```r
# Balance Table
bal.tab(res_PSC)
```

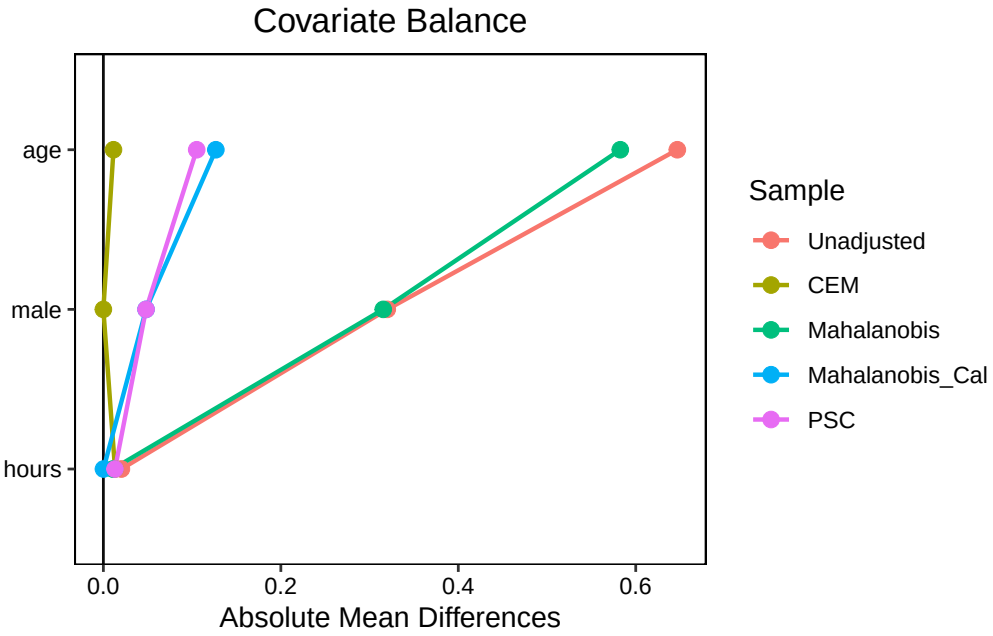
```
Balance Measures
             Type Diff.Adj
distance Distance   0.1640
age       Contin.  -0.0976
male       Binary   0.0481
hours     Contin.   0.0134

Sample sizes
          Control Treated
All           151     149
Matched       104     104
Unmatched      47      45
```

Laut Balance-Table führt Eins-zu-Eins-Matching basierend auf Propensity Scores zu einem Datensatz mit 104 gematchten Beobachtungen in der Behandlungsgruppe. Hinsichtlich der standardisierten Mittelwertdifferenz (`Diff.Adj`) erzielt diese Methode die beste Balance unter den betrachteten Ansätzen.

Vergleich der Balance verschiedener Verfahren mit Love-Plot

Standardisierte Mittelwertdifferenzen für verschiedene Matching-Verfahren können grafisch mit einem Love-Plot (Love 2004) veranschaulicht werden. Hierzu nutzen wir `cobalt::love.plot()` und übergeben die mit `matchit()` generierten Objekte im Argument `weights`.

```r
# Love-Plot für
love.plot(
  x = dark_mode ~ age + male + hours,
```




```
  weights = list(
    CEM = res_CEM,
    Mahalanobis = res_maha,
    Mahalanobis_Cal = res_mahaC,
    PSC = res_PSC
  ),
  data = darkmode,
  line = T,
  # absolute Mittelwertdifferenz plotten
  abs = T
)
```

Die Grafik zeigt, dass Coarsened Exact Matching (CEM) unter allen betrachteten Verfahren die Stichprobe mit der besten Balance ergibt. Diesen gematchten Datensatz erhalten wir mit `MatchIt::match.data()`.

```
# gematchten Datensatz zuweisen
darkmode_matched_CEM <- match.data(res_CEM)
head(darkmode_matched_CEM)
```

```
# A tibble: 6 x 7
  read_time dark_mode  male   age hours weights subclass
      <dbl>     <dbl> <dbl> <dbl> <dbl>   <dbl> <fct>
```

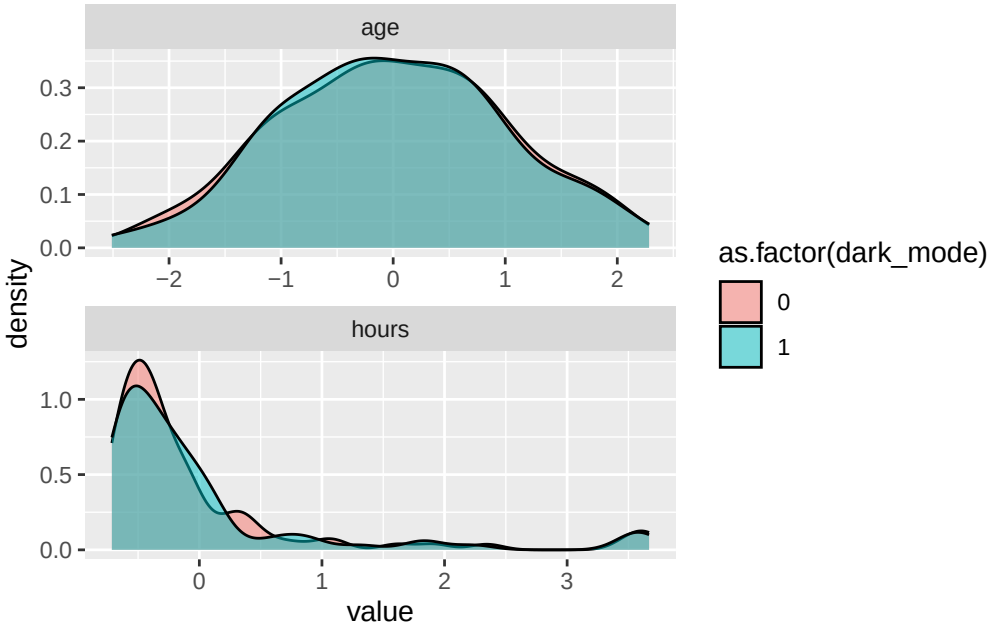
```
1      15.4          0     1    55  125.        1 26
2      20.9          1     0    23  643.        1 70
3      21.5          1     0    29  190.        1 79
4      22            1     0    18  334.        1 80
5      17.4          0     0    53  279.        1 11
6      20.4          0     0    43  138.        1 9
```

`darkmode_matched` enthält Gewichte (`weights`) für die jeweilige Gruppe zu denen gemachte Beobachtungen gehören (`subclass`). Dies ist relevant, falls Beobachtungen mehrfach gematcht werden. Wegen Eins-zu-eins-Matching *ohne* Zurücklegen gibt es in unserem Beispiel 82 Beobachtungspaare und sämtliche Gewichte sind 1. Die Berücksichtigung der Gewicht in den nachfolgenden Aufrufen von Schätzfunktionen (bspw.`lm()`) ist daher nicht nötig und erfolgt lediglich zur Illustration der grundsätzlichen Vorgehensweise.

Eine Wiederholung der grafischen Analyse in Kapitel 3.1 zeigt eine deutlich verbesserte Vergleichbarkeit hinsichtlich der Verteilung der Matching-Variablen in `darkmode_matched`.

```
darkmode_matched_CEM %>%
  group_by(dark_mode) %>%
  select(age, hours) %>%
  mutate_all(scale) %>%
  pivot_longer(cols = c(-dark_mode)) %>%

  ggplot(
    aes(x = value, fill = as.factor(dark_mode))
  ) +
  geom_density( alpha = .5) +
  facet_wrap(
    facets = ~ name,
    scales = "free",
    nrow = 3
  )
```

```
darkmode_matched_CEM %>%
  group_by(dark_mode) %>%
  mutate(
    male = as.factor(male),
```



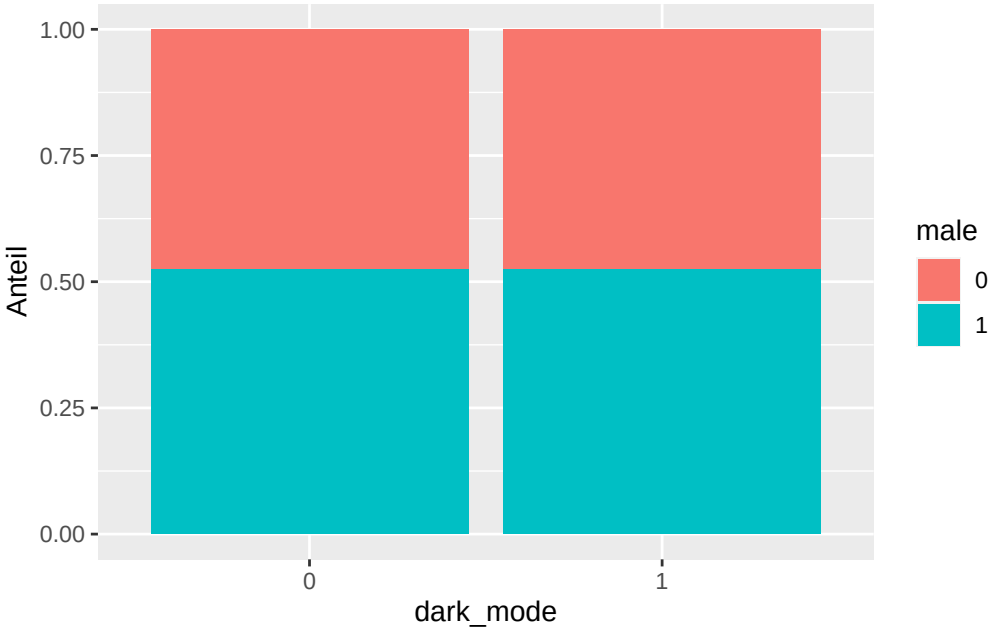

```
  dark_mode = as.factor(dark_mode)
) %>%

ggplot(
  aes(x = dark_mode, fill = male)
) +
geom_bar(position = "fill") +
ylab("Anteil")
```

Wir beobachten eine bessere Balance bei **age** und **hours**. Inbesondere ist **male** für Kontroll- und Behandlungsgruppe ausgeglichen!

3.3 Schätzung und Inferenz für den Behandlungseffekts nach Matching

Wir schätzen nun den Behandlungseffekt von **dark_mode** auf **read_time** für die mit CEM und Propensity Score Matching ermittelten Datensätze mittels linearer Regression und vergleichen mit einer Regression anhand des ursprünglichen Datensatzes.

```
# ATT mit linearem Modell schätzen: CEM Datensatz
ATT_mod_CEM <- lm(
```




```
  formula = read_time ~ age + male + hours + dark_mode,
  data = darkmode_matched_CEM,
  weights = weights
)
summary(ATT_mod_CEM)
```

```
Call:
lm(formula = read_time ~ age + male + hours + dark_mode, data =
darkmode_matched_CEM,
    weights = weights)

Residuals:
    Min      1Q  Median      3Q     Max
-8.9310 -2.3856 -0.0194  2.5407 14.0020

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 16.6088953  1.5255529  10.887  < 2e-16 ***
age          0.0380982  0.0344699   1.105  0.27072
male        -4.0915725  0.6858131  -5.966 1.53e-08 ***
hours        0.0050129  0.0006977   7.185 2.45e-11 ***
dark_mode    1.6532234  0.6149439   2.688  0.00794 **
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.937 on 159 degrees of freedom
Multiple R-squared:  0.414, Adjusted R-squared:  0.3992
F-statistic: 28.08 on 4 and 159 DF,  p-value: < 2.2e-16
```

```r
# Datensatz für Propensity Score Matching zuweisen
darkmode_matched_PSC <- match.data(res_PSC)

# ATT mit linearem Modell schätzen: PSM Datensatz
ATT_mod_PSC <- lm(
  formula = read_time ~ age + male + hours + dark_mode,
  data = darkmode_matched_PSC,
  weights = weights
)
summary(ATT_mod_PSC)
```

```
Call:
lm(formula = read_time ~ age + male + hours + dark_mode, data =
darkmode_matched_PSC,
    weights = weights)

Residuals:
   Min     1Q Median     3Q    Max
-9.972 -2.605 -0.044  2.587 14.951

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 17.3654519  1.2762648  13.606  < 2e-16 ***
age          0.0283941  0.0301484   0.942  0.34741
male        -3.9858702  0.6480072  -6.151 4.03e-09 ***
hours        0.0046119  0.0006685   6.899 6.53e-11 ***
dark_mode    1.6346636  0.5769812   2.833  0.00507 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.141 on 203 degrees of freedom
```

```
Multiple R-squared:  0.3171,    Adjusted R-squared:  0.3037
F-statistic: 23.57 on 4 and 203 DF,  p-value: 5.098e-16
```

```r
# ATT mit linearem Modell für vollständigen Datensatz
↪  schätzen
ATT_mod_org <- lm(
  formula = read_time ~ age + male + hours + dark_mode,
  data = darkmode
)
summary(ATT_mod_org)
```

```

Call:
lm(formula = read_time ~ age + male + hours + dark_mode, data =
darkmode)

Residuals:
     Min       1Q   Median       3Q      Max
-10.2697  -2.6710   0.0164   2.5909  14.5739

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 16.859075   1.082303  15.577  < 2e-16 ***
age          0.051332   0.022215   2.311  0.02154 *
male        -4.485545   0.498957  -8.990  < 2e-16 ***
hours        0.004348   0.000516   8.427 1.58e-15 ***
dark_mode    1.385810   0.523793   2.646  0.00859 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.03 on 295 degrees of freedom
Multiple R-squared:  0.3434,    Adjusted R-squared:  0.3345
F-statistic: 38.57 on 4 and 295 DF,  p-value: < 2.2e-16
```

Beachte, dass für die gematchten Datensätze jeweils ein durchschnittlicher Behandlungseffekt für die Beobachtungen *mit* erfolgter Behandlung ermittelt wird: In sämtlichen oben gezeigten Verfahren werden mit `estimand = "ATT"` vergleichbarere Kontrollbeobachtungen für die behandelten Beobachtungen ermittelt. Wir schätzen den Effekt der Behandlung, indem wir die Ergebnisse von behandelten Personen mit denen von gematchten (d.h.ähnlichen) Personen

vergleichen, die keine Behandlung erhalten haben. Diese Vergleichsgruppe dient als Ersatz für den hypothetischen Zustand der Behandlungsgruppe, wenn keine Behandlung erfolgt wäre. Dies ist die Definition eines ATT — ein average treatment effect *on the treated*.

Für Matching-Verfahren (`ATT_mod`) sind die von `summary()` berechneten Standardfehler (und damit Konfidenzintervalle, t-Statistiken und p-Werte) für den Behandlungseffekt *grundsätzlich ungültig*. Je nach Matching-Verfahren liegen unterschiedliche Quellen von Schätzunsicherheit vor, die bei der Berechnung von Standardfehlern zusätzlich zu der "üblichen" Stichproben-Variabilität berücksichtig werden müssen, bspw. aufgrund der Schätzung von Propensity Scores und der Matching-Prozess ansich. Wir nutzen daher nachfolgende Funktionen gem. Empfehlungen aus der aktuellen Forschung für Standardfehlerberechnung.

```
library(marginaleffects)

# Inferenz: Multiple Regression bei ungematchten
↪  Beobachtungen
(
  sum_orig <- avg_comparisons(
    model = ATT_mod_org,
    variables = "dark_mode",
    # Heteroskedastie-robuste SE:
    vcov = "HC3",
    # Identifizierung der Kontrollgruppe:
    newdata = subset(darkmode, dark_mode == 1)
  )
)
```

```
     Term Contrast Estimate Std. Error    z Pr(>|z|)   S 2.5 %
     97.5 %
dark_mode    1 - 0     1.39      0.537 2.58  0.00988 6.7 0.333
2.44

Columns: term, contrast, estimate, std.error, statistic,
p.value, s.value, conf.low, conf.high
Type:  response
```

```r
# Inferenz: Multiple Regression bei CEM
(
  sum_CEM <- avg_comparisons(
  model = ATT_mod_CEM ,
  variables = "dark_mode",
  # Cluster-robuster SE
  vcov = ~ subclass,
  newdata = subset(darkmode_matched_CEM, dark_mode == 1),
  wts = "weights"  # = 1
  )
)
```

```
     Term Contrast Estimate Std. Error    z Pr(>|z|)   S 2.5 %
     97.5 %
dark_mode    1 - 0     1.65      0.549 3.01  0.00262 8.6 0.576
2.73

Columns: term, contrast, estimate, std.error, statistic,
p.value, s.value, conf.low, conf.high
Type:  response
```

```r
# Inferenz: Multiple Regression bei PSM
(
  sum_PSC <- avg_comparisons(
    model = ATT_mod_PSC ,
    variables = "dark_mode",
    vcov = ~ subclass,
    newdata = subset(darkmode_matched_PSC, dark_mode == 1),
    wts = "weights"  # = 1
  )
)
```

```
     Term Contrast Estimate Std. Error    z Pr(>|z|)   S 2.5 %
     97.5 %
dark_mode    1 - 0     1.63      0.565 2.89  0.00381 8.0 0.527
2.74
```

	(1)	(2)	(3)
dark_mode	1.386	1.653	1.635
	(0.537)	(0.549)	(0.565)
Num.Obs.	300	164	208
R2	0.343	0.414	0.317
R2 Adj.	0.334	0.399	0.304
AIC	1694.6	921.9	1188.4
BIC	1716.9	940.5	1208.4
Log.Lik.	−841.321	−454.933	−588.180
F	38.569	28.079	23.567
RMSE	4.00	3.88	4.09
Std.Errors	HC3		

```
Columns: term, contrast, estimate, std.error, statistic,
p.value, s.value, conf.low, conf.high
Type:  response
```

```
library(modelsummary)
modelsummary(
  models = list(sum_orig, sum_CEM, sum_PSC)
)
```

3.4 Inferenz für ATT/ATE: Propensity-Score-Matching mit Bootstrap

Bei Matching mit Zurücklegen besteht zusätzliche Unsicherheit durch Zurücklegen, d.h. Beobachtungen aus der Kontroll-Gruppe können mehrfach als Match für Beobachtungen aus der Treatment-Gruppe genutzt werden. Mit `summary()` berechnete Standardfehler berücksichtigen dies nicht!

Ein Bootstrap-Verfahren generiert mit Resampling (wiederholtes Ziehen mit Zurücklegen) aus dem Original-Datensatz (viele) künstliche Datensätze, für die der Schätzer (d.h. das gesamte Verfahren inkl. Matching!) jeweils berechnet wird. Die Verteilung der so gewonnenen Bootstrap-Schätzwerte approximiert die wahre, unbekannte Stichprobenverteilung des Schätzers des Behandlungseffekts. Mit dieser simulierten Verteilung können wir Inferenz betreiben: Wir können einen Bootstrap-Punktschätzer des Behandlungseffekts (Stichprobenmittel der Bootstrap-Schätzungen) sowie Standardfehler (Standardabweichung der der Bootstrap-Schätzungen) und p-Werte berechnen.

Wir Implementieren nun einen Bootstrap-Schätzer des ATT als R-Funktion `boot_fun()`.

```
boot_fun <- function(data, i) {

  boot_data <- data[i, ]

  # 1:1 PS Matching _mit_ Zurücklegen
  match_res <- matchit(
    dark_mode ~ age + hours + male,
    estimand = "ATT",
    distance = "glm",
    method = "nearest",
    caliper = .3,
    data = boot_data,
    # mit Zurücklegen:
    replace = TRUE
  )

  # Gematchten Datensatz zuweisen
  darkmode_matched <- match.data(match_res, data =
↪   boot_data)

  # Outcome-Modell schätzen
  ATT_mod <- lm(
    formula = read_time ~ age + male + hours + dark_mode,
    data = darkmode_matched,
    weights = weights # hier teilweise > 1 wg. Matching mit
    ↪   Zurücklegen!
  )

  #  ATT-Schätzer auslesen
  return(
    ATT_mod$coefficients["dark_mode"]
  )
}
```

Abadie & Imbens (2008) zeigen analytisch, dass ein Standard-Bootstrap bei Matching grundsätzlich ungültig ist: Die unbekannte Varianz der Stichprobenverteilung des Matching-Schätzers (und damit der Standardfehler des Schät-

zers) kann durch den Bootstrap nicht repliziert werden. Problematisch hierbei sind grundsätzlich zu liberale (d.h. zu große) mit dem Bootstrap berechnete Standardfehler. Es gibt jedoch Simulationsnachweise die zeigen, dass Bootstrap-Standardfehler bei Matching mit Zurücklegen konservativ sind (Bodory et al., 2020), also tendentiell zu kleine Standardfehler produzieren und damit das gewünschte nominale Signifikanzniveau eines Bootstrap-Hypothesentests nicht überschritten wird.

Wir berechnen nun eine Bootstrap-Schätzung des ATT von **dark_mode** auf **readingtime** sowie den zugehörigen Standardfehler und ein 95%-KI mit der zuvor definierten Funktion **boot_fun**.

```
library("boot")
set.seed(4321)
boot_out <- boot(darkmode, boot_fun, R = 999)

boot_out
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = darkmode, statistic = boot_fun, R = 999)


Bootstrap Statistics :
    original    bias    std. error
t1* 1.307298 -0.163572   0.8561828
```

```
# Bootstrap-Schätzer für den Treatment-Effekt
mean(boot_out$t)
```

```
[1] 1.143725
```

```
# = mean(t0) + bias = mean(Bootstrap_samples)
# vgl. 't0 = boot_fun(darkmode, i = 1:1e3)'

# Bootstrap-Standardfehler
```

```r
sd(boot_out$t)
```

```
[1] 0.8561828
```

```r
# 95% Bootstrap-KI für den Treatment-Effekt
boot.ci(boot_out, type = "perc")
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 999 bootstrap replicates

CALL :
boot.ci(boot.out = boot_out, type = "perc")

Intervals :
Level     Percentile
95%   (-0.660,  2.721 )
Calculations and Intervals on Original Scale
```

3.5 Doubly-Robust-Schätzer für ATT/ATE

Implementieren und berechnen Sie einen Doubly-Robust-Schätzer des ATT (vgl. Wooldridge, 2010) für den kausalen Effekt in Aufgabe 5. Vergleichen Sie mit den Ergebnissen der Aufgaben 1 (d), 4 (f) und 5 (d).

```r
# IPW estimation with regression adjustment
ipwra <- function(br, index = 1:nrow(br)) {
    # slice bootstrapped observations
    br <- br %>% slice(index)

    # estimate and predict propensity score
    m <- glm(formula = dark_mode ~ age + hours + male,
             data = br,
             family = binomial(link = 'logit'))

    br <- br %>%
        mutate(ps = predict(m, type = 'response'))
```

```r
# trim control observations outside of treated PS range
# minps <- br %>%
#     filter(dark_mode == 1) %>%
#     pull(ps) %>%
#     min(na.rm = TRUE)
#
# maxps <- br %>%
#     filter(dark_mode == 1) %>%
#     pull(ps) %>%
#     max(na.rm = TRUE)
#
# # do the trimming
# br <- br %>%
#     filter(ps >= minps & ps <= maxps)

br <- br %>%
  filter(
    between(
      x = ps,
      left = .2,
      right = .7
      )
)

# compute IPWs
br <- br %>%
  mutate(
    ipw = case_when(
      dark_mode == 1 ~ 1 / ps,
      dark_mode == 0 ~ 1 / (1 - ps))
  )

# Simple _ATT_ estimate:
# w_means <- br %>%
#     group_by(dark_mode) %>%
#     summarize(m = weighted.mean(read_time, w = ipw))
↪  %>%
#     arrange(dark_mode)
```

```
    #
    # # simple diff-in-means _ATT_ estimate
    #  return(w_means$m[2] - w_means$m[1])

    # Do regression adjustment for _ATE_ estimate
    # TE prediction for whole sample based on TG model
    mtreat <- br %>%
      filter(dark_mode == 1) %>%
      lm(read_time ~ 1 + age + hours + male, data = .,
      ↪  weights = .$ipw) %>%
      predict(newdata = br) %>%
      mean()

    # TE prediction for whole sample based on CG model
    mcont <- br %>%
      filter(dark_mode == 0) %>%
      lm(read_time ~ 1 + age + hours + male, data = .,
      ↪  weights = .$ipw) %>%
      predict(newdata = br) %>%
      mean()

    return(mtreat - mcont) # Regression adjusted _ATE_
     ↪  estimate
}
```

```
b <- boot(data = darkmode, ipwra, R = 999)
# Bootstrap estimate and standard error
mean(b$t)
```

```
[1] 1.945025
```

```
sd(b$t)
```

```
[1] 0.6190766
```

```
# 95% Bootstrap-KI für den Treatment-Effekt
boot.ci(b, type = "perc")
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 999 bootstrap replicates

CALL :
boot.ci(boot.out = b, type = "perc")

Intervals :
Level     Percentile
95%   ( 0.728,  3.085 )
Calculations and Intervals on Original Scale
```

4 Literatur

Abadie, Alberto, and Guido W. Imbens. (2008). *On the Failure of the Bootstrap for Matching Estimators.*Econometrica 76 (**6**): 1537–57.

Bodory, H., Camponovo, L., Huber, M., & Lechner, M. (2020). *The Finite Sample Performance of Inference Methods for Propensity Score Matching and Weighting Estimators.* Journal of Business & Economic Statistics, 38(**1**), 183–200.

Wooldridge, J. M. (2010). *Econometric analysis of cross section and panel data.* MIT press.

5 Regression Discontiniuty Designs

Regression Discontinuity Design (RDD) ist ein Ansatz für die Schätzung von Behandlungseffekten mit Regression, wenn durch einen experimentell oder natürlich gegebenen Umstand die Behandlung an einem Schwellenwert (c) einer *Laufvariable* (X) sprunghaft beeinflusst wird. Ein RDD-Schätzer wird so implementiert, dass lediglich Beobachtungen mit Ausprägungen von X, die knapp ober- oder knapp unterhalb von c liegen, berücksichtigt werden. Die zentrale Idee hierbei ist, dass Individuen nahe bei c im Durchschnitt ähnliche Merkmale aufweisen. Beobachtungen nahe c sind dann insbesondere hinsichtlich potentieller Backdoor-Variablen vergleichbar, sodass deren problematische Pfade geschlossen sind. Das kausale Diagram in Abbildung 5.1 zeigt den grundsätzlichen Zusammenhang.

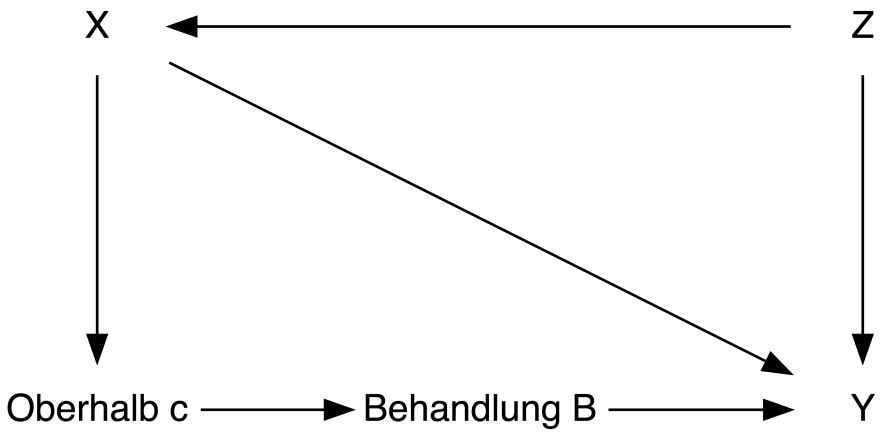


Abbildung 5.1: Kausales Diagramm für Sharp RDD

RDD isoliert Variation auf dem Pfad *Oberhalb* $C \rightarrow$ *Behandlung* $B \rightarrow Y$. Somit können Backdoor-Pfade über X oder weitere (möglichweise unbeobachtbare) Confounder (Z) vermieden werden, siehe Abbildung 5.1. Der kausale Effekt wird dabei als (lokaler) durchschnittlicher Behandlungseffekt der Diskontinuität auf die Outcome-Variable (Y) anhand von Beobachtungen *nahe bei* c ermittelt.

Hinsichtlich der Beeinflussung der Behandlung unterscheiden wir zwischen *Sharp* und *Fuzzy* Regression Discontinuity Designs (SRDD/FRDD). Bei einem SRDD ist die Zuweisung der Behandlung *deterministisch*, d.h. der Schwellenwert in der Laufvariable ist eine harte Grenze für die Gruppenzugehörigkeit: Die *Wahrscheinlichkeit* der Behandlung p springt bei $X = c$ von $p = 0$ um $\Delta p = 1$ auf $p = 1$.

Bei einem FRDD ist die Zuordnung in Behandlungs- und Kontrollgruppe nicht perfekt durch den Schwellenwert c bestimmt: Die Behandlungswahrscheinlichkeit p springt bei $X = c$ um $\Delta p < 1$. Im FRDD können grundsätzlich also sowohl behandelte Subjekte als auch Kontroll-Beobachtungen auf beiden Seiten der Diskontinuität vorliegen – die Trennung der Gruppen ist "unscharf"[1]. Dieser Umstand ist oft in empirischen Studien mit nicht-experimentellen Daten gegeben, wenn es neben der Überschreitung von c weitere Determinanten der Behandlung gibt (für die wir nicht kontrollieren können). Die Wahl zwischen SRDD und FRDD hängt grundsätzlich vom datenerzeugenden Prozess und der Forschungsfrage ab.

5.1 Sharp Regression Discontinuity Design

Modell und funktionale Form

Die korrekte Spezifikation der funktionalen Form für ein RDD ist wichtig, um eine verzerrte Schätzung des Effekts zu vermeiden. Die einfachste Form eines SRDD kann anhand der linearen Regression

$$Y_i = \beta_0 + \beta_1 B_i + \beta_2 X_i + u_i \tag{5.1}$$

geschätzt werden, wobei B_i eine Dummy-Variable für das Überschreiten des Schwellenwertes c ist, d.h.

$$B_i = \begin{cases} 0 & X_i < c \\ 1 & X_i \geq c. \end{cases}$$

Damit ist B_i eine *deterministische* Funktion der Laufvariable X_i und zeigt die Zugehörigkeit zur Behandlungs- oder Treatmentgruppe an. Der Koeffizient β_1 misst den Behandlungseffekt.

Das Modell (5.1) unterstellt, dass X links- und rechtsseitig von c denselben Effekt auf Y hat. Diese Annahme ist restriktiv. Eine Alternative ist ein lineares

[1] Engl. *fuzzy*.

Interaktionsmodell

$$Y_i = \beta_0 + \beta_1 B_i + \beta_2 (X_i - c) + \beta_3 (X_i - c) \times B_i + u_i. \tag{5.2}$$

Das Modell (5.2) kann unterschiedliche lineare Effekte von X auf Y unterhalb (β_2) und oberhalb ($\beta_2 + \beta_3$) von c abbilden. Beachte, dass $(X_i - c)$ die um den Schwellenwert zentrierte Laufvariable ist, sodass β_1 wie in (5.1) den Unterschied des Effekts von X auf Y für Beoabachtungen am Schwellenwert erfasst.

Um unterschiedliche nicht-lineare Zusammenhänge von X und Y unterhalb und oberhalb von c abzubilden, können (interargierte) Polynom-Terme in X verwendet werden. Häufig wird eine quadratische Regressionsfunktion genutzt,

$$\begin{aligned} Y_i = &\beta_0 + \beta_1 B_i + \beta_2 (X_i - c) + \beta_3 (X_i - c)^2 && (5.3) \\ &+ \beta_4 (X_i - c) \times B_i + \beta_5 (X_i - c) \times B_i + u_i. && (5.4) \end{aligned}$$

Gelman und Imbens (2019) zeigen, dass Polynome höherer Ordnung zu verzerrten Schätzern und hoher Varianz führen können.[2] Die Authoren empfehlen stattdessen die Schätzung mit lokaler Regression.

Nicht-parametrische Schätzung und Bandweite

Aktuelle Studien nutzen nicht-parametrische Schätzer, die den Behandlungseffekt als Differenz der geschätzten Regressionsfunktionen am Schwellenwert c berechnen. Um auch nicht-lineare Regressionsfunktionen abzubilden zu können, wird häufig lokale Regression verwendet. Dieses Verfahren liefert eine "lokale" Schätzung der Regressionsfunktionen am Schwellenwert, bei der nur Beobachtungen nahe $X = c$ für die Schätzung berücksichtigt werden. Hinreichende Nähe wird hierbei durch eine sogenannte Bandweite h festgelegt, wobei

$$|(X_i - c)| \leq h \tag{5.5}$$

das Kriterium für eine Berücksichtigung von Beobachtung i bei der Schätzung ist.

Unter Verwendung einer Bandweite h wird der Regressionsansatz (5.2) als *lokale lineare Regression* mit Uniform-Kernelfunktion bezeichnet. Der Uniform-Kernel gibt allen Beobachtungen, innerhalb der Bandweite h dasselbe Gewicht. Ist h so groß, dass der gesamte Datensatz in die Schätzung einbezogen wird, entspricht der lokale lineare Regressions-Schätzer mit Uniform-Kernel dem (globalen) KQ-Schätzer in einem linearen Interaktionsmodell anhand aller Beobachtungen.

[2] Ursachen sind Überanpassung an die Daten sowie instabiles Verhalten der Schätzung nahe des Schwellenwertes.

Neben dem Uniform-Kernel ist der Triangular-Kernel eine in der Praxis häufig genutzte lineare Kernelfunktion. Der nachstehende Code plottet die Uniform- (grün) sowie die Triangular-Kernelfunktion (blau), siehe Abbildung 5.2.

```
library(ggplot2)
library(cowplot)

# Kernelfunktionen zeichnen
ggplot() +
    geom_function(
      fun = ~ ifelse(
        test = abs(.) <= 1,
        yes =  1/2,
        no = 0
      ),
      col = "green",
      n = 1000
      ) +
    geom_function(
      fun = ~ ifelse(
        test = abs(.) <= 1,
        yes = 1 - abs(.),
        no = 0
      ),
      col = "blue",
      n = 100
      ) +
    scale_x_continuous(
      name = "x",
      limits = c(-1.5, 1.5),
      breaks = c(-1, 0, 1)
    ) +
    scale_y_continuous(
      name = "K(x)",
      breaks = c(0, 1),
      limits = c(0, 1.25)
    ) +
    theme_cowplot()
```

In empirischen Studien wird als Basis-Spezifikation oft eine lokale lineare

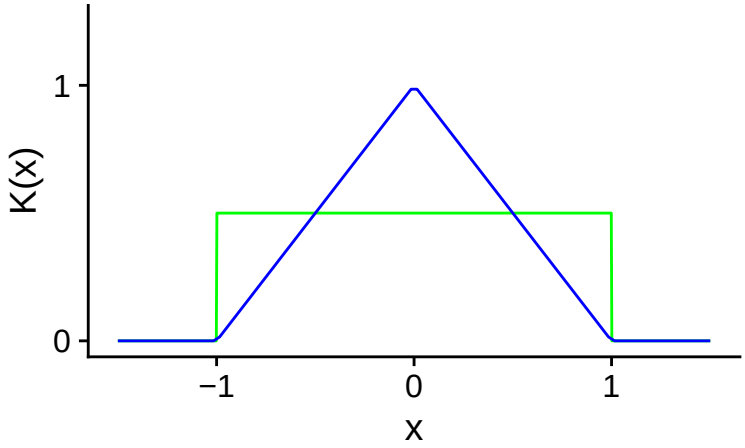


Abbildung 5.2: Kernelfunktionen auf [-1, 1]

Regression anhand von (5.2) mit einer linearen Kernelfunktionen und geringer bandweite h genutzt. Anschließend wird die Robustheit der Ergebnisse anhand flexiblerer Spezifikationen, die Nicht-Linearitäten in der Regressionsfunktion besser abbilden können, geprüft.

Die nachstehende Visualisierung zeigt die Schätzung des kausalen Effektes der Behandlung B_i anhand lokaler linearer Regression mit einem Uniform-Kernel für wiefolgt simulierte Daten:

$$Y_i = \beta_1 X_i + \beta_2 B + \beta_3 X_i^2 \times B_i + u_i,$$

$$u_i \sim N(0, 0.5), \quad X_i \sim U(0, 10), \quad B = \mathbb{I}(X_i \geq c = 5)$$
$$\beta_1 = .5, \quad \beta_2 = 1.5, \quad \beta_3 = -0.15$$

Diese Vorschrift ist schnell mit R umgesetzt:

```
set.seed(1234)
# Anz.Beobachtungen
n <- 750

# Parameter definieren
c <- 5
beta_1 <- .5
beta_2 <- 1.5
beta_3 <- -.15
```

```r
# Regressionsfunktion definieren
f <- function(X) {
  beta_1 * (X - c) + beta_2 * B + beta_3 * B * (X - c)^2
}

# Daten erzeugen
X <- runif(n, 0, 11)
B <- ifelse(X - c >= 0, 1, 0)
Y <- f(X) + rnorm(n, sd = .5)

# Beoabchtungen sammeln
dat <- data.frame(
  Y = Y, X = X - c, B = B
)
```

Diese interaktive Komponente des Buchs ist nur in der Online-Version verfügbar

---

Der interssierende Effekt am Schwellenwert $c = 5$ beträgt $\beta_2 = 1.5$. Beachte, dass aufgrund des Terms $\beta_3 X_i^2 \times B_i$ ein quadratischer Zusammenhang von Y und X oberhalb von $X_i = c$ vorliegt. Es können folgende Eigenschaften der Schätzung in Abhängigkeit von der Bandweite h beobachtet werden:

- Für die voreingestellte Bandweite $h = 1.3$ liefert die lokale lineare Regression eine gute Approximation des Regressionszusammenhangs auf beiden Seiten des Schwellenwertes und die Schätzung des Behandlungseffekts liegt nahe beim wahren Wert $\beta_2 = 1.5$.
- Für kleinere Bandweiten verringert sich die Datenbasis der Schätzung. Die Varianz der Schätzung nimmt zu und die Approximation der Regressionsfunktion verschlechtert sich. Wir beobachten eine mit $h \to 0$ zunehmende Verzerrung bei der Schätzung des Behandlungseffekts.
- Größere Bandweiten h erhöhen die Datenbasis der Schätzung, führen aber zu einer Annäherung der lokalen Schätzung an die globale KQ-Schätzung.

Linksseitig des Schwellenwertes erzielen wir damit eine Schätzung mit hoher Güte. Rechsseitig von $X_i = c$ verschlechtert sich die lokale Anpassung am Schwellenwert deutlich, weil die lineare Schätzung den tatsächlichen (nicht-linearen) Zusammenhang nicht adäquat abbilden kann. Die Schätzung des Behandlungseffekts ist hier deutlich verzerrt.

Die Wahl der Bandweite ist also eine wichtige Komponenten der RDD-Schätzung: Kleine Bandweiten erlauben eine Schätzung der Regressionsfunktion nahe des Schwellenwertes mit wenig Verzerrung. Allerdings kann diese Schätzung unpräzise sein, wenn nur wenige Beobachtungen (5.5) erfüllen. In der Praxis wird h daher mit einem analytischen Schätzer (vgl. G. Imbens und Kalyanaraman 2012) oder anhand von *Cross Validation* (bspw. G. W. Imbens und Lemieux 2008) bestimmt. Die später in diesem Kapitel betrachteten R-Pakete halten diese Methoden bereit.

5.2 Manipulation am Schwellenwert

Eine wichtige Annahmen für die Gültigkeit einer RDD-Schätzung ist, dass keine Manipulation der Gruppenzugehörigkeit am Schwellenwert vorliegt. Wenn sich Subjekte nahe des Schwellenwertes c — d.h. in Abhängigkeit der Laufvariable X — systematisch in den Confoundern Z unterscheiden, können wir den Backdoor-Pfad *Oberhalb* $C \rightarrow$ *Behandlung* $B \rightarrow Y$ nicht isolieren. Wir erhalten dann eine verzerrte Schätzung des Behandlungseffekts.

In empirischen Studien mit Individuen kann Selbstselektion auftreten: Menschen mit $X < c$ aber nahe c (hier Kontrollgruppe) könnten aufgrund unbeobachtbarer Eigenschaften Z die Ausprägung ihrer Laufvariable zu $X > c$ (hier Behandlungsgruppe) manipulieren. Wenn Z die Outcome-Variable beeinflusst, bleibt der Backdoor-Pfad *Oberhalb* $C \rightarrow$ *Behandlung* $B \rightarrow Y$ so bestehen.

Manipulation resultiert in Häufung von Beobachtungen am Schwellenwert. Dei Verteilung der Laufvariable kann auf diese Unregelmäßigkeit hin untersucht werden. McCrary (2008) schlägt hierfür einen Verfahren vor, das die Kontinuität der Dichtefunktion von X am Schwellenwert testet.

Der Test von McCrary (2008) ist in `rdd::DCdensity()` implementiert. Wir zeigen die Anwendung des Tests anhand der oben simulierten Daten. Beachte, dass $X_i \sim U(0, 10)$, d.h. die Laufvariable ist bei $X_i = c$ kontinuierlich verteilt. Die Nullhypothese (keine Manipulation) gilt für die simulierten Daten

```r
# McCrary-Test durchführen
p_mccrary <- rdd::DCdensity(
  runvar = X,
  cutpoint = c,
  plot = F
)

# p-Wert
p_mccrary
```

```
[1] 0.5013939
```

Der p-Wert 0.5 ist größer als jedes übliche Signifikanzniveau. Damit liegt starke Evidenz für die Nullhypothese (keine Diskontinuität) und gegen Manipulation am Schwellenwert vor.

Cattaneo, Jansson, und Ma (2020) (CMJ) schlagen eine Weiterentwicklung des McCrary-Tests vor, die höhere statistische Macht gegenüber Diskontinuitäten hat am Schwellenwert hat. Der CJM-Test ist im Paket `rddensity` implementiert.

```r
library(rddensity)

# CJM Schätzer berechnen
CJM <- rddensity(X, c = 5)
```

Mit der Funktion `rddensity::rdplotdensity()` erzeugen wir Abbildung 5.3.

```r
# Plot für Dichtefunktion erstellen
plot <- rdplotdensity(
  rdd = CJM,
  X = X,
  # für Punkte- und Linienplots:
  type = "both"
)
```

Abbildung 5.3 zeigt die geschätzten Dichtefunktionen. Erwartungsgemäß finden wir eine große Überlappung der zugehörigen Konfidenzbänder (schattierte Flächen) am Schwellenwert $c = 5$.

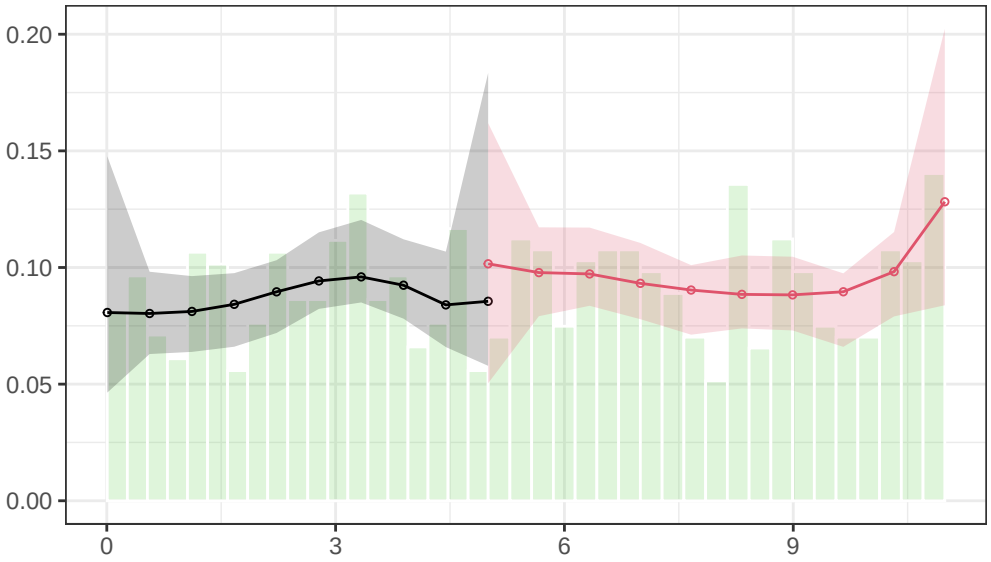


Abbildung 5.3: CJM-Test – geschätzte Dichtefunktionen der Laufvariable auf beiden Seiten des Schwellenwerts c = 5

Mit `summary()` erhalten wir eine detaillierte Zusammenfassung des Tests.

```
# Statistische Zusammenfassung des CJM-Tests
summary(CJM)
```

```
Manipulation testing using local polynomial density estimation.

Number of obs =       750
Model =               unrestricted
Kernel =              triangular
BW method =           estimated
VCE method =          jackknife

c = 5                 Left of c           Right of c
Number of obs         329                 421
Eff. Number of obs    133                 154
Order est. (p)        2                   2
Order bias (q)        3                   3
BW est. (h)           1.918               2.124

Method                T                   P > |T|
Robust                -0.3338             0.7385
```

```
P-values of binomial tests (H0: p=0.5).

Window Length              <c     >=c    P>|T|
0.346     + 0.346          20      21    1.0000
0.521     + 0.544          34      37    0.8126
0.696     + 0.742          44      57    0.2323
0.870     + 0.939          54      64    0.4075
1.045     + 1.137          62      77    0.2349
1.220     + 1.334          73      98    0.0661
1.394     + 1.532          86     106    0.1701
1.569     + 1.729          96     124    0.0685
1.743     + 1.927         119     140    0.2139
1.918     + 2.124         133     154    0.2377
```

Gemäß des p-Werts (`P > |T|`) von 0.74 spricht der CJM-Test noch deutlicher gegen eine Diskontinuität als der McCrary-Test.

5.2.1 Case Study: Amtsinhaber-Vorteil (Lee 2008)

Lee (2008) untersucht den Einfluss des Amtsinhaber-Vorteils auf die Wahl von Mitgliedern des US-Repräsentantenhaus. In den meisten Wahlkreisen entfallen große Anteile der Stimmen (oder gar ausschließlich) auf demokratische und republikanische Kanditat*innen, sodass sich die Studie auf diese Parteien beschränkt. Entfällt die Mehrheit der Stimmen auf eine*n Kandiat*in, gewinnt diese*r den Sitz für den Wahlkreis. Durch die Analyse der 6558 Wahlen im Zeitraum 1946-1998 mit einem SRDD kommt die Studie zu dem Ergebnis, dass Amtsinhabende im Durchschnitt einen Vorteil von etwa 8% bis 10% bei der Wahl haben. Dieses Ergebnis kann verschiedene Ursachen haben, bspw. dass die amtierende Partei höhere finanzielle Ressourcen besitzt und von einer besseren Organisation und durch Instrumenalisierung staatlicher Strukturen für die eigenen Zwecke profitiert.

Anhand der Datensätze `house` und `house_binned` illustrieren wir nachfolgend die Schätzung von SRDD-Modellen für den Wahlerfolg der demokratischen Partei, wenn diese Amtsinhaber ist. Wir lesen hierfür zunächst die Datensätze `house` und `house_binned` ein und verschaffen uns einen Überblick.

```r
library(tidyverse)
library(modelsummary)

# Daten einlesen
house <- read_csv("datasets/house.csv")
# Gruppierter Datensatz
house_binned <- read_csv("datasets/house_binned.csv")

# Überblick verschaffen
glimpse(house)
```

```
Rows: 6,558
Columns: 2
$ StimmenTm1 <dbl> 0.1049, 0.1393, -0.0736, 0.0868, 0.3994,
0.1681, 0.2516, 0.~
$ StimmenT   <dbl> 0.5810, 0.4611, 0.5434, 0.5846, 0.5803,
0.6244, 0.4873, 0.5~
```

```r
glimpse(house_binned)
```

```
Rows: 100
Columns: 2
$ StimmenT   <dbl> 0.5995600, 0.5657000, 0.4272554, 0.5637456,
0.6868627, 0.60~
$ StimmenTm1 <dbl> 0.104764444, 0.135005263, -0.075690769,
0.084570886, 0.3951~
```

Der Datensatz `house` enthält die Stimmenanteile demokratischer Kandidat*innen bei der Wahl zum Zeitpunkt T ($StimmenT$) sowie die Differenz zwischen demokratischen und republikanischen Stimmenanteilen bei der vorherigen Wahl, d.h. zum Zeitpunkt $T-1$ ($StimmenTm1$). Der Schwellenwert für einen Wahlsieg liegt bei Stimmengleichheit, d.h. $StimmenTm1 = 0$.

`house_binned` ist eine aggregierte Version von `house` mit Mittelwerten von jeweils 50 gleichgroßen Intervallen oberhalb und unterhalb der Schwelle von $StimmenTm1 = 0$. Dieser Datensatz eignet sich, um einen ersten Eindruck des funktionalen Zusammenhangs auf beiden Seiten zu erhalten. Wir stellen zunächst diese klassierten Daten mit `ggplot2` graphisch dar.

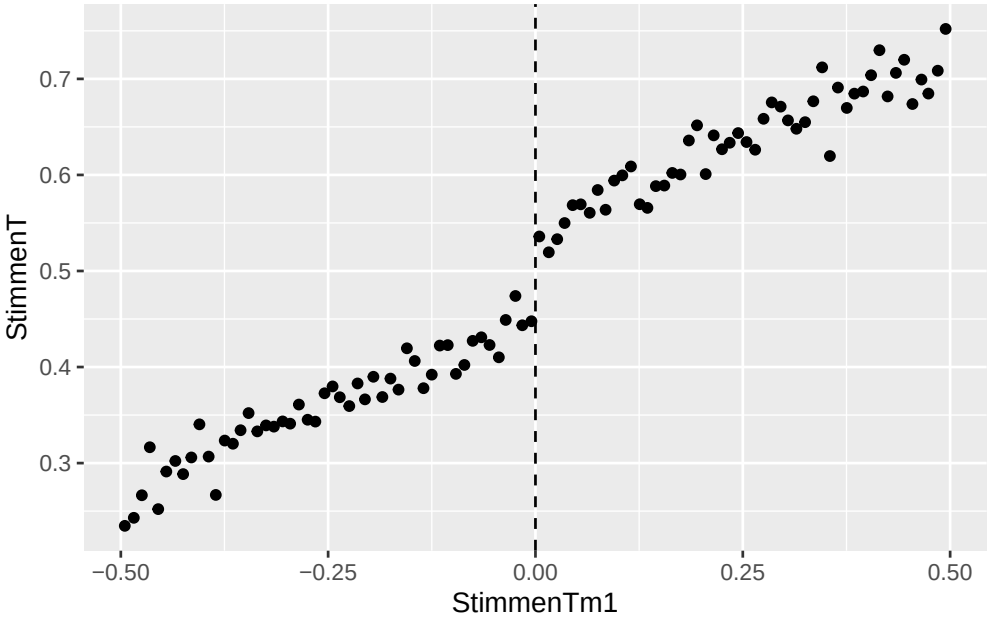


Abbildung 5.4: Klassierte Daten aus Lee (2008)

```
# Klassierte Daten plotten
house_binned %>%
  ggplot(
    aes(x = StimmenTm1, y = StimmenT)
    ) +
  geom_point() +
  geom_vline(xintercept = 0, lty = 2)
```

Die Grafik zeigt eindeutig einen Sprung von $StimmenT$ bei $StimmenTm1 = 0$. Weiterhin erkennen wir, dass der Zusammenhang nahe 0 vermutlich jeweils gut durch eine lineare Funktion approximiert werden kann. Eine Modell-Spezifikation mit gleicher Steigung auf beiden Seiten des Schwellenwertes scheint hingegen weniger gut geeignet. Wir vergleichen diese Spezifikationen nachfolgend.

Zunächst fügen wir dem Datensatz eine Dummyvariable `B` hinzu. Diese dient als Indikator für den Wahlgewinn in der letzten Wahl und zeigt die Amtsinhaberschaft (Behandlung) an.

```
# Behandlungsindikator B hinzufügen
house <- house %>%
  mutate(B = StimmenTm1 > 0)
```

```
glimpse(house)
```

```
Rows: 6,558
Columns: 3
$ StimmenTm1 <dbl> 0.1049, 0.1393, -0.0736, 0.0868, 0.3994,
0.1681, 0.2516, 0.~
$ StimmenT   <dbl> 0.5810, 0.4611, 0.5434, 0.5846, 0.5803,
0.6244, 0.4873, 0.5~
$ B          <lgl> TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, TRUE,
TRUE, TRUE, TRUE~
```

Wir überprüfen die Laufvariable mit dem CJM-Test auf Manipulation am Schwellenwert $c = 0$.

```
# CJM-Test durchführen
CJM_Lee <- rddensity(X = house$StimmenTm1)

# Zusammenfassung anzeigen
summary(CJM_Lee)
```

```
Manipulation testing using local polynomial density estimation.

Number of obs =       6558
Model =               unrestricted
Kernel =              triangular
BW method =           estimated
VCE method =          jackknife

c = 0                 Left of c           Right of c
Number of obs         2740                3818
Eff. Number of obs    1297                1360
Order est. (p)        2                   2
Order bias (q)        3                   3
BW est. (h)           0.236               0.243

Method                T                   P > |T|
Robust                1.4346              0.1514
```

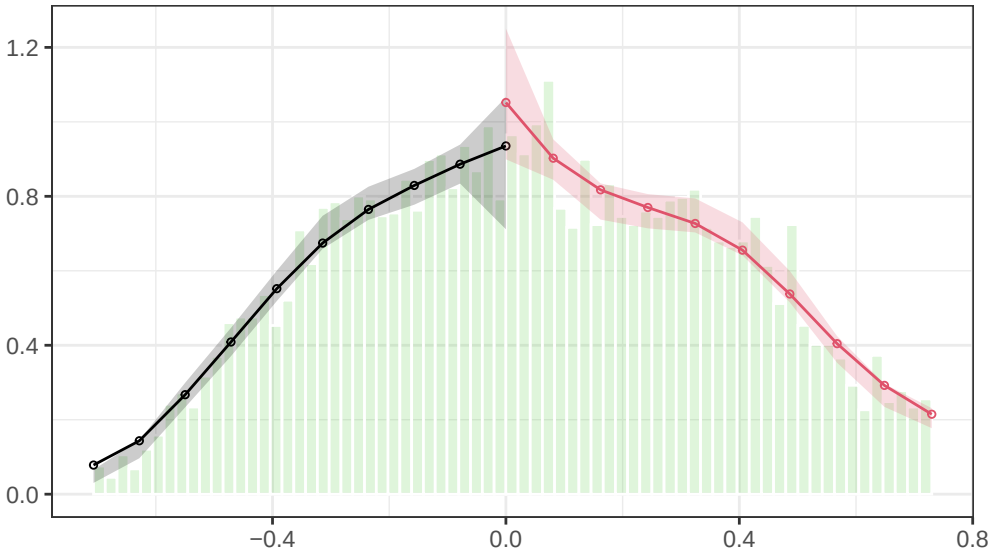


Abbildung 5.5: CJM-Test – geschätzte Dichtefunktionen der Laufvariable

```
P-values of binomial tests (H0: p=0.5).

Window Length / 2          <c     >=c    P>|T|
0.004                      21      24    0.7660
0.007                      38      46    0.4452
0.011                      50      60    0.3909
0.014                      73      77    0.8066
0.018                      91     104    0.3902
0.022                     124     132    0.6618
0.025                     149     149    1.0000
0.029                     163     174    0.5860
0.032                     176     202    0.1984
0.036                     197     223    0.2225
```

```r
# CJM-Plot
plot <- rdplotdensity(
  rdd = CJM_Lee,
  X = house$StimmenTm1,
  type = "both",
)
```

Abbildung 5.5 und der p-Wert von 0.15 sind Evidenz gegen eine Manipulation am Schwellenwert.

Um den Behandlungseffekt anhand eines SRDDs zu ermitteln, schätzen wir das Interaktionsmodell

$$\begin{aligned}\text{StimmenT}_i = \beta_0 + \beta_1 B_i + \beta_2(\text{StimmenTm1}_i - 50) \\ + \beta_3(\text{StimmenTm1}_i - 50) \times B_i + u_i\end{aligned}$$

zunächst für eine Bandweite von $h = 0.5$. Aufgrund der Skalierung der Daten (Wahlergebnisse in %) bedeutet dies die Verwendung des *gesamten* Datensatzes für die Schätzung.

```
# Interaktionsmodell schätzen
house_llr1 <- lm(
  formula = StimmenT ~ B * StimmenTm1,
  data = house
)

# Zusammenfassung anzeigen
modelsummary(
  models = house_llr1,
  vcov = "HC1", # robuste Standardfehler
  stars = T,
  gof_map = "nobs",
  output = "gt"
) %>%
  tabopts
```

	(1)
(Intercept)	0.433***
	(0.004)
BTRUE	0.118***
	(0.006)
StimmenTm1	0.297***
	(0.016)
BTRUE × StimmenTm1	0.046*
	(0.018)
Num.Obs.	6558

+ p < 0.1, * p < 0.05, ** p < 0.01, *** p < 0.001

Der geschätzte Koeffizient von B (`BTRUE`) beträgt etwa 0.12 und ist hochsi-

gnifikant. Übereinstimmend mit Abbildung 5.4 erhalten wir also eine positive Schätzung des Behandlungseffekts. Die Interpretation ist, dass die amtierenden Demokraten bei der Wahl von einem Amtsinhabervorteil profitieren. Dieser Effekt schlägt sich als Stimmenbonus von geschätzten 12% nieder. Diese Schätzung des Behandlungseffekts könnte jedoch verzerrt sein:

- Die (implizite) Wahl von $h = 0.5$ in unserer Schätzung macht die Isolation des relevanten Frontdoor-Paths ($c = 0 \rightarrow$ Treatment $\rightarrow$ StimmenT) wenig plausibel. h sollte mit einer datengetriebenen Methode gewählt werden.
- Weiterhin könnte die lineare funktionale Form der Regression inadäquat sein: Die lineare Approximation der wahren Regressionsfunktion nahe des Schwellenwerts 0 könnte unzureichend sein und in einer verzerrten Schätzung des Effekts resultieren. Zur Überprüfung der Robustheit der Ergebnisse sollte mit Schätzungen anhand nicht-linearer Spezifikationen verglichen werden.

Um diesen Gefahren für die Validität der Studie zu begegnen, schätzen wir nun weitere Spezifikationen. Im Folgenden verwenden wir eine Bandweitenschätzung gemäß G. Imbens und Kalyanaraman (2012).

```
# Bandweite mit Schätzer von IK (2012) berechnen
(
IK_BW <-
  rdd::IKbandwidth(
    X = house$StimmenTm1,
    Y = house$StimmenT
  )
)
```

```
[1] 0.2685123
```

Wir schätzen zunächst erneut das lineare Interaktionsmodell, diesmal jedoch mit der Bandweite `IK_BW`.

```
# Lineares Interaktionsmodelle mit IK-Bandweite
house_llin_IK <- lm(
  formula = StimmenT ~ B * StimmenTm1,
  data = house %>%
    filter(
      abs(StimmenTm1) <= IK_BW
```

```
    )
)
```

Für den Vergleich mit einer nicht-linearen Spezifikation schätzen wir auch ein quadratisches Interaktionsmodell.

```
# Quadratisches Interaktionsmodell mit IK-Bandweite
house_poly_IK <- update(
  object = house_llin_IK,
  formula = StimmenT ~ B * poly(StimmenTm1, degree = 2, raw
  ↪  = T)
)
```

Für eine Gegenüberstellung der Ergebnisse verwenden wir `modelsummary()`.

```
# Tabellarischer Modellvergleich
modelsummary(
  models = list(
    "Linear int." = house_llin_IK,
    "Quadratisch int." = house_poly_IK
  ),
  vcov = "HC1",
  stars = T,
  gof_map = "nobs",
  output = "gt"
) %>%
  tabopts
```

Tabelle 5.1: Vergleich von SRDD-Interaktionsmodellen für Lee (2008)

	Linear int.	Quadratisch int.
(Intercept)	0.450***	0.460***
	(0.005)	(0.008)
BTRUE	0.085***	0.068***
	(0.008)	(0.012)
StimmenTm1	0.360***	
	(0.036)	
BTRUE × StimmenTm1	0.055	
	(0.059)	

poly(StimmenTm1, degree = 2, raw = T)1		0.573***
		(0.138)
poly(StimmenTm1, degree = 2, raw = T)2		0.798
		(0.493)
BTRUE × poly(StimmenTm1, degree = 2, raw = T)1		0.036
		(0.219)
BTRUE × poly(StimmenTm1, degree = 2, raw = T)2		-1.529+
		(0.834)
Num.Obs.	2956	2956

+ p < 0.1, * p < 0.05, ** p < 0.01, *** p < 0.001

Die Spalte (1) in Tabelle 5.1 zeigt die lokale Schätzung mit einem linearen Interaktionsmodell. Wir erhalten damit einen Behandlungseffekt von etwa 8.5%. Der Schätzwert fällt also etwas geringer aus als für die globale KQ-Schätzung des linearen Interaktionsmodells. Für das Modell (2) mit quadratischer Spezifikation liegt der Schätzwert mit 6.8% in der selben Größenordnung. Beide Schätzungen ergeben einen signifikant von 0 verschieden Effekt. Weiterhin fällt auf, dass in beiden Modellen keine Evidenz für unterschiedliche Formen der Regressionsfunktionen auf beiden Seiten des Schwellenwerts vorliegen: sämtliche Koeffizientenschätzwerte der Interaktionsterme haben hohe Standardfehler und sind nicht signifikant. Im quadratischen Modell hat auch der Term $StimmenTm1^2$ keinen signifikanten Effekt. Diese Ergebnisse deuten darauf hin, dass eine lineare Spezifikation ausreichend ist.

SRDD-Schätzung mit LOESS

Wir illustrieren nachfolgend die Schätzung des Behandlungseffekts mit einer flexiblen und in der Praxis häufig verwendeten Methode für lokale Regression. Die nachfolgende interaktive Grafik zeigt die klassierten Daten aus Lee (2008) auf dem Intervall $[-0.5, 0.5]$ gemeinsam mit einer nicht-parametrischen Schätzung des Zusammenhangs von `StimmenT` und `StimmenTm1` mittels LOESS.[3] Diese Implementierung von lokaler Regression nutzt einen tricube kernel. Über den Input kann eine Bandweite $l \in (0, 1]$ für den LOESS-Schätzer auf beiden Seiten des Schwellenwerts 0 gewählt werden. Die Bandweite ist hier der *Anteil der Beobachtungen an der gesamten Anzahl an Beobachtungen*, die in die Schätzung einbezogen werden sollen.

Für die Schätzung am Schwellenwert berücksichtigte Daten sind in orange kenntlich gemacht. Die rote linie zeigt die geschätzte Regressionsfunktion über gleichmäßig verteilte Werte von `StimmenTm1` auf $[-0.5, 0.5]$. Die Grafik

[3] LOESS ist eine Variante von lokaler Polynom-Regression.

verdeutlicht, dass die LOESS-Methode flexibel genug ist, um lineare und nicht-lineare Zusammenhänge abbilden zu können. Wie zuvor ist eine adäquate Wahl der Bandweite wichtig:

- Der mit LOESS geschätzte Zusammenhang auf beiden Seiten des Schwellenwerts ist etwa linear für den voreingestellten Parameter ($l = 0.28$).
- Für größere Werte von l nähert sich die Schätzung weiter einem linearen Verlauf an. Die Schätzung des Effekts bleibt vergleichbar mit den Ergebnissen des linearen Interaktionsmodell (s. oben).
- Für kleinere l erhalten wir eine stärkere Anpassung der Schätzung an die Daten. Zu kleine Werte führen zu einer Überanpassung (*overfitting*). Insbesondere tendiert die geschätzte Funktion zu extremer Steigung nahe des Schwellenwerts $\rightarrow$ stark verzerrte Schätzung des Effekts!

Diese interaktive Komponente des Buchs ist nur in der Online-Version verfügbar.

5.3 Fuzzy Regression Discontinuity Design

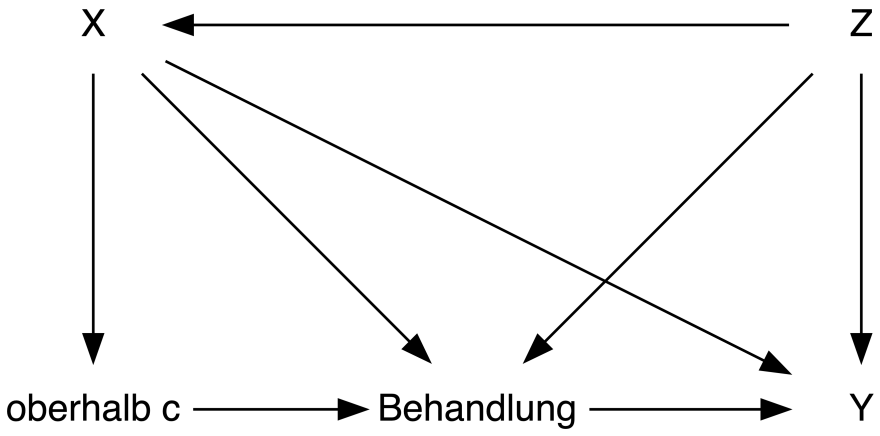


Abbildung 5.6: Kausales Diagram für FRDD

Ein FRDD liegt vor, wenn die Zuweisung der Behandlung B durch die Laufvariable X (und möglicherweise weitere Variablen Z) beeinflusst wird. Im Vergleich zum SRDD ist die Behandlung dann also *nicht* ausschließlich durch Überschreiten des Schwellenwerts $X = c$ bestimmt.

Abbildung 5.6 zeigt den grundsätzlichen Zusammenhang. Hier genügt es weiterhin für X (und ggf. Z) zu kontrollieren, um den Pfad *oberhalb* $C \rightarrow$ *Behandlung* $B \rightarrow Y$ zu isolieren. Der so für *Behandlung* B ermittelte Effekt auf Y entspricht jedoch *nicht* dem "vollständigen" Behandlungseffekt, da bei c die Zuweisung der Behandlung nicht von 0 auf 100% springt. Die Schätzung des FRDD berücksichtigt dies und skaliert den geschätzten Effekt entsprechend.

Wir betrachten zunächst den Zusammenhang

$$Y_i = \beta_0 + \beta_1 B_i + \beta_2 (X_i - c) + u_i. \tag{5.6}$$

In einem FRDD springt die Behandlungswahrscheinlichkeit am Schwellenwert c um $\Delta p < 1$. Wir können B also nicht als deterministische Funktion von X, welche die Zuweisung zu Behandlungs- bzw. Kontrollgruppe am Schwellenwert c anzeigt (wie im SRDD), definieren. Stattdessen betrachten wir

$$P(B_i = 1 | X_i) = \begin{cases} g_{X_i<c}(X_i), & X_i < c \\ g_{X_i \geq c}(X_i) & X_i \geq c \end{cases}. \tag{5.7}$$

Die Funktionen $g_{X_i<c}$ und $g_{X_i \geq c}$ können verschieden sein. Es muss jedoch

$$g_{X_i<c}(X_i = c) \neq g_{X_i \geq c}(X_i = c)$$

gelten. Die Behandlungsvariable B_i ist im FRDD also eine (binäre) Zufallsvariable, deren bedingte Wahrscheinlichkeitsfunktion $P(B_i = 1|X_i)$ am Schwellenwert c eine Diskontinuität aufweist. Abbildung 5.7 zeigt heispielhafte Verläufe nicht-linearer bedingter Wahrscheinlichkeitsfunktion für die Behandlung mit einer Diskontinuität bei $X_i = c$.

```
library(ggplot2)
library(cowplot)

# Bedingte Behandlungswahrscheinlichkeit im FRDD
↪  illustrieren
ggplot() +
  geom_function(
```

```
    fun = ~ ifelse(
      . < 0,
      -.1 * .^2 + .25,
      -.1 * (.-1.5)^2 + 1
    ),
    n = 1000
  ) +
    geom_function(
    fun = ~ ifelse(
      . < 0,
     .35,
     .65
    ),
    n = 1000,
    lty = 2,
    col = "red"
  ) +
  scale_x_continuous(
    name = "Laufvariable X",
    limits = c(-1.5, 1.5),
    labels = NULL,
    breaks = NULL
  ) +
  scale_y_continuous(
    name = "P(D=1|X)",
    breaks = c(0, 1),
    limits = c(0, 1)
  ) +
  theme_cowplot()
```

Definition (5.7) bedeutet, dass eine KQ-Schätzung von β_1 anhand (5.6) eine *verzerrte* Schätzung des Behandlungseffekts ist: Der in $\widehat{\beta}_1$ erfasste Effekt auf Y ist auf einen Sprung der Behandlungswahrscheinlichkeit bei $X_i = c$ um *weniger* als 100% zurückzuführen. Der wahre Behandlungseffekt wird also *unterschätzt*. Daher muss $\widehat{\beta}_1$ skaliert werden, sodass die Schätzung als Effekt einer Änderung der Behandlungswahrscheinlichkei um 100% interpretiert werden kann — der erwartete Effekt, wenn ausschließlich Subjekte mit $X_i \geq c$ behandelt würden. Diese skalierte Schätzung erhalten wir mit IV-Regression (vgl. Kapitel XYZ).

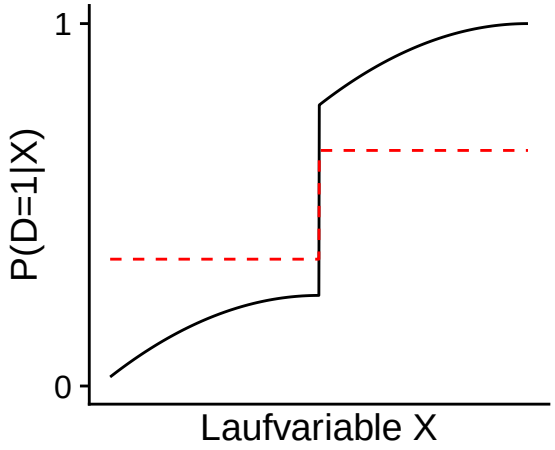


Abbildung 5.7: Bedingte Behandlungswahrscheinlichkeiten im FRDD

Hierfür nutzen wir für B_i die Instrumentvariable

$$D_i = \begin{cases} 0, & X_i < c \\ 1, & X_i \geq c. \end{cases}$$

Angenommen $g_{X_i \geq c}(X_i) = \alpha_0$ und $g_{X_i < c}(X_i) = \alpha_0 + \alpha_1$ mit $\alpha_0 + \alpha_1 < 1$ (vgl. rote Funktion in Abbildung 5.6). Der FRDD-Schätzer des Behandlungseffekts ist dann $\widehat{\gamma}_{\text{FRDD}}$ im 2SLS-Verfahren mit den Regressionen

$$\begin{aligned} &\text{(I)} \qquad B_i = \alpha_0 + \alpha_1 D_i + \alpha_2 (X_i - c) + e_i, \\ &\text{(II)} \qquad Y_i = \gamma_0 + \gamma_1 \widehat{B}_i + \gamma_2 (X_i - c) + \epsilon_i, \end{aligned} \tag{5.8}$$

wobei $\widehat{B}_i$ die angepassten Werte aus Stufe (I) und e_i sowie ϵ_i Fehlterterme sind.

Analog zum SRDD müssen in empirischen Anwendungen geeignete Spezifikationen für die Regressionsfunktionen (5.6) und (5.7) gewählt und der 2SLS-Schätzer (5.8) entsprechend angepasst werden. Ein einfaches Interaktionsmodell wäre

$$\begin{aligned} \text{(I)} \qquad B_i &= \alpha_0 + \alpha_1 D_i + \alpha_2 (X_i - c) \\ &\quad + \alpha_3 (X_i - c) \times D_i + e_i, \\ \text{(II)} \qquad Y_i &= \gamma_0 + \gamma_1 \widehat{B}_i \\ &\quad + \gamma_2 (X_i - c) + \gamma_3 (X_i - c) \times \widehat{B}_i, \epsilon_i \end{aligned} \quad , \tag{5.9}$$

d.h. wir instrumentieren B_i mit D_i und dem Interaktionsterm $(X_i - c) \times D_i$.

Wie im SRDD werden die IV-Ansätze für das FRDD (5.8) und (5.9) in empirischen Studien unter Berücksichtigung einer Bandweite (i.d.R. dieselbe Bandweite für beide Stufen) angewendet.

5.4 Case Study: Protestantische Arbeitsethik

Die Studie *Beyond Work Ethic: Religion, Individual, and Political Preferences* (Basten und Betz 2013) untersucht den Zusammenhang zwischen Religion, individuellen Merkmalen und politischen Präferenzen. Das Hauptaugenmerk ist die Rolle von Religiosität als Einflussfaktor auf politische Einstellungen. Die Hypothese der Autoren ist, dass Religiosität eines Individuums über den traditionellen Rahmen von Moralvorstellungen und sozialen Normen hinaus auch die politischen Präferenzen beeinflusst. Eine entsprechende Theorie wurde zu Beginn des 20. Jahrhunderts entwickelt und prominent von Max Weber (vgl. Weber 2004) vertreten. Weber argumentiert, dass die protestantische Arbeitsethik einen entscheidenden Einfluss auf die Entwicklung des Kapitalismus hatte. Laut Weber führte der protestantische Glaube an harte Arbeit, ein sparsames Leben und ethisches Verhalten zur einer in den damaligen Gesellschaften weit verbreiteten Geisteshaltung, die wirtschaftliches Wachstum förderte und den Aufstieg des Kapitalismus begünstigte.

Basten und Betz (2013) nutzen Wahlergebnisse sowie geo- und soziodemographische Datensätze für schweizer Gemeinden, um den Zusammenhang zwischen Religiosität und politischen Präferenzen wie links-rechts-Ausrichtung, Einstellungen zur Umverteilung und Einwanderung zu untersuchen. Hierfür verwenden die Autoren ein FRDD, dass eine historisch bedingte Diskontinuität der geographischen Verteilung von evanglischer bzw. katholischer Religionszugehörigkeit zwischen den Kantonen Freiburg (überwiegend dunkelrote Region, frz. *Fribourg*) und Waadt (kleinere hellrote Region, frz. *Vaud*) ausnutzt. Die historische Verteilung der Konfessionen in der betrachteten Region im 16. Jahrhundert durch Abspaltung des Kantons Freiburg ist in Abbildung 5.8 dargestellt.

Aufgrund von Bevölkerungsbewegungen ist die Verteilung der Konfessionen zwar nicht mehr eindeutig durch die Kantonsgrenze bestimmt, jedoch sind die Gemeinden der betrachteten Kantone auch heute noch mehrheitlich protestantisch bzw. katholisch. Es ist plausibel, dass eine Prägung gemäß Webers Theorie vorliegt, sich die Gemeinden nahe der Grenz aber hinsichtlich anderer Charakteristika (insb. der Bevölkerungsstruktur) nicht systematisch unterscheiden. Somit liegt ein quai

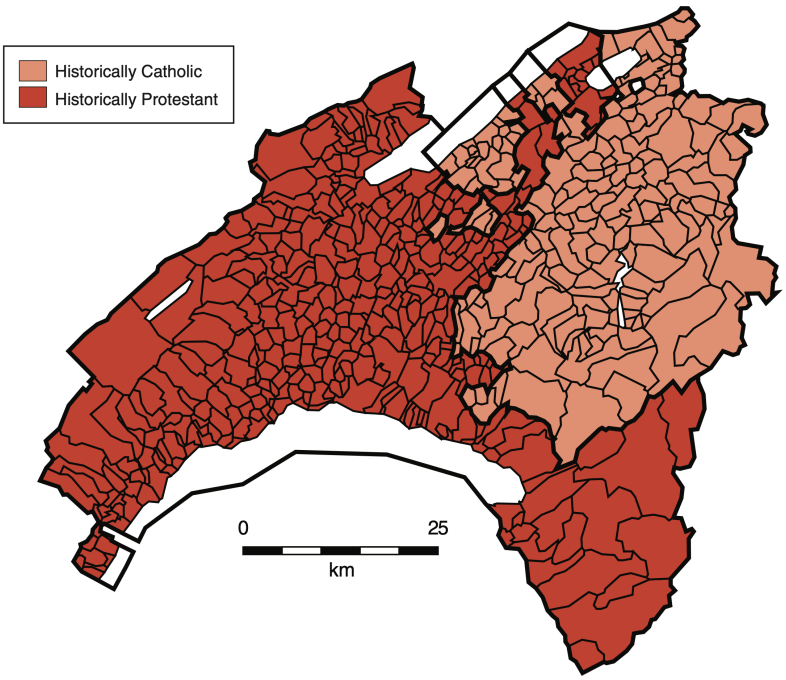


Abbildung 5.8: Historische Verteilung von Religionszugehörigkeit in Schweizer Gemeinden im 16. Jahrhundert. Quelle: Basten und Betz (2013).

Die Ergebnisse der Studie zeigen einen signifikanten Einfluss von Protestantismus auf politische Präferenzen, die über traditionelle Moralvorstellungen hinausgehen: Die Autoren finden Hinweise, dass Einwohner evangelisch geprägter Gemeinden eher konservative soziale und politische Ansichten vertreten. Eine mögliche Erklärung für diesen Effekt ist, dass religiöse Institutionen auch eine soziale und politische Agenda verfolgen, die von den Gläubigen internalisiert wird.

5.4.1 Aufbereitung der Daten

In diesem Kapitel zeigen wir, wie die Kernergebnisse der Studie mit R reproduziert werden können. Hierfür werden folgende Pakete benötigt.

```
library(tidyverse)
library(haven)
library(vtable)
library(rdrobust)
```

Das Papier sowie der Datensatz `BastenBetz.dta` sind auf der Übersichtsseite der AEA verfügbar und liegt im STATA-Format `.dta` vor.[4]

[4]Siehe alternativ das working paper, falls kein Abbonement für AEA-Journals vorliegt.

```
# Datensatz einlesen
BastenBetz <- read_dta('BastenBetz.dta')
```

Der Datensatz `BastenBetz` enthält Beobachtungen zu 509 schweizer Gemeinden. Eine Vielzahl an Variablen ist lediglich für Robustheits-Checks relevant. Für die Reproduktion der Kernergebnisse erstellen wir zunächst einen reduzierten Datensatz und transformieren einige Variablen.

```
# Reduzierten Datensatz erstellen
BastenBetz <- BastenBetz %>%
  transmute(
    gini = Ecoplan_gini,
    prot = prot1980s,
    bord = borderdis,
    vaud,
    pfl,
    pfr,
    pfi
  )
```

Die Definitionen der Variablen sind in Tabelle 5.2 gegeben. Die Präferenzen `pfl`, `pfr` und `pfi` basieren auf Wahlergebnissen auf Gemeindeebene zu Volksentscheiden.

Tabelle 5.2: `BastenBetz` – Variablen und Definitionen

Variable	Definition
`prot`	Anteil Prothestanten im Jahr 1980 (%)
`gini`	Gini-Koeffizient
`bord`	Laufdistanz zur Kantonsgrenze (Km)
`vaud`	Dummyvariable: Gemeine im Kanton Waadt
`pfl`	Präferenz für Freizeit (%)
`pfr`	Präferenz für Umverteilung (%)
`pfi`	Präferenz für wirtschaftliche Intervention des Staats (%)

Für die Berechnung der optimalen Bandweite des FRDD verwenden wir einen MSE-optimalen Schätzer, der in der Funktion `rdrobust::rdbwselect()` implementiert ist.[5]

[5] Basten und Betz (2013) setzen BW = 5.01, den Durchschnitt von IK-Schätzungen über Modelle sämtlicher betrachteter Outcome-Variablen. Diese Bandweite liegt nahe des

```r
# Bandweite schätzen (Bsp. für Freizeitpräferenz)
bw_selection <- rdbwselect(
  y = BastenBetz$pfl,
  x = BastenBetz$bord,
  fuzzy = BastenBetz$prot,
  bwselect = "mserd",
  kernel = "uniform"
)

# Bandweite auslesen und zuweisen
(OB <- bw_selection$bws[1])
```

```
[1] 5.078001
```

5.4.2 Deskriptive Statistiken

Zur Reproduktion von Tabelle 1 aus Basten und Betz (2013) erzeugen wir eine nach Kantonen gruppierte Zusammenfassung der Daten und berechnen deskriptive Statistiken. Wie im Paper berücksichtigen wir hierbei nur Gemeinden innerhalb der geschätzten optimalen Bandweite `OB`.

```r
# Datensatz für Reproduktion von Table 1 formatieren
T1 <- BastenBetz %>%
  filter(abs(bord) < OB) %>%
  mutate(
    vaud = ifelse(
      test = vaud == 1,
      yes = "Waadt",
      no = "Freiburg"
    ),
    prot = prot * 100
  ) %>%
  group_by(vaud) %>%
  summarise(
    across(
      everything(),
      list(
```

Ergebnisses von `rdbwselect`. Wir verwenden nachfolgend die Schätzung `OB`.

```
        Mean = mean,
        SD = sd,
        N = length
      )
    )
  ) %>%
  pivot_longer(
    cols = -vaud,
    names_to = c("variable", "statistic"),
    names_sep = "_"
  )
```

Für die tabellarische Darstellung transformieren wir in ein weites Format, sodass die Tabelle die deskriptive Statistiken spaltenweise für die Kantone zeigt.

```
# Daten in weites Format überführen
T1_wider <- T1 %>%
  pivot_wider(
    names_from = c("vaud", "statistic")
  )
```

Die Tabelle erzeugen wir mit `gt::gt()`.

```
# Tabelle mit gt() erzeugen
T1_wider %>%
  gt(rowname_col = "Variable") %>%
  tab_spanner_delim(
    delim = "_",
  ) %>%
 tabopts
```

Tabelle 5.3: Datensatz `BastenBetz` – Zusammenfassende Statistiken

	Freiburg			Waadt		
variable	Mean	SD	N	Mean	SD	N
gini	0.302	0.029	49	0.367	0.052	84
prot	9.428	5.695	49	83.245	11.411	84

bord	−2.327	1.274	49	2.493	1.201	84
pfl	48.239	4.774	49	39.508	5.723	84
pfr	43.049	2.634	49	39.19	5.025	84
pfi	52.642	2.94	49	47.086	3.368	84

Die Statistiken in Tabelle 5.3 scheinen konsistent mit der (historischen) Verteilung der Religionszugehörigkeit und politischen Einstellung gemäß der Hypothese: Im überwiegend katholischen Freiburg finden wir eine größere Einkommensungleichheit und höhere aus Wahlergebnissen abgeleitete Präferenzen für Freizeit, Umverteilung sowie staatliche Interventionen.

5.4.3 Modellspezifikation und First-Stage-Ergebnisse

Die Kantone Waadt und Freiburg haben bis heute mehrheitlich protestantische bzw. katholische Gemeinden. Die Verteilung von Protestantismus ist also, u.a. aufgrund von Bevölkerungsbewegungen, nicht mehr deterministisch. An der Kantonsgrenze besteht jedoch eine deutliche Diskontinuität im Anteil protestantischer Einwohner, die auf die historische Verteilung der Religionszugehörigkeit zurückzuführen ist. Damit kann ein FRDD implementiert werden, bei dem die Distanz zur Grenze (`bord`) die zentrierte Laufvariable ist und die Zugehörigkeit zum Kanton Waadt (`vaud`) ein Instrument für die Behandlungsvariable (`prot`) ist.

Wir nutzen die Funktion `rdrobust::rdplot` um diesen Zusammenhang für verschiedene Bandweiten anhand des linearen Interaktionsmodells

$$\begin{aligned} prot_i = \alpha_0 + \alpha_1 vaud_i + \alpha_2 bord_i \\ + \alpha_3 bord_i \times vaud_i + u_i \end{aligned} \tag{5.10}$$

grafisch darzustellen. Dies ist die First-Stage-Regression für die 2SLS-Schätzung der Behandlungseffekte.

```
# Reproduktion von Abbildung 3 in Basten und Betz (2013)
plots_BB <- list(
  # gesch. optimale Bandweite
  p_OB = rdplot(
    y = BastenBetz$prot,
    x = BastenBetz$bord,
    h = c(OB, OB),
    x.label = "Distanz zur Grenze (bord)",
```

```
  y.label = "Anteil Protestanten (prot)",
  title = "Gesch. Bandweite",
  p = 1,
  nbins = c(6, 14),
  masspoints = "off"
),

# Bandweite 10
p_BW10 = rdplot(
  y = BastenBetz$prot,
  x = BastenBetz$bord,
  h = c(10, 10),
  x.label = "Distanz zur Grenze (bord)",
  y.label = "Anteil Protestanten  (prot)",
  title = "Bandweite = 10",
  p = 1,
  nbins = c(6, 14),
  masspoints = "off"
),

# Bandweite 20
p_BW20 = rdplot(
  y = BastenBetz$prot,
  x = BastenBetz$bord,
  h = c(20, 20),
  x.label = "Distanz zur Grenze (bord)",
  y.label = "Anteil Protestanten  (prot)",
  title = "Bandweite = 20",
  p = 1,
  nbins = c(6, 14),
  masspoints = "off"
),

# Gesamter Datensatz
p_G = rdplot(
  y = BastenBetz$prot,
  x = BastenBetz$bord,
  x.label = "Distanz zur Grenze (bord)",
```

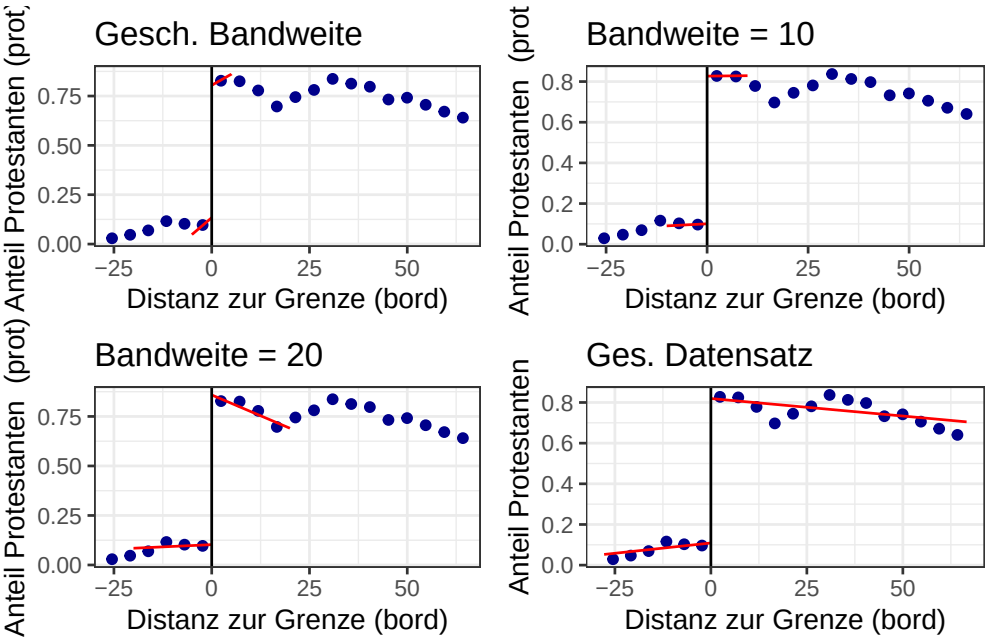


Abbildung 5.9: First-Stage-Regressionen

```
    y.label = "Anteil Protestanten",
    title = "Ges. Datensatz",
    p = 1,
    nbins = c(6, 14),
    masspoints = "off"
  )
)
```

Wir sammeln die Ergebnisse in einem Plot-Gitter mit `cowplot::plot_grid()`.

```
# Reproduktion von Abbildung 3 in Basten und Betz (2013)
plot_grid(
  plotlist = map(plots_BB, ~ .$rdplot), ncol = 2
)
```

Die Grafiken in Abbildung 5.9 zeigen deutliche Hinweise auf die Diskontinuität in `prot` nahe der Kantonsgrenze. Die Größe des geschätzten Sprungs scheint nur wenig sensitiv gegenüber der gewählten Bandweite zu sein. Die Signifikanz des Effekts können wir anhand der jeweiligen KQ-Regressionen beurteilen.[6]

[6] Wir nutzen `update()` um die Regression mit weniger Code für verschiedene Bandweiten zu schätzen.

```r
# Reproduktion der First-Stage-Regressionen
# s. Tabelle 2 in Basten und Betz (2013)

# (1) BW = OB
FS1 <- lm(
  formula = prot ~ vaud + bord + vaud * bord,
  data = BastenBetz %>%
    filter(
      abs(bord) <= OB
    )
)

# (2) BW = 10
FS2 <- update(
  FS1,
  data = BastenBetz %>%
    filter(
      abs(bord) <= 10
    )
)

# (3) BW = 20
FS3 <- update(
  FS1,
  data = BastenBetz %>%
    filter(
      abs(bord) <= 20
    )
)

# (4) Ges. Datensatz
FS4 <- update(
  object = FS1,
  data = BastenBetz
)
```

```r
# Tabellarische Darstellung
modelsummary(
```

```
  list(
    "BW = OB"= FS1,
    "BW = 10" = FS2,
    "BW=20" = FS3,
    "Ges. Datensatz" = FS4
  ),
  vcov = "HC1",
  stars = T,
  gof_map = "nobs",
  output = "gt"
) %>%
  tabopts()
```

Tabelle 5.4: First-Stage Regressionen

	BW = OB	BW = 10	BW=20	Ges. Datensatz
(Intercept)	0.134***	0.100***	0.103***	0.109***
	(0.017)	(0.013)	(0.010)	(0.009)
vaud	0.671***	0.726***	0.756***	0.710***
	(0.034)	(0.022)	(0.018)	(0.014)
bord	0.017**	0.001	0.001	0.002*
	(0.006)	(0.003)	(0.001)	(0.001)
vaud × bord	-0.006	-0.001	-0.009***	-0.004***
	(0.012)	(0.005)	(0.003)	(0.001)
Num.Obs.	133	207	312	509

+ p < 0.1, * p < 0.05, ** p < 0.01, *** p < 0.001

Für die geschätze Bandweite schätzen wir einen hochsignifikanten Sprung in `prot` von etwa 67% an der Kantonsgrenze. Auch für größere Bandweiten von 10km und 20km sowie für den gesamten Datensatz finden wir vergleichbare signifikante Effekte, was eine bei zunehmender Distanz zur Grenze persistente Diskrepanz der Religionszugehörigkeit bestätigt.

5.4.4 Second-Stage-Ergebnisse

Wir schätzen nun den LATE von Protestantismus für die Outcome-Variablen `gini`, `pfl`, `pfi` und `pfr`, vgl. Tabelle 5.2. Die Spezifikation für die Second-

Stage-Regression der FRDD-Schätzung ist

$$Y_i = \gamma_0 + \gamma_1 \widehat{prot}_i + \gamma_2 bord_i + \gamma_3 bord_i \times vaud_i + e_i, \tag{5.11}$$

wobei $\widehat{prot}_i$ angepasste Werte aus der KQ-Schätzung von (5.10) mit Bandweite `OB` sind. Dazu erzeugen wir zunächst eine angepasste Version des Objekts `BastenBetz`, welche nur Gemeinden innerhalb der Bandweite enthält.

```
# Gemeinden innerhalb der Bandweite filtern
BastenBetz_OB <- BastenBetz %>%
  filter(
    abs(bord) <= OB
  )
```

Zur Illustration schätzen wir nun die Second-Stage-Regression für $Y = pfl$.

```
# Second-Stage-Regression für `pfl`
BastenBetz_OB %>%
  mutate(
    prot_fitted = fitted(FS1)
    ) %>%

lm(
  pfl ~ prot_fitted + bord + vaud:bord,
  data = .
) %>%
  summary()
```

```
Call:
lm(formula = pfl ~ prot_fitted + bord + vaud:bord, data = .)

Residuals:
     Min       1Q   Median       3Q      Max
-12.8870  -3.8621  -0.0423   3.4993  12.1636

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  50.5275     1.9721  25.621  < 2e-16 ***
prot_fitted -13.4600     3.1749  -4.240 4.24e-05 ***
```

```
bord          0.4380     0.6528   0.671    0.503
bord:vaud    -0.3636     0.7939  -0.458    0.648
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.433 on 129 degrees of freedom
Multiple R-squared:  0.383, Adjusted R-squared:  0.3686
F-statistic: 26.69 on 3 and 129 DF,  p-value: 1.704e-13
```

Der Koeffizient `prot_fitted` ist der gesuchte Behandlungseffekt. Beachte, dass die von `summary()` berechneten Standardfehler ungültig sind, weil diese die zusätzliche Unsicherheit durch die Berechnung von $\widehat{prot}$ über die First-Stage-Regression nicht berücksichtigen. Nachfolgend nutzen wir `AER::ivreg()`, um komfortabel gültige (heteroskedastie-robuste) Inferenz betreiben zu können.[7]

```
# Schätzung mit 2SlS
# s. Tabelle 4 in Basten und Betz (2013)
#
# Wir instrumentieren Treatment (`prot1980s`) mit dem
↪  Schwellenindikator (`vaud`)
# ivreg: exogene Variablen instrumentieren sich selbst,
↪  daher
# ' | vaud * borderdis '
library(AER)
# (1) Präferenz für Freizeit
SS_pfl <- ivreg(
  formula = pfl ~ prot + bord:vaud + bord | vaud * bord,
  data = BastenBetz_OB
)

# (2) Präferenz für Umverteilung
SS_pfr <- update(
  object = SS_pfl,
  formula = pfr ~ prot + bord:vaud + bord | vaud * bord,
)
```

[7] Die Autoren geben an, robuste SEs zu nutzen. Das scheint nicht der Fall zu sein, denn `vcov = "HC0"` liefert die Ergebinsse im Paper. Die von Stata berechneten HC1-SEs weichen ab. Dies ändert allerdings nichts an der Signifikanz der Koeffizienten. Wir nutzen `vcov = "HC1"`.

```r
# (3) Präferenz für Intervention
SS_pfi <- update(
  object = SS_pfl,
  formula = pfi ~ prot + bord:vaud + bord | vaud * bord,
)

# (4) Einkommensungleichheit
SS_gini <- update(
  object = SS_pfl,
  formula = pfi ~ prot + bord:vaud + bord | vaud * bord,
)
```

```r
# Tabellarische Darstellung
modelsummary(
  list(
    "(1) Freizeit"= SS_pfl,
    "(2) Umverteilung" = SS_pfr,
    "(3) Intervention" = SS_pfi,
    "(4) Ungleichheit" = SS_gini
  ),
  vcov = "HC1",
  stars = T,
  gof_map = "nobs",
  output = "gt"
) %>%
  tabopts()
```

Tabelle 5.5: Ergebnisse der Second-Stage-Regressionen

	(1) Freizeit	(2) Umverteilung	(3) Intervention	(4) Ungleichheit
(Intercept)	50.528***	44.560***	52.871***	52.871***
	(1.918)	(0.950)	(1.063)	(1.063)
prot	-13.460***	-5.061*	-6.487***	-6.487***
	(3.161)	(2.161)	(1.738)	(1.738)
bord	0.438	0.444	-0.165	-0.165
	(0.639)	(0.357)	(0.332)	(0.332)
bord × vaud	-0.364	-0.909	0.011	0.011
	(0.811)	(0.561)	(0.432)	(0.432)

Num.Obs.	133	133	133	133

+ p < 0.1, * p < 0.05, ** p < 0.01, *** p < 0.001

Die Koeffizienten von `prot` in Tabelle 5.5 sind die mit 2SLS ermittelten erwarteten Behandlungseffekte einer 100%-Reformation (d.h. von 100% katholisch zu 100% protestantisch) für eine durchschnittliche Gemeine nahe der Kantonsgrnze. Es handelt sich jeweils um einen lokalen durchschnittlichen Behandlungseffekt (LATE). Gem. der Definition der abhängigen Variablen, interpretieren wir die Koeffizienten von `prot` in de Regressionen (1), (2) und (3) als erwartete Prozentänderung durch Reformation. Der Koeffizient in Regression (4) gibt die erwartete Änderung des Gini-Index an. Sämtliche geschätzte Effekte sind signifikant und haben ein mit der Hypothese der Autoren konsistentes negatives Vorzeichen.

Die Ergebnisse sind Evidenz, dass Protestantismus zu verringerter Präferenz für Freizeit, Umverteilung sowie wirtschaftspolitische Intervention seitens des Staats führt. Auch die ökonomische Ungleichheit ist signifikant geringer, als in einer durchschnittlichen vollständig katholischen Gemeinde.

5.4.5 Addendum: FRDD-Schätzung mit `rdrobust()`

Die Funktion `rdrobust::rdrobust()` erlaubt die Schätzung von SRDD und FRDD mit einer Vielzahl von Optionen, s. `?rdrobust`. Dies erleichtert die Schätzung mehrerer Modellspezifikationenen und Bandweiten. Mit dem nachstehenden Befehl schätzen wir den LATE von Reformation auf die Präferenz für Umverteilung anhand lokaler quadratischer Regression. Der Output gibt einen Überblick der Bandweitenschätzung sowie der 2 Stufen des 2SLS-Schätzers, inkl. robuster Inferenzstatistiken.

```
pfr_rdr <- rdrobust(
  y = BastenBetz$pfr,
  x = BastenBetz$bord,
  fuzzy = BastenBetz$prot,
  p = 2,
  kernel = "uniform",
  vce = "HC1"
)

pfr_rdr %>%
```

```
summary()
```

```
Fuzzy RD estimates using local polynomial regression.

Number of Obs.                  509
BW type                       mserd
Kernel                      Uniform
VCE method                      HC1

Number of Obs.                  127          382
Eff. Number of Obs.              85          131
Order est. (p)                    2            2
Order bias  (q)                   3            3
BW est. (h)                  10.796       10.796
BW bias (b)                  22.271       22.271
rho (h/b)                     0.485        0.485
Unique Obs.                      97          261

First-stage estimates.

=============================================================================
        Method     Coef. Std. Err.         z     P>|z|      [
        95% C.I. ]
=============================================================================
  Conventional     0.701     0.039    17.782     0.000
  [0.624 , 0.778]
        Robust         -         -    15.837     0.000
        [0.599 , 0.768]
=============================================================================

Treatment effect estimates.

=============================================================================
        Method     Coef. Std. Err.         z     P>|z|      [
        95% C.I. ]
=============================================================================
  Conventional    -5.047     2.254    -2.239     0.025
  [-9.464 , -0.629]
```

```
        Robust          -          -     -2.210      0.027
        [-10.114 , -0.607]
================================================================================
```

Auch für die quadratische Spezifikation erhalten wir mit -5.047 ein vergleichbares signifikantes Ergebnis für den LATE von Protestantismus auf Umverteilung, vgl. Spalte (2) in Tabelle 5.5.

Mit der Option `bwselect = "msetwo"` kann die Bandweite jeweils für die lokale Regression links- und rechtssetig des Schwellenwerts geschätzt werden.

```
pfr_rdr %>%
  update(bwselect = "msetwo") %>%
  summary()
```

```
Fuzzy RD estimates using local polynomial regression.

Number of Obs.                  509
BW type                      msetwo
Kernel                      Uniform
VCE method                      HC1

Number of Obs.                  127          382
Eff. Number of Obs.              51          134
Order est. (p)                    2            2
Order bias  (q)                   3            3
BW est. (h)                   5.340       11.387
BW bias (b)                  13.917       22.330
rho (h/b)                     0.384        0.510
Unique Obs.                      97          261

First-stage estimates.

================================================================================
        Method     Coef. Std. Err.         z     P>|z|      [
        95% C.I. ]
================================================================================
  Conventional     0.649     0.046    14.216     0.000
  [0.560 , 0.739]
        Robust          -         -    11.970     0.000
        [0.534 , 0.743]
```

```
==============================================================================

Treatment effect estimates.

==============================================================================
        Method     Coef. Std. Err.          z     P>|z|      [
        95% C.I. ]
==============================================================================
  Conventional    -7.487     3.378    -2.216     0.027
  [-14.109 , -0.866]
        Robust          -          -    -2.156     0.031
        [-14.750 , -0.704]
==============================================================================
```

Trotz Diskrepanz der geschätzten Bandweiten erhalten wir eine größere aber vergleichbare Schätzung für einen negativen Effekt.

6 Regularisierte Regression

In diesem Kapitel betrachten wir Varianten von Koeffizientenschätzern im linearen Modell

$$Y_i = \beta_1 X_{1,i} + \cdots + \beta_k X_{k,i} + u_i, \quad i = 1, \ldots, n, \tag{6.1}$$

deren Motivation die Schätzung von $\boldsymbol{\beta} := (\beta_1, \ldots, \beta_k)'$ in Anwendungen ist, in denen der KQ-Schätzer

$$\begin{aligned} \widehat{\boldsymbol{\beta}} &= \arg\min_{\boldsymbol{\beta}} \mathrm{RSS}(\boldsymbol{\beta}) \\ &= \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (Y_i - \beta_1 X_{1,i} + \cdots + \beta_k X_{k,i})^2 \end{aligned} \tag{6.2}$$

keine stabile Schätzung zulässt oder nicht eindeutig definiert ist, und damit gar nicht erst berechnet werden kann. Solche Szenarien ergeben sich in der empirischen Forschung, wenn die Regressoren stark korreliert sind und/oder das Modell viele Regressoren enthält ($k \lesssim n$), oder das Regressionsproblem hoch-dimensional ist ($k > n$).

Regularisierte Regressionsschätzer begegnen dieser Problematik mit einer Modifikation der Verlustfunktion RSS in (6.2),

$$\mathrm{RSS}(\boldsymbol{\beta}, p, \lambda) := \mathrm{RSS}(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_p. \tag{6.3}$$

Hierbei ist $\lambda > 0$ ein Tuning-Parameter und $p \geq 1$ definiert die p-Norm des Koeffizientenvektors,

$$\|\boldsymbol{\beta}\|_p := \left(\sum_{j=1}^{k} |\beta_j|^p \right)^{1/p} > 0. \tag{6.4}$$

Wegen $\lambda \|\boldsymbol{\beta}\|_p > 0$ kann die p-Norm des Koeffizientenvektors $\boldsymbol{\beta}$ das Optimierungsproblem

$$\min_{\boldsymbol{\beta}} \mathrm{RSS}(\boldsymbol{\beta}, p, \lambda) | \, p, \, \lambda$$

derart restringieren, dass die geschätzten Koeffizienten

$$\widehat{\boldsymbol{\beta}}_{p,\lambda} := \arg\min_{\boldsymbol{\beta}} \text{RSS}(\boldsymbol{\beta}, p, \lambda)$$

im Erwartungswert absolut kleiner ausfallen als bei der KQ-Schätzung: Der Schätzer ist in Richtung 0 verzerrt.[1] Dieser Effekt der Regularisierung wird in der Literatur als *Shrinkage* bezeichnet.

Die grundlegenden Eigenschaften des Schätzers $\widehat{\boldsymbol{\beta}}_{p,\lambda}$ werden maßgeblich durch den Parameter p bestimmt, der hinsichtlich des zu lösenden Regressionsproblems *a priori* gewählt wird.[2]

Shrinkage ist eine Motivation für die Anwendung regularisierter Schätzer in Modellen, die auch mit KQ geschätzt werden könnten. Um dies zu verstehen, nehmen wir an, dass die Gauss-Markov-Annahmen in (6.1) gelten. Dann hat der KQ-Schätzer die kleinste Varianz unter allen *unverzerrten* Schätzern. Aufgrund der Shrinkage fallen regularisierte Schätzer zwar nicht unter das Gauss-Markov-Theorem, können dafür aber eine geringere Varianz haben als KQ. Schätzer mit solchen Eigenschaften sind nützlich, wenn eine unverzerrte Schätzung von $\boldsymbol{\beta}$ nicht unser primäres Ziel ist: Für Vorhersagen kann es hilfreich sein, etwas Verzerrung bei der Koeffizientenschätzung in Kauf zu nehmen, um eine hinreichend große Varianzreduktion zu erreichen, sodass ein geringerer erwarteter Vorhersagefehler als für KQ resultiert. Hierbei liegt, eine Abwägung zwischen Verzerrung und Varianz (*Bias Variance Tradeoff*) vor, der durch den Regularisierungsparameter λ beeinflusst wird.

Für die Berechnung des Schätzers in empirischen Anwendungen wird λ meist datengetrieben (mit Cross Validation oder einem Informationskriterium) geschätzt oder mit einer analytisch fundierten Faustregel gewählt.

Nachfolgend betrachten wir zwei häufig verwendete regularisierte Schätzer, die sich durch die Wahl $p = 1$ (Lasso Regression) bzw. $p = 2$ (Ridge Regression) ergeben und illustrieren ihre Anwendung mit R.

6.1 Ridge Regression

Ridge Regression wurde von Hoerl und Kennard (1970) als Alternative zur KQ-Schätzung bei hoch-korrelierten Regressoren eingeführt. Die Verlustfunktion

[1] Beachte, dass für $\lambda = 0$ die Verlustfunktion des KQ-Schätzers folgt.

[2] D.h. wir wählen p, um einen Schätzer mit für die konkrete Anwendung hilfreichen Eigenschaften zu erhalten.

lautet

$$\text{RSS}(\boldsymbol{\beta}, p = 2, \lambda) = \text{RSS}(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_2, \tag{6.5}$$

d.h. der Parameter λ reguliert den Einfluss eines ℓ_2-Strafterms

$$\|\boldsymbol{\beta}\|_2 = \sqrt{\sum_{j=1}^{k} \beta_j^2}$$

auf die Verlustfunktion $\text{RSS}(\boldsymbol{\beta}, p = 2, \lambda)$. Der Ridge-Schätzer ergibt sich als

$$\widehat{\boldsymbol{\beta}}_\lambda^{\text{R}} := \arg\min_{\boldsymbol{\beta}} \text{RSS}(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_2. \tag{6.6}$$

Für Das Optimierungsproblem (6.6) kann wir aus den Bedingungen 1. Ordnung

$$-2\boldsymbol{X}'(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}) + 2\lambda\boldsymbol{\beta} = \boldsymbol{0} \tag{6.7}$$

die analytische Lösung

$$\widehat{\boldsymbol{\beta}}_\lambda^{\text{R}} = (\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{I}_p)^{-1}\boldsymbol{X}'\boldsymbol{Y}, \tag{6.8}$$

bestimmt werden, wobei $\boldsymbol{I}_k$ die $k \times k$ Einheitsmatrix ist. Aus Gleichung (6.8) kann die Wirkungsweise des Strafterms $\lambda\|\boldsymbol{\beta}\|_2$ abgeleitet werden: Ridge Regression modifiziert die Diagonale der zu invertierenden Matrix $\boldsymbol{X}'\boldsymbol{X}$ durch Addition von $\lambda > 0$. Dies ist hilfreich, wenn

- $k \geq n$ und damit $\boldsymbol{X}'\boldsymbol{X}$ nicht invertiertbar (singulär) ist. Dann kann der KQ-Schätzer nicht berechnet werden.[3] Die Inverse $(\boldsymbol{X}'\boldsymbol{X} + \lambda\boldsymbol{I}_p)^{-1}$ hingegen existiert unter milden Bedingungen.
- hohe Kollinearität vorliegt, sodass $(\boldsymbol{X}'\boldsymbol{X})^{-1}$ zwar existiert, aber zu einer instablilen KQ-Schätzung mit hoher Varianz führt.

Für eine grafische Betrachtung des Optimierungskalküls (6.6) betrachten wir die äquivalente Darstellung als Lagrange-Problem

$$\widehat{\boldsymbol{\beta}}_\lambda^{\text{R}} := \arg\min_{\|\boldsymbol{\beta}\| < t} \text{RSS}(\boldsymbol{\beta}). \tag{6.9}$$

[3] Beispiel: `X <- matrix(rnorm(100), ncol = 10)`. Vergleiche `solve(t(X) %*% X)` und `solve(t(X) %*% X + diag(.01, nrow = 10))`

In der folgenden interaktiven Grafik illustrieren wir das Optimierungsproblem (6.9) sowie den resultierenden Schätzer der Koeffizienten (β_1, β_2) in einem multiplen Regressionsmodell mit den Regressoren X_1 und X_2.

- Die blaue Ellipse ist die Menge aller Schätzwerte $\left(\widehat{\beta}_1,\, \widehat{\beta}_2\right)$ für den angegebenen Wert von RSS. Im Zentrum der Ellipse liegt der KQ-Schätzer, welcher RSS minimiert.
- Der blaue Kreis ist die Menge aller Koeffizienten-Paare (β_1, β_2), welche die Restriktion $\beta_1^2 + \beta_2^2 \leq t$ erfüllen. Beachte, dass die Größe des Kreises nur durch den Parameter t bestimmt wird, welcher für einen vorgegebenen Wertebereich variiert werden kann.
- Der blaue Punkt ist der Ridge-Schätzer $(\widehat{\beta}_{1,t}^R,\, \widehat{\beta}_{2,t}^R)$. Dieser ergibt sich als Schnittpunkt zwischen der blauen RSS-Ellipse und der Restriktionsregion und variiert mit t. Die gestrichelte rote Kurve zeigt den Ridge-Lösungspfad.
- Für kleine Werte t drückt die Shrinkage die geschätzten Koeffizienten Richtung 0, wobei der Lösungspfad i.d.R. nicht-linear verläuft, d.h. die Shrinkage auf den Koeffizienten ist grundsätzlich unterschiedlich. Die Lösung $(\widehat{\beta}_{1,t}^R,\, \widehat{\beta}_{2,t}^R) = (0, 0)$ existiert nur als Grenzwert für $t \to 0$.
- Beachte, dass der Effekt von t auf die Schätzung umgekehrt für λ verläuft: Größere λ führen zu stärkerer Regularisierung.

Diese Interaktive Komponente des Buchs ist nur in der Online-Version verfügbar.

6.1.1 Eigenschaften des Schätzers

Der Ridge-Schätzer $\widehat{\boldsymbol{\beta}}_\lambda^{\mathrm{R}}$ ist nicht invariant gegenüber der Skalierung der Regressoren. Für empirische Daten sollte daher vorab eine Standardisierung der erklärenden Variablen durchgeführt werden.[4] Um die Eigenschaften des Ridge-Schätzers besser zu verstehen, betrachten wir hier den Fall orthonormaler

[4] Bspw. mit der Funktion `scale()`.

Regressoren $\boldsymbol{X}_j$.[5] Dann ist

$$\widehat{\beta}^{\text{R}}_{\lambda,j} = (1+\lambda)^{-1} \cdot \widehat{\beta}_j, \quad j = 1, \ldots, k, \tag{6.10}$$

d.h. der Ridge-Schätzer skaliert die KQ-Lösung mit einem von λ abhängigen Faktor.[6]

Wir illustrieren dies, indem wir den Zusammenhang zwischen KQ- und Ridge-Schätzer im orthonormalen Fall als R-Funktion `ridge_ortho()` implementieren und für die Parameterwerte $\lambda \in \{0, 0.5, 2\}$ plotten.

```
library(tidyverse)

# Funktion für Rige Regression bei orthonormalen
↪  Regressoren
ridge_ortho <- function(KQ, lambda) {
  1/(1 + lambda) * KQ
}
```

```
# KQ-Schätzer gegen Ridge-Schätzer plotten
dat <- tibble(KQ = seq(-1, 1, .01))

ggplot(dat) +
  geom_function(fun = ridge_ortho,
                args = list(lambda =  0),
                lty = 2) +
  geom_function(fun = ridge_ortho,
                args = list(lambda = .5),
                col = "red") +
  geom_function(fun = ridge_ortho,
                args = list(lambda = 2),
                col = "blue") +
  xlim(-.4, .4) +
  xlab("KQ-Schätzer von beta_1") +
  ylab("Ridge-Schätzer von beta_1")
```

Abbildung 6.1 zeigt, dass der Ridge-Schätzer eine lineare Transformation des KQ-Schätzers (gestrichelte Linie) ist. Größere Werte des Regularisierungspara-

[5] Orthonormalität heißt $\boldsymbol{X}_i'\boldsymbol{X}_j = 1$ für $i = j$ und 0 sonst. Dann ist $\boldsymbol{X}'\boldsymbol{X} = \boldsymbol{I}_k$.
[6] $(1+\lambda)^{-1}$ wird auch als *Shrinkage-Faktor* bezeichnet.

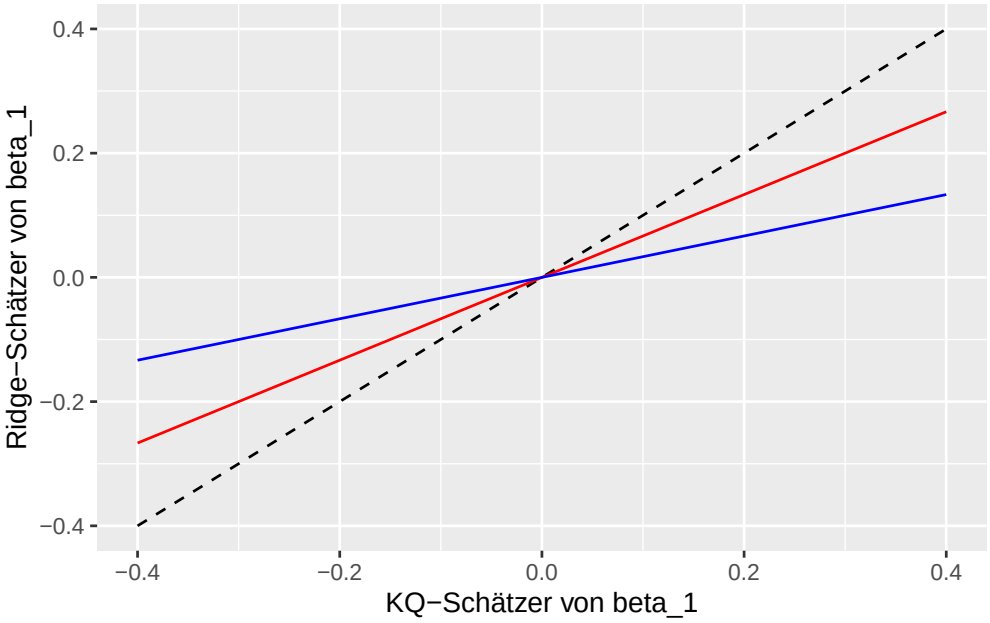


Abbildung 6.1: Shrinkage des OLS-Schätzers bei Ridge Regression

meters λ führen zu stärkerer Shrinkage des Koeffizientenschätzers in Richtung 0. Die ℓ_2-Norm führt zu proportional zum Absolutwert des KQ-Schätzers verlaufender Shrinkage: Größere Koeffizienten werden stärker bestraft als kleine Koeffizienten.

Die Eigenschaft

$$\mathrm{E}\left(\widehat{\boldsymbol{\beta}}^{\mathrm{R}}_{\lambda,j}\right) = (1+\lambda)^{-1} \cdot \beta_j$$

zeigt, dass $\widehat{\boldsymbol{\beta}}^{\mathrm{R}}_{\lambda,j}$ (für fixes $\lambda > 0$) nicht erwartungstreu für β_j ist. Weiterhin ist

$$\begin{aligned}\mathrm{Var}\left(\widehat{\beta}^{\mathrm{R}}_{\lambda,j}\right) &= \mathrm{Var}\left(\widehat{\beta}_j\right) \cdot \left(\frac{\lambda}{1+\lambda^2}\right) \\ &= \sigma^2 \cdot \left(\frac{\lambda}{1+\lambda^2}\right),\end{aligned}$$

wobei σ^2 die Varianz des Regressionsfehlers u ist. Wegen $\lambda < (1+\lambda)^2$ für $\lambda > 0$ gilt

$$\mathrm{Var}\left(\widehat{\beta}^{\mathrm{R}}_{\lambda,j}\right) < \mathrm{Var}\left(\widehat{\beta}_j\right).$$

Der Ridge-Schätzer hat also eine kleinere Varianz als der KQ-Schätzer. Diese Eigenschaften können auch für korrelierte Regressoren gezeigt werden.

6.1.2 Ridge Regression mit `glmnet`

Wir zeigen nun anhand simulierter Daten, wie der Ridge-Lösungspfad mit dem R-Paket **glmnet** berechnet werden kann. Wir erzeugen zunächst Daten gemäß der Vorschrift

$$
\begin{aligned}
Y_i &= \boldsymbol{X}_i' \boldsymbol{\beta} + u_i, \\
\beta_j &= \frac{5}{j^2}, \qquad\qquad j = 1, \ldots, 5, \\
\beta_j &= -\frac{5}{(j-5)^2}, \quad j = 6, \ldots, 10, \\
\boldsymbol{X}_i &\sim N(\mathbf{0}, \boldsymbol{\Sigma}), \quad u_i \overset{u.i.v.}{\sim} N(0,1), \quad i = 1, \ldots, 25.
\end{aligned}
\tag{6.11}
$$

Hierbei wird $\boldsymbol{\Sigma}$ so definiert, dass jeder Regressor $N(0,1)$-verteilt ist und eine Korrelation von 0.8 mit allen anderen Regressoren aufweist. Mit der Vorschrift für die β_j stellen wir sicher, dass es wenige Variablen gibt, die Y stark beeinflussen, da der Absolutbetrag der Koeffizienten in j abnimmt.[7]

```
library(gendata)
set.seed(1234)

# Parameter definieren
N <- 80
k <- 10

coefs <- 5/(1:(k/2))^2
beta <- c(coefs, -coefs)

# Beobachtungen simulieren
X <- as.matrix(
  genmvnorm(
    k = k,
    cor = rep(.8, (k^2-k)/2),
    n = N)
  )
Y <- X %*% beta + rnorm(N)
```

[7] Für bessere Interpretierbarkeit der Grafischen Auswertung, wählen wir positive und negative Koeffizienten mit gleichem Bertag.

Wir schätzen nun ein Modell mit allen 10 Regressoren mit `glmnet`. Beachte, dass für den Ridge-Strafterm `alpha = 0` gesetzt werden muss.[8]

```
library(glmnet)

# Ridge-Regression anpassen
ridge_fit <- glmnet(
  x = X,
  y = Y,
  alpha = 0 # für Ridge-Strafterm
)
```

Der Lösungspfad der Ridge-Schätzung kann nach Transformation der geschätzen Koeffizienten und der zugehörigen λ-Werte in ein langes Format überführt und komfortabel mit `ggplot2` dargestellt werden.

```
# Lambda-Sequenz auslesen
lambdas <- ridge_fit$lambda

# Ridge-Schätzung für Lambdas im langen Format
as.matrix(ridge_fit$beta) %>%
  as_tibble() %>%
  rownames_to_column("Variable") %>%
  pivot_longer(-Variable) %>%
  group_by(Variable) %>%
  mutate(lambda = lambdas) %>%

  # Grafik mit ggplot erzeugen
  ggplot(
    mapping = aes(
      x = lambda,
      y = value,
      col = Variable
    )
  ) +
  geom_line() +
  ylab("gesch. Koeffizienten") +
  scale_x_log10("log_10(lambda)")
```

[8] `alpha` ist ein Mischparameter im Algorithmus für elastic net, siehe `?glmnet`.

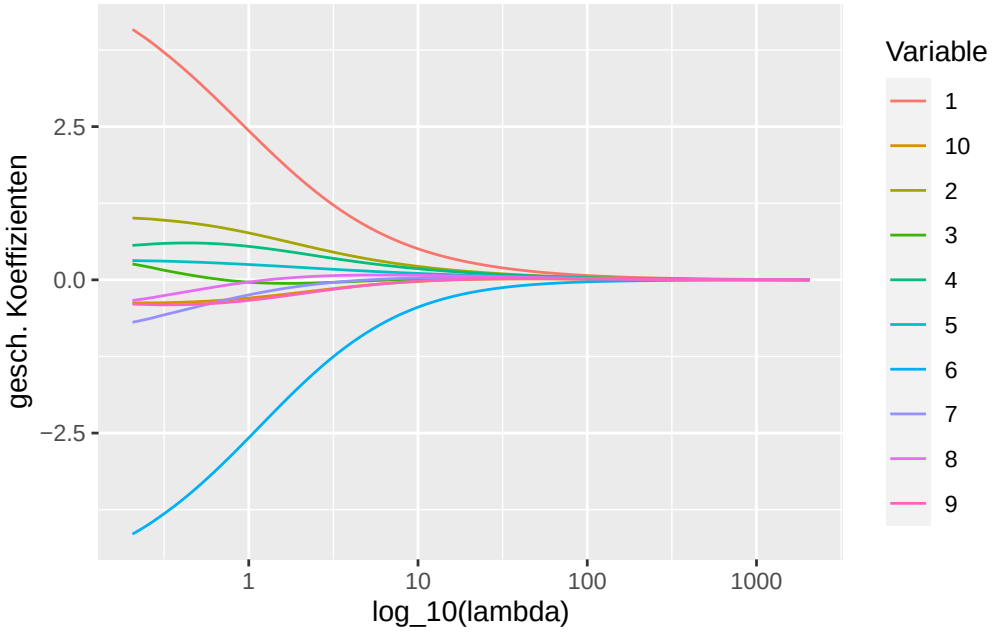


Abbildung 6.2: Lösungspfad für Ridge-Schätzung

Abbildung 6.2 zeigt den nicht-linearen Verlauf der Shrinkage auf den geschätzten Modellkoeffizienten. Die Koeffizienten werden mit zunehmendem λ von der KQ-Lösung ausgehend (linkes Ende der Skala) in Richtung 0 gezwungen.

Über die Funktion `cv.glmnet()` kann ein optimales λ mit Cross Validation (CV) ermittelt werden. Ähnlich wie bei `glmnet()` wird für die Validierung automatisch eine λ-Sequenz erzeugt. Wir nutzen `autoplot()` aus dem R-Paket `ggfortify` für die Visualisierung der Ergebnisse mit `ggplot2`.

```
library(ggfortify)

# Cross-validierte Bestimmung von lambda
ridge_cvfit <- cv.glmnet(
  y = Y,
  x = X,
  intercept = F,
  alpha = 0
)

# Ergebnisse plotten
ridge_cvfit %>%
  autoplot(label.n = 0)
```

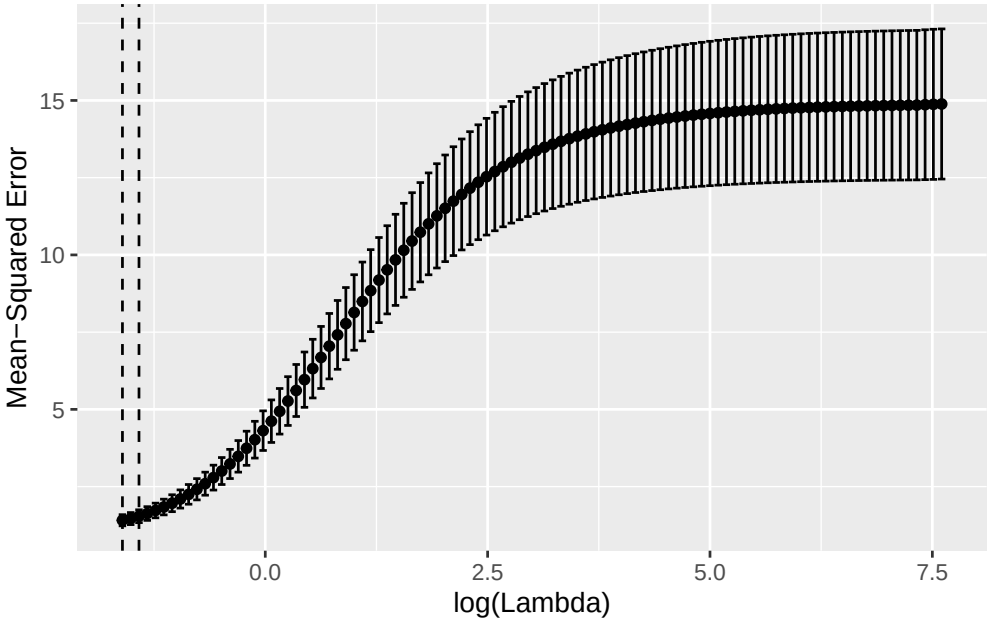


Abbildung 6.3: Lösungspfad für Ridge-Schätzung

Abbildung 6.3 zeigt `ridge_cvfit$lambda.min`, das optimale λ mit dem geringsten CV Mean-Squarred-Error (linke gestrichelte Linie) und `ridge_cvfit$lambda.1se`, das größte λ, welches innerhalb einer Standardabweichung entfernt ist (rechte gestrichelte Linie).[9] Wir berechnen die Schätzung für `lambda.min`.

```
(
  ridge_coefs <- coef(
    object = ridge_cvfit,
    s = ridge_cvfit$lambda.min
  )
)
```

```
11 x 1 sparse Matrix of class "dgCMatrix"
                    s1
(Intercept)  .
X1           4.1302194
X2           1.0245661
X3           0.3139297
X4           0.5697498
X5           0.2928664
```

[9] Die Wahl von `lambda.1se` ist eine Heuristik, welche die Schätzunsicherheit berücksichtigt und zu einem "sparsameren" Modell tendiert.

```
X6          -4.1693524
X7          -0.7509305
X8          -0.3844761
X9          -0.3841997
X10         -0.4078514
```

Wir schätzen das Modell nun mit KQ und vergleichen die Koeffizienten mit der Ridge-Schätzung.

```r
# KQ-Schätzung durchführen
KQ_fit <- lm(Y ~ X - 1)

# Koeffizienten auslesen und transformieren:
tibble(
  Ridge = as.matrix(ridge_coefs)[2:11, ],
  KQ = KQ_fit$coefficients
) %>%
  mutate(j = factor(1:10)) %>%
  pivot_longer(
    cols = Ridge:KQ,
    names_to = "Methode",
    values_to = "Koeffizient"
  ) %>%

# Bar-Plot für Koeffizientenvergleich erzeugen
  ggplot(
    mapping = aes(
      x = j,
      y = Koeffizient,
      fill = Methode
    )
  ) +
  geom_bar(
    position = "dodge",
    stat = "identity",
    width = .5
  )
```

Der Vergleich anhand von Abbildung 6.4 zeigt deutlich, dass Ridge Regression im Vergleich mit KQ zu absolut kleineren Koeffizientenschätzungen tendiert.

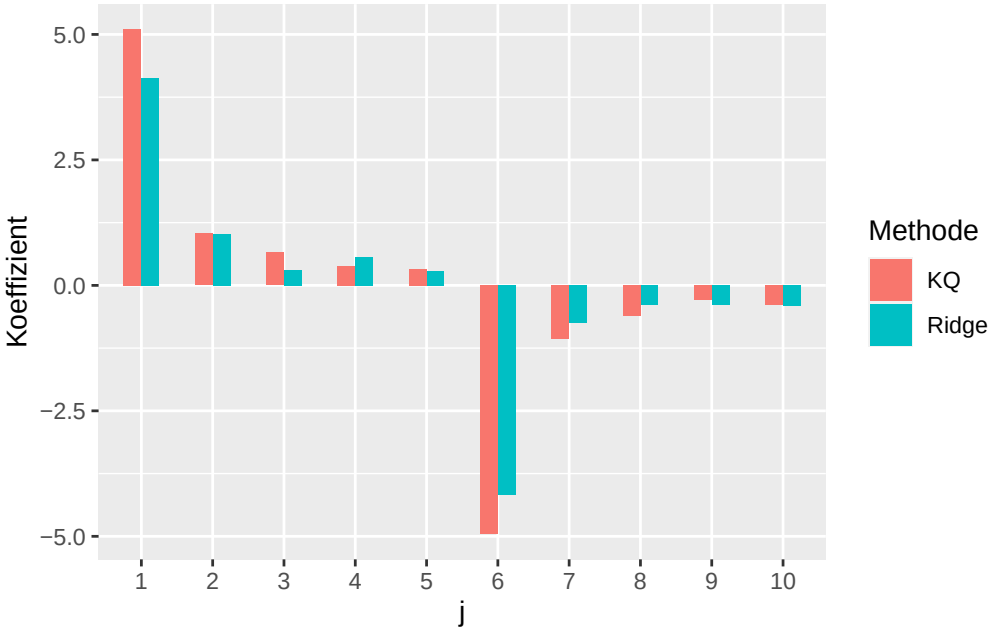


Abbildung 6.4: Koeffizientenvergleich: Ridge vs. KQ

Inwiefern dies Konsequenzen für die Prognosegüte der Schätzung hat, können wir Anhand eines Testdatensatzes bestimmen. Hierzu vergleichen wir die mittleren Fehler (MSE) bei der Prognose von Y für die Beobachtungen im Testdatensatz. Für die Simulation des Testdatensatzes nutzen wir erneut die Vorschrift (6.11) um 80 neue Beobachtungen zu erzeugen.

```
# Test-Datensatz erstellen
set.seed(4321)
# Regressoren
new_X <- as.matrix(
  genmvnorm(
    k = k,
    cor = rep(.85, (k^2-k)/2),
    n = N
  )
)
# Abh. Variable
new_Y <- new_X %*% beta + rnorm(N)
```

Für beide Methoden können wir `predict()` für die Prognosen von Y für den Testdatensatz (`new_Y`) nutzen.

```r
# Ridge: Vorhersage von new_Y für Test-Datensatz
Y_predict_ridge <- predict(
  object = ridge_cvfit,
  newx = new_X,
  s = ridge_cvfit$lambda.min
)

# Ridge: MSE für Test-Datensatz berechnen
mean((Y_predict_ridge - new_Y)^2)
```

```
[1] 1.288457
```

Die Vorhersage für `lm()` benötigt dieselben Variablennamen wie im angepassten Modell, s. `KQ_fit$coefficients`.

```r
# Test-Datensatz für predict.lm() formatieren
new_X <- as.data.frame(new_X)
colnames(new_X) <- paste0("X", 1:k)

# KQ: Vorhersage von new_Y für Test-Datensatz
Y_predict_KQ <- predict(
  object = KQ_fit,
  newdata = new_X
)

# KQ: MSE für Test-Datensatz berechnen
mean((Y_predict_KQ - new_Y)^2)
```

```
[1] 29.33797
```

Die Ergebnisse zeigen, dass der Ridge-Schätzer trotz seiner Verzerrung einen deutlich geringeren mittleren Vorhersagefehler für die Testdaten erzielt als der KQ-Schätzer. Diese Eigenschaft der Koeffizientenschätzung kann die Prognosegüte von Ridge Regression gegenüber der KQ-Regression verbessern.

6.1.3 Beispiel: Vorhersage von Abschlussnoten in Mathe

Zur Illustration von Ridge Regression nutzen wir den Datensatz `SP` aus Cortez und Silva (2008).[10] `SP` enhält Beobachtungen zu Leistungen von insgesamt 100 Schülerinnen und Schülern im Fach Mathematik in der Sekundarstufe an zwei portugiesischen Schulen. Neben der Abschlussnote in Mathe (`G3`, Skala von 0 bis 20) beinhaltet `SP` diverse demografische, soziale und schulbezogene Merkmale, die mithilfe von Schulberichten und Fragebögen erhoben wurden. Ziel ist es, ein Modell für die Prognose von `G3` anzupassen.

Wir lesen zunächst die Daten (im .csv-Format) ein.

```
# Daten einlesen
SP <- read_csv(file = "datasets/SP.csv")
```

Ein Überblick zeigt, dass der Großteil der Regressoren aus kategorialen Variablen mit sozio-ökonomischen Informationen besteht.

```
# Überblick
glimpse(SP)
```

```
Rows: 100
Columns: 31
$ school     <chr> "GP", "GP", "GP", "MS", "GP", "GP", "GP",
"GP", "GP", "GP",~
$ sex        <chr> "M", "M", "F", "F", "M", "F", "F", "F", "F",
"F", "M", "M",~
$ age        <dbl> 17, 18, 19, 17, 16, 16, 19, 16, 16, 16, 18,
16, 15, 17, 17,~
$ address    <chr> "R", "R", "U", "U", "U", "U", "U", "U", "U",
"R", "U", "U",~
$ famsize    <chr> "GT3", "GT3", "LE3", "GT3", "LE3", "GT3",
"GT3", "GT3", "GT~
$ Pstatus    <chr> "T", "T", "T", "T", "A", "T", "T", "T", "A",
"T", "T", "T",~
$ Medu       <dbl> 1, 4, 3, 2, 3, 2, 0, 2, 3, 4, 4, 2, 1, 2, 2,
3, 3, 4, 4, 2,~
```

[10] Wir verwenden eine Auszug aus dem Orignaldatensatz, der nebst ausführlicher Variablenbeschreibung hier verfügbar ist.

```
$ Fedu       <dbl> 2, 3, 2, 2, 4, 3, 1, 1, 1, 4, 4, 2, 2, 3, 2,
3, 1, 3, 4, 2,~
$ Mjob       <chr> "at_home", "teacher", "services", "other",
"services", "oth~
$ Fjob       <chr> "other", "services", "other", "at_home",
"other", "other", ~
$ reason     <chr> "home", "course", "reputation", "home",
"home", "reputation~
$ guardian   <chr> "mother", "mother", "other", "mother",
"mother", "mother", ~
$ traveltime <dbl> 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1,
2, 1, 1, 1, 1,~
$ studytime  <dbl> 2, 3, 2, 3, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 1,
1, 2, 3, 1, 2,~
$ failures   <dbl> 0, 0, 1, 0, 0, 0, 3, 0, 3, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0,~
$ schoolsup  <chr> "no", "no", "no", "no", "yes", "yes", "no",
"no", "no", "no~
$ famsup     <chr> "no", "no", "yes", "no", "yes", "yes", "yes",
"no", "yes", ~
$ paid       <chr> "no", "no", "yes", "no", "no", "yes", "no",
"no", "yes", "n~
$ activities <chr> "no", "no", "no", "yes", "yes", "yes", "no",
"no", "no", "y~
$ nursery    <chr> "yes", "yes", "no", "yes", "yes", "yes",
"no", "yes", "yes"~
$ higher     <chr> "yes", "yes", "yes", "yes", "yes", "yes",
"no", "yes", "yes~
$ internet   <chr> "no", "yes", "yes", "no", "yes", "no", "no",
"yes", "yes", ~
$ romantic   <chr> "no", "yes", "yes", "yes", "no", "no", "no",
"yes", "no", "~
$ famrel     <dbl> 3, 5, 4, 3, 5, 4, 3, 4, 2, 2, 1, 5, 4, 5, 3,
5, 4, 4, 5, 5,~
$ freetime   <dbl> 1, 3, 2, 4, 3, 4, 4, 5, 3, 4, 4, 4, 3, 3, 4,
4, 5, 2, 3, 4,~
$ goout      <dbl> 3, 2, 2, 3, 3, 3, 2, 2, 3, 4, 2, 4, 2, 3, 4,
2, 4, 2, 3, 4,~
$ Dalc       <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 1, 1, 1,
1, 2, 1, 1, 1,~
```

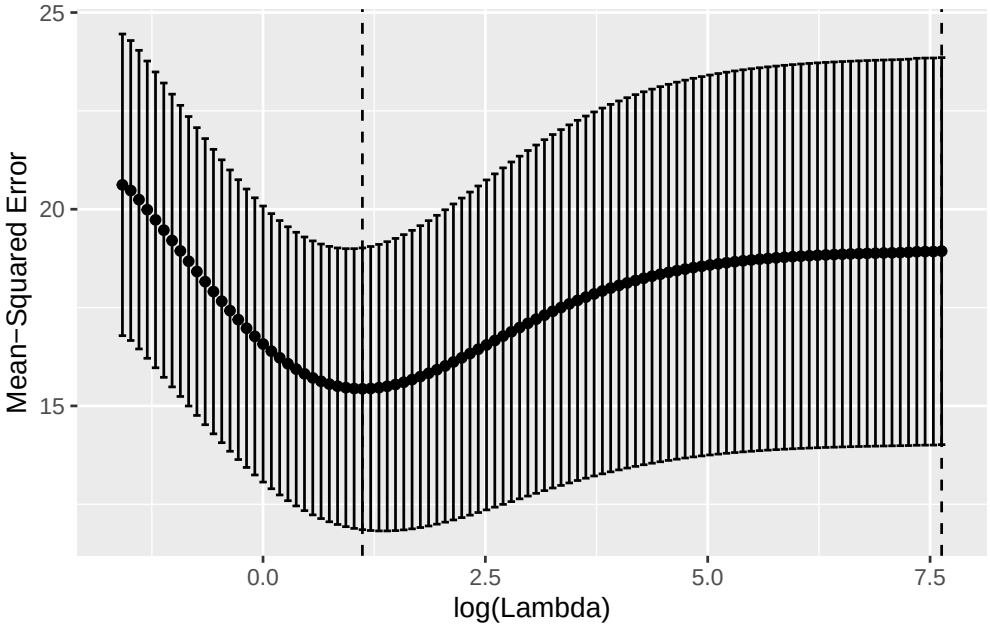
```
$ Walc       <dbl> 5, 2, 2, 1, 1, 3, 1, 1, 2, 3, 2, 4, 1, 3, 3,
1, 3, 2, 1, 1,~
$ health     <dbl> 3, 4, 1, 3, 5, 4, 5, 5, 4, 4, 1, 5, 5, 3, 5,
5, 1, 3, 5, 5,~
$ absences   <dbl> 4, 9, 22, 8, 4, 6, 2, 20, 5, 6, 5, 0, 2, 2,
12, 0, 17, 0, 4~
$ G3         <dbl> 10, 16, 11, 11, 11, 10, 9, 12, 7, 11, 16, 12,
9, 12, 12, 13~
```

Um die Prognosegüte des Modells beurteilen zu können, partitionieren wir `SP` zufällig in einen Test- sowie einen Trainingsdatensatz (mit 30 und 70 Beobachtungen), jeweils für die Regressoren und die abhängige Variable.

```r
# ID für Beobachtungen im Testdatensatz zufällig erzeugen
set.seed(1234)
ID <- sample(1:nrow(SP), size = 30)

# Regressoren aufteilen
SP_test <- SP[ID,]
SP_train <- SP[-ID,]

# Abh. Variable aufteilen
Y_test <- SP_test$G3
Y_train <- SP_train$G3
```

Als nächstes passen wir ein Ridge-Regressionsmodell für alle Regressoren in `SP_train` an und ermitteln ein optimales λ mit Cross Validation. Beachte, dass `cv.glmnet` nicht für Regressoren im `data.frame`/`tibble`-Format ausgelegt ist, sondern ein `matrix`-Format erwartet. Wir transformieren `SP_train` daher mit `data.matrix()`.

```r
# Ridge-Regression und CV für Trainingsdaten
SP_fit_cv <- cv.glmnet(
  x = data.matrix(SP_train %>% select(-G3)),
  y = Y_train,
  alpha = 0
)

# CV-Ergebnisse für lambda visualisieren
```




```
SP_fit_cv %>%
  autoplot(label.n = 0)
```

Wie für das Beispiel mit simulierten Daten erhalten wir mit `predict()` Vorhersagen für die erzielte Punktzahl. Beachte, dass wir den MSE nicht für die Trainingsdaten `SP_train`, sondern für die Testdaten `SP_test` berechnen.

```
# Prognose von G3 anhand des Ridge-Modells
Y_predict_ridge <- predict(
  object = SP_fit_cv,
  newx = data.matrix(
    SP_test %>%
      select(-G3)
    ),
  s = SP_fit_cv$lambda.min
)

# MSE für Testdaten berechnen
mean((Y_predict_ridge - Y_test)^2)
```

```
[1] 21.13249
```

Auch in diesem empirischen Beispiel zeigt ein Vergleich der MSEs, dass Ridge Regression dem KQ-Schätzer hinsichtlich der Vorhersagegüte überlegen ist.

```r
# Modell mit KQ schätzen
SP_fit_KQ <- lm(G3 ~ ., SP_train)

# Prognose
Y_predict_KQ <- predict(
  object = SP_fit_KQ,
  newdata = SP_test %>%
    select(-G3)
)

# Testset-MSE berechnen
mean((Y_predict_KQ - Y_test)^2)
```

```
[1] 29.76893
```

Der MSE für Ridge ist mit 21.13 deutlich kleiner als 29.77, der MSE für KQ.

Für die Interpretation der Ridge-Schätzung erweitern den Code für die **ggplot2**-Grafik der Koeffizienten-Pfade um eine vertikale Linie des mit CV ermittelten λ und fügen mit dem Paket **ggrepel** Labels für die Pfade der größten Koeffizienten hinzu.

```r
library(ggrepel)

# Lambda-Sequenz auslesen
lambdas <- SP_fit_cv$lambda

# Ridge-Schätzung für Lambdas im langen Format
df <- as.matrix(SP_fit_cv$glmnet.fit$beta) %>%
  as_tibble() %>%
  mutate(
    Variable = rownames(SP_fit_cv$glmnet.fit$beta)
  ) %>%
  pivot_longer(-Variable) %>%
  group_by(Variable) %>%
  mutate(lambda = lambdas)

# Grafik mit ggplot erzeugen
df %>%
```

```
ggplot(
  mapping = aes(
    x = lambda,
    y = value,
    col = Variable
  )
) +
geom_line() +
geom_label_repel(
  data = df %>%
    filter(lambda == min(lambdas)),
  mapping = aes(label = Variable),
  seed = 1234,
  size = 5,
  max.overlaps = 8,
  nudge_x = -.5) +
ylab("gesch. Koeffizienten") +
scale_x_log10("log_10(lambda)") +
geom_vline(
  xintercept = SP_fit_cv$lambda.min,
  col = "red",
  lty = 2
) +
theme(legend.position = "none")
```

Abbildung 6.5 gibt Hinweise darauf, dass neben der Schulzugehörigkeit und Indikatoren für schulische Leistung (bspw. `failures`) sozio-ökonomische Prädiktoren wie `internet` (Internetzugang zuhause), `Pstatus` (Zusammenleben der Eltern) und `address`/`traveltime` (sozialer Status) relevante Variablen zu sein scheinen.

Das optimale $\lambda_{cv} \approx 0.21$ (gestrichelte rote Linie in Abbildung 6.5) führt zu deutlicher Shrinkage, was eine mögliche Erklärung für den besseren Testset-MSE von Ridge Regression ist: Die Koeffizienten von Variablen mit wenig Erklärungskraft werden durch die Regularisierung in Richtung 0 gezwungen und reduzieren so die Varianz der Vorhersage gegenüber der (idealerweise) unverzerrten KQ-Schätzung.

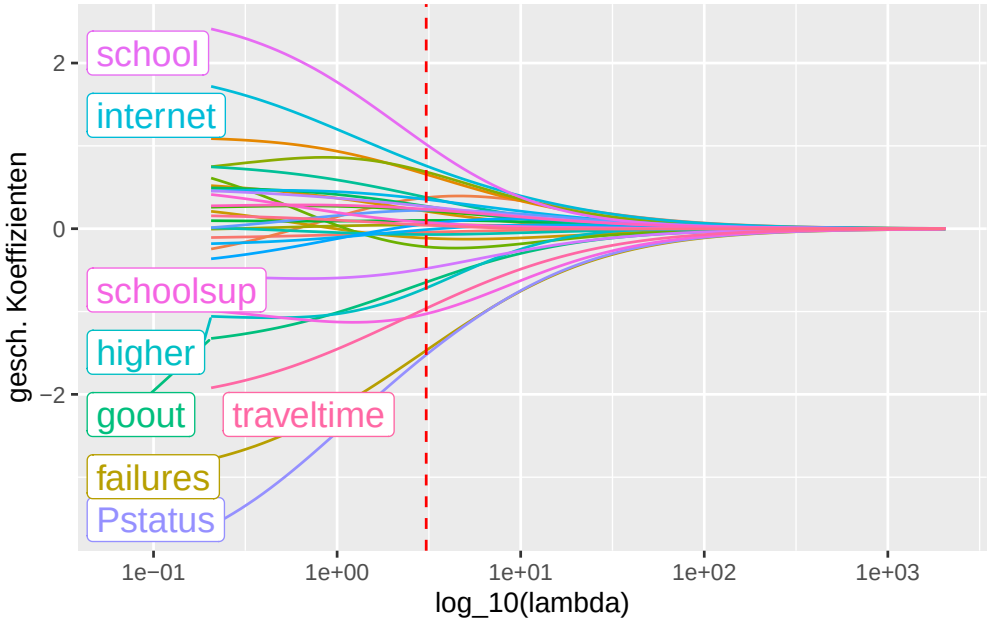


Abbildung 6.5: Lösungspfad für Ridge-Schätzung

Key Facts zu Ridge Regression

- Ridge-Regression regularisiert den KQ-Schätzer mit der ℓ_2-Norm der Koeffizienten. Diese Form von Regularisierung ist eine Alternative für KQ in Anwendungen mit mehr Regressoren als Beobachtugen ($k \geq n$) und/oder wenn KQ aufgrund starker Kollinearität eine hohe Varianz aufweist.
- Der Ridge-Schätzer $\widehat{\boldsymbol{\beta}}_\lambda^{\mathrm{R}}$ ist *nicht* erwartungstreu. Die geschätzten Koeffizienten sind auch für $n \to \infty$ verzerrt.
- Aufgrund der verzerrten Schätzung ist statistische Inferenz für Koeffizienten mit $\widehat{\boldsymbol{\beta}}_\lambda^{\mathrm{R}}$ problematisch. Anstatt für strukturelle Modelle oder die Schätzung kausaler Effekte wird Ridge Regression in der Praxis daher überwiegend für Prognosen verwendet.
- Die Wahl von λ impliziert einen Tradeoff zwischen Verzerrung und Varianz: Große λ schrumpfen die Koeffizientenschätzer Richtung 0 (mehr Verzerrung), führen aber zu einer kleineren Varianz der Schätzung. Entsprechend können Vorhersagen mit mehr Verzerrung aber weniger Varianz als mit KQ getroffen werden.
- Ridge Regression kann in R mit dem Paket `glmnet` berechnet werden.

6.2 Lasso Regression

Least Absolute Shrinkage and Selection Operator (Lasso) ist ein von Tibshirani (1996) vorgeschlagener Schätzer, der die Verlustfunktion des KQ-Schätzers um einen Strafterm für die Summe der (absoluten) Größe der Koeffizienten $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_k)'$ erweitert. Die Verlustfunktion des Lasso-Schätzers von $\boldsymbol{\beta}$ lautet

$$\text{RSS}(\boldsymbol{\beta}, p = 1, \lambda) = \text{RSS}(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1. \tag{6.12}$$

Für den Strafterm wird also die ℓ_1-norm

$$\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{k} |\beta_j|$$

verwendet. Der Lasso-Schätzer $\widehat{\boldsymbol{\beta}}^{\text{L}}_{\lambda}$ für $\boldsymbol{\beta}$ minimiert (6.12),

$$\boldsymbol{\beta}^{\text{L}}_{\lambda} = \arg\min_{\boldsymbol{\beta}} \ \text{RSS}(\boldsymbol{\beta}, p = 1, \lambda). \tag{6.13}$$

Entsprechend erhalten wir in Abhängigkeit von λ ein Kontinuum an Lösungen

$$\left\{ \widehat{\boldsymbol{\beta}}^{\text{L}}_{\lambda} \right\}_{\lambda=0}^{\lambda=\infty}, \tag{6.14}$$

der sogenannte *Lasso-Pfad*.

Das Optimierungsproblem (6.12) hat die äquivalente Darstellung

$$\begin{aligned} \widehat{\boldsymbol{\beta}}^{\text{L}}_{\lambda} &= \arg\min_{\boldsymbol{\beta}} \text{RSS}(\boldsymbol{\beta}) + \lambda \left(\|\boldsymbol{\beta}\|_1 - t \right) \\ &= \arg\min_{\|\boldsymbol{\beta}\|_1 \leq t} \text{RSS}(\boldsymbol{\beta}), \end{aligned} \tag{6.15}$$

welche über den Lagrange-Ansatz unter der Nebenbedingung $\|\boldsymbol{\beta}\|_1 \leq t$ gelöst werden kann.

Ähnlich wie der KQ-Schätzer ist der Lasso-Schätzer $\widehat{\boldsymbol{\beta}}^{\text{L}}_{\lambda}$ durch Bedingungen 1. Ordnung bestimmt. Diese Bedingungen lassen sich komfortabel in Matrix-Schreibweise darstellen als

$$-2\boldsymbol{X}_j'(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda \cdot \text{sgn}(\beta_j) = 0, \quad j = 1, \ldots, k. \tag{6.16}$$

Aus Gleichung (6.16) folgt, dass der Lasso-Schätzer aufgrund des Strafterms im Allgemeinen nicht algebraisch bestimmt werden kann.[11]

[11] Zur Bestimmung des Schätzers werden Algorithmen der nicht-linearen Optimierung genutzt.

In Abhängigkeit von λ zwingt der Lasso-Schätzer die KQ-Schätzung von β_j zu einem (absolut) kleineren Wert: Ähnlich wie bei Ridge Regression bewirkt der ℓ_1-Strafterm eine mit λ zunehmende Schrumpfung der geschätzen Koeffizienten in Richtung 0. Charakteristisch für die Lösung des Lasso-Schätzers ist, dass $\widehat{\boldsymbol{\beta}}_j^{\mathrm{L}} = 0$, wenn die Bedingung

$$\left| \boldsymbol{X}_j'(\boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}_\lambda^{\mathrm{L}}) \right| - \lambda/2 \leq 0 \tag{6.17}$$

erfüllt ist. In Abhängigkeit von λ kann der Lasso-Schätzer folglich geschätzte Regressionskoeffizienten nicht nur in Richtung 0, sondern diese auch *exakt* mit 0 schätzen und damit *Variablenselektion* betreiben. Aufgrund der mit λ zunehmenden Shrinkage bis die Bedingung (6.17) erfüllt und der Koeffizient gleich 0 gesetzt wird, bezeichnet man Lasso auch als einen *Soft Thresholding Operator*. Im nächsten Abschnitt betrachten wir die Eigenschaften von Lasso-Regularisierung unter vereinfachten Annahmen bzgl. der Regressoren.

6.2.1 Lasso ist Soft Thresholding

Wir betrachten nun eine mathematische Darstellung von Selektions- und Shrinkage-Eigenschaft des Lasso-Schätzers in einem vereinfachten Modell. Wenn die Regressoren $\boldsymbol{X}$ orthonormal zueinander sind, existiert eine analytische Lösung des Lasso-Schätzers,

$$\widehat{\boldsymbol{\beta}}_\lambda^{\mathrm{L}} = \begin{cases} \widehat{\boldsymbol{\beta}}_j - \lambda/2 & , \quad \widehat{\boldsymbol{\beta}}_j > \lambda/2 \\ 0 & , \quad |\widehat{\boldsymbol{\beta}}_j| \leq \lambda/2 \,, \\ \widehat{\boldsymbol{\beta}}_j + \lambda/2 & , \quad \widehat{\boldsymbol{\beta}}_j < \lambda/2 \end{cases} \tag{6.18}$$

wobei $\widehat{\boldsymbol{\beta}}_j$ der KQ-Schätzer von β_j ist. Anhand von (6.18) können wir die Selektionseigenschaft sowie die Schrumpfung der KQ-Koeffizientenschätzung in Abhängigkeit der durch λ regulierten ℓ_1-Strafe erkennen. Für eine Visualisierung implementieren wir (6.18) als R-Funktion `lasso_st()` und zeichnen die resultierenden Koeffizientenschätzungen für die Parameterwerte $\lambda \in \{0, 0.2, 0.4\}$.

Wir definieren zunächst die Funktion `lasso_st()`.

```
library(tidyverse)

# Funktion für Lasso soft-thresholding definieren
lasso_st <- function(KQ, lambda) {
```

```
  case_when(
    KQ > lambda/2          ~ KQ - lambda/2,
    abs(KQ) <= lambda/2    ~ 0,
    KQ < -lambda/2         ~ KQ + lambda/2,
  )
}
```

Im nächsten Schritt zeichnen wir `lasso_st()` für eine Sequenz von KQ-Schätzwerten gegeben λ.

```
# Sequenz von KQ-Schätzwerten für Illustration definieren
dat <- tibble(
  KQ = seq(-1, 1, .01)
)

# Lasso-Schätzer als Funktion des KQ-Schätzers plotten
ggplot(dat) +
  geom_function(
    fun = lasso_st,
    args = list(lambda = 0),
    lty = 2
  ) +
  geom_function(
    fun = lasso_st,
    args = list(lambda = .2),
    col = "red"
  ) +
  geom_function(
    fun = lasso_st,
    args = list(lambda = .4),
    col = "blue"
  ) +
  xlim(-.4, .4) +
  xlab("KQ-Schätzer von beta_1") +
  ylab("Lasso-Schätzer von beta_1")
```

Abbildung 6.6 zeigt, dass der ℓ_1-Strafterm des Lasso-Schätzers zu einem linearen Verlauf der auf den KQ-Schätzer (gezeichnet für $\lambda = 0$, gestrichelte Linie) applizierten Shrinkage führt: Der Lasso-Schätzer ist eine abschnittsweise-lineare

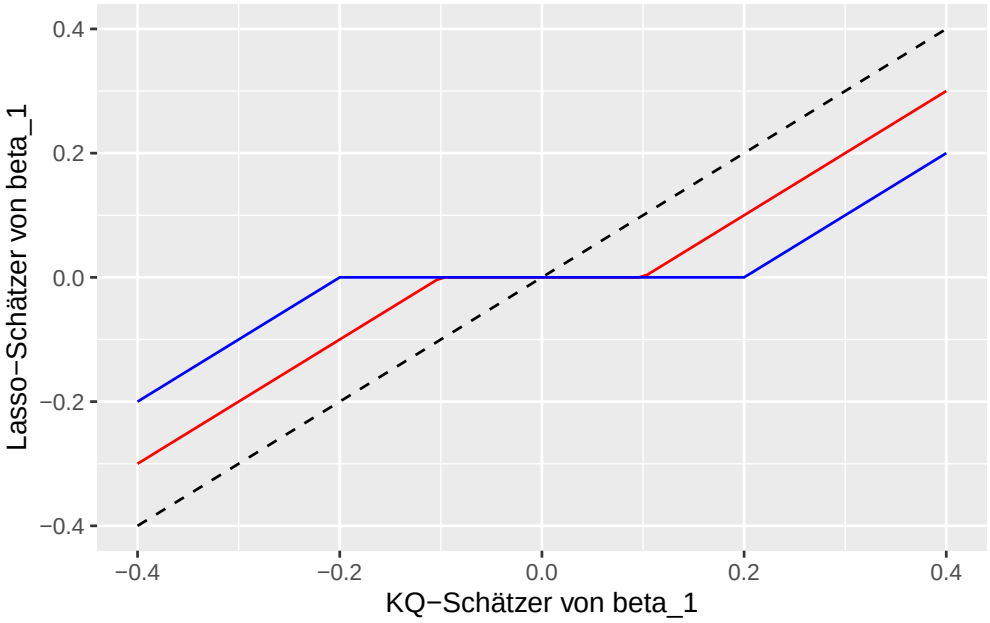


Abbildung 6.6: Shrinkage und Selektion von OLS-Koeffizienten mit Lasso

Funktion des KQ-Schätzers in λ: Je größer der Parameter λ, desto größer ist das Intervall von KQ-Schätzwerten $[-\lambda/2, \lambda/2]$, wo der Lasso-Schätzer zu Variablenselektion führt, d.h. hier den Koeffizienten β_j als 0 schätzt (rote bzw. blaue Linie).

Anhand von Abbildung 6.6 kann abgeleitet werden, dass der Lasso-Schätzer nicht invariant gegenüber der Skalierung der Regressoren ist: Die Stärke der Regularisierung durch λ ist hängt von der Magnitude des KQ-Schätzers ab. Daher müssen die Regressoren vor Berechnung der Schätzung standardsiert werden. Üblich ist hierbei eine Normierung auf einen Mittelwert von 0 und eine Varianz von 1.

Die nachstehende interaktive Grafik illustriert das Lasso-Optimierungsproblem (6.15) sowie den resultierenden Schätzer der Koeffizienten (β_1, β_2) in einem multiplen Regressionsmodell mit korrelierten Regressoren X_1 und X_2.

- Die blaue Ellipse ist die Menge aller Schätzwerte $\left(\widehat{\beta}_1, \widehat{\beta}_2\right)$ für den angegebenen Wert von RSS. Im Zentrum der Ellipse liegt der KQ-Schätzer, welcher RSS minimiert.
- Das graue Quadrat ist die Menge aller Koeffizienten-Paare (β_1, β_2), welche die Restriktion $|\beta_1| + |\beta_2| \leq t$ erfüllen. Beachte, dass die Größe dieser Region nur durch den Parameter t bestimmt wird.
- Der blaue Punkt ist der Lasso-Schätzer $(\widehat{\boldsymbol{\beta}}^L_{1,t}, \widehat{\boldsymbol{\beta}}^L_{2,t})$. Dieser ergibt sich als Schnittpunkt zwischen der blauen RSS-Ellipse und der Restriktionsregion

und variiert mit t. Die gestrichelte rote Linie zeigt den Lasso-Lösungspfad.

- Für kleine Werte, erhalten wir starke Shrinkage auf $\widehat{\beta}_{1,t}$ bis zum Wertebereich $t \leq 50$, wo $\widehat{\boldsymbol{\beta}}_{1,t}^L = 0$. Hier erfolgt Variablenselektion: Die Regularisierung führt zu einem geschätzten Modell, das lediglich X_2 als erklärende Variable enthält. In diesem Bereich von t bewirkt die Shrinkage, dass $\widehat{\boldsymbol{\beta}}_{2,t}^L \to 0$ für $t \to 0$.

Diese Interaktive Komponente des Buchs ist nur in der Online-Version verfügbar.

Beachte, dass der rote Lasso-Pfad (die Menge aller Lasso-Lösungen) äquivalent als Funktion von λ im Optimierungsproblem (6.12) dargestellt werden kann. Implementierungen mit statistischer Software berechnen die Lasso-Lösung häufig in Abhängigkeit von λ. Ein Algorithmus hierfür ist LARS.

6.2.2 Berechnung der Lasso-Lösung mit dem LARS-Algorithmus

Für die Berechnung des Lasso-Lösungspfads kann der LARS-Algorithmus von Efron u. a. (2004) im Lasso-Modus genutzt werden.[12] Der Lasso-Lösungspfad beinhaltet geschätzte Koeffizienten über ein Intervall für λ, welches sämtliche Modellkomplexitäten zwischen der (trivialen) Lösung mit maximaler Shrinkage auf allen Koeffizienten (λ groß, alle gesch. Koeffizienten sind 0) und der unregularisierten Lösung ($\lambda = 0$, KQ-Schätzung) abbildet. Der LARS-Algorithmus erzeugt den Lösungspfad sequentiell, sodass die Schätzung als Funktion von λ veranschaulicht werden kann, ähnlich wie bei Ridge Regression.

Wir zeigen nun anhand simulierter Daten, wie Lasso-Lösungen mit dem R-Paket `lars` berechnet werden können. Hierfür erzeugen wir Daten gemäß der Vorschrift

$$
\begin{aligned}
Y_i &= \boldsymbol{X}_i' \boldsymbol{\beta}_v + u_i \\
\boldsymbol{\beta}_v &= (-1.25, -.75, 0, 0, 0, 0, 0, .75, 1.25)' \qquad (6.19) \\
\boldsymbol{X}_i &\sim N(\boldsymbol{0}, \boldsymbol{I}_{9\times 9}), \quad u_i \overset{u.i.v.}{\sim} N(0,1), \quad i = 1, \ldots, 25.
\end{aligned}
$$

[12] LARS steht für *Least Angle Regression*.

```r
library(lars)
set.seed(1234)

# Parameter definieren
N <- 25
beta_v <- c(-1.25, -.75, 0, 0, 0, 0, 0, .75, 1.25)

# Beobachtungen simulieren
X <- matrix(rnorm(N * 9), ncol = 9)
Y <- X %*% beta_v + rnorm(N)
```

Entsprechend des DGP passen wir ein Modell ohne Konstante an. Damit `lars::lars()` den Lösungspfad des Lasso-Schätzers berechnet, muss `type = "lasso"` gewählt werden.[13]

```r
# Lösungen des Lasso-Schätzers mit LARS berechnen
(
  fit_lars <- lars(
    x = X,
    y = Y,
    intercept = F,
    type = "lasso" # Wichtig: Lasso-Modus
  )
)
```

```
Call:
lars(x = X, y = Y, type = "lasso", intercept = F)
R-squared: 0.858
Sequence of LASSO moves:

Var  9 2 8 1 3 5 4 7 6
Step 1 2 3 4 5 6 7 8 9
```

Die Zusammenfassung zeigt, dass der LARS-Algorithmus als erstes die (relevante) Variable X_9 aktiviert.[14] Mit abnehmender Regularisierung (kleinere λ) werden in den nächsten 3 Schritten die übrigen relevanten Variablen X_2,

[13] `lars()` standardisiert die Regressoren standardmäßig (aufgrund des DGPs hier nicht nötig).
[14] Aktivierung meint die Aufnahme einer Variable in der Modell gegeben eines hinreichend kleinen λ.

X_8 und X_1 aktiviert. Über die weiteren Schritte nähert der Algorithmus die Lösung an die *saturierte* Schätzung (das Modell mit allen neun Regressoren) an und aktiviert schrittweise die übrigen, irrelevanten Variablen.

Wir visualisieren die geschätzen Koeffizienten an jedem Schritt des Lösungspfads als Funktion von λ. In der Praxis wird der Regularisierungsparameter häufig auf der natürlichen log-Skala dargestellt.

```
# Transformation in ein weites Format
fit_lars$beta %>%
  as_tibble() %>%
  mutate(
    lambda = c(fit_lars$lambda, 1e-2)
  ) %>%
  pivot_longer(
    cols = 1:9,
    names_to = "Variable",
    values_to = "gesch. Koeffizient"
  ) %>%

# Visualisierung mit ggplot
  ggplot(
    mapping = aes(
      x = log(lambda),
      y = `gesch. Koeffizient`,
      color = Variable
    )
  ) +
  geom_line()
```

Abbildung 6.7 zeigt, dass die Shrinkage der geschätzten Koeffizienten nach der Aktivierung rasch abnimmt und sich für kleine Werte von λ der KQ-Lösung annähert. Wir sehen auch, dass es einen Bereich von λ-Werten gibt, für die das wahre Modell mit den Variablen X_1, X_2, X_8 und X_9 selektiert werden kann. Je nach Ziel der Analyse kann es sinnvoll sein, ein λ in diesem Intervall zu schätzen.

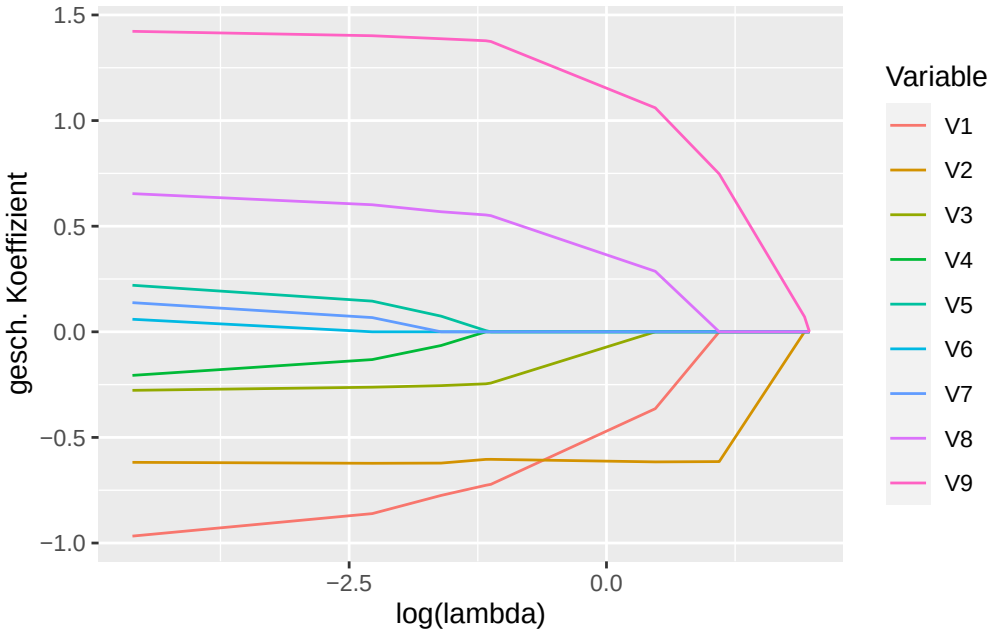


Abbildung 6.7: LARS-Lösungspfad für Lasso-Schätzung

6.2.3 Wahl des Regularisierungsparameters λ für den Lasso-Schätzer

Wie zuvor bei Ridge Regression muss in empirischen Anwendungen ein Wert für den Tuning-Parameter λ gewählt werden. Hierbei besteht die Herausforderung darin, einen geeigneten Wert zu finden, der zu wünschenswerten Eigenschaften des resultierenden Modells führt. So ist für gute Vorhersagen wichtig, dass das Modell nicht zu sehr an die Daten angepasst ist (*Overfitting*), um eine gute Generalisierung auf neue Daten zu ermöglichen. Gleichzeitig muss das Modell flexibel genug sein, um wesentliche Eigenschaften des datenerzeugenden Prozesses hinreichend gut zu erfassen. In der Regel wird hierbei eine sparsame Modellierung angestrebt, die nur eine Teilmenge der Prädiktoren nutzt.

In der Praxis werden verschiedene Verfahren verwendet, um den Wert für den Tuning-Parameter λ zu bestimmen. Gängige Methoden sind *Cross Validation* (CV) und Informationskriterien. In Abhängigkeit der Methode und der Daten ergeben sich ober- oder unterparameterisierte Modelle. Aufgrund der Implementierung im R-Paket `lars` betrachten wir CV.[15] Wir zeigen nachfolgend anhand der simulierten Daten aus dem letzten Abschnitt, wie für die LARS-Schätzung ein optimales λ mit leave-one-out CV (LOO-CV) bestimmt werden kann. Hierzu nutzen wir `lars::cv.lars()` unter Verwendung derselben Argumente wie zuvor im Aufruf von `lars()`.

[15] Chetverikov, Liao, and Chernozhukov (2020) zeigen, dass CV zu konsistenter Modellselektion führen kann.

```
# LARS-Lösungen mit CV evaluieren
fit_lars_cv <- cv.lars(
  x = X,
  y = Y,
  intercept = F,
  normalize = T,
  type = "lasso",
  plot.it = F,
  K = N # für LOO-CV
)
```

Das Objekt `fit_lars_cv` ist eine Liste mit den CV-Ergebnissen. Wir können diese einfach mit `ggplot` visualisieren. `index` ist hierbei das Verhältnis der ℓ_1-Norm des Lasso-Schätzers für einen spezifischen Wert von λ und der ℓ_1-Norm des KQ-Schätzers. Das optimale λ wird so implizit geschätzt. `cv.error` ist der mit CV geschätzte MSE.

```
# CV-MSE
fit_lars_cv %>%
  as_tibble() %>%

  ggplot(
    mapping = aes(
      x = index,
      y = cv.error
    )
  ) +
  geom_line() +
  xlab("|beta_lambda| / |beta|") +
  ylab("CV-MSE")
```

In der Grafik erkennen wir ein Minimum des CV-MSEs bei etwa 0.73.

```
# CV-MSE-minimierendes Lambda bestimmen
ID <- which.min(fit_lars_cv$cv.error) # Index

(
  fraction_opt <- fit_lars_cv$index[ID]
```

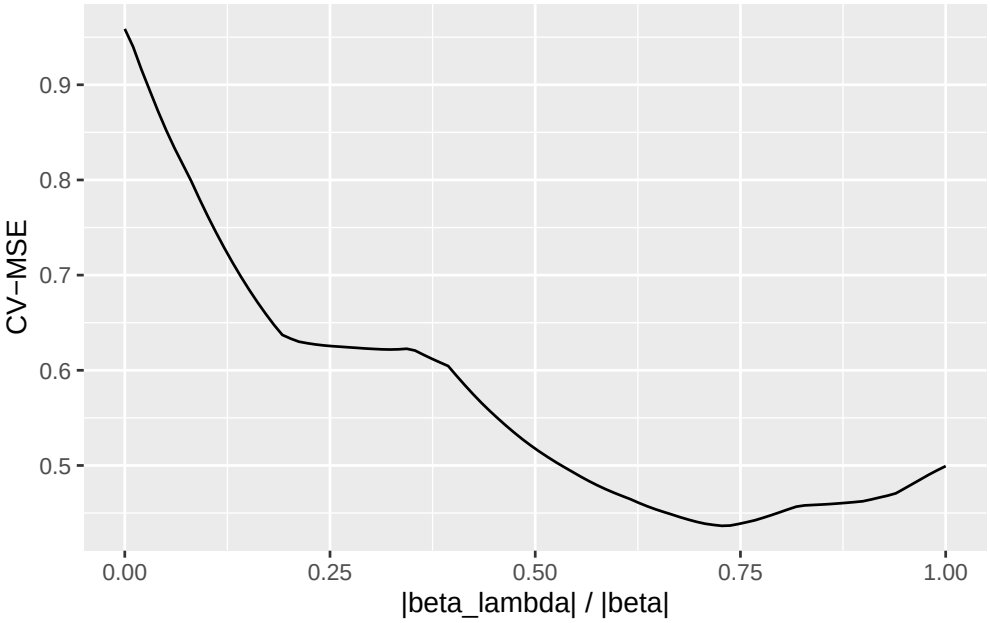


Abbildung 6.8: CV-MSE und relative Position von λ auf dem Lassopfad

```
)
```

```
[1] 0.7272727
```

Die geschätzten Koeffizienten für die optimale Regularisierung können mit `coef()` ausgelesen werden.

```
# LARS-Lasso-Fit für optimales lambda bestimmen
coef(
  object = fit_lars,
  s = fraction_opt,
  mode = "fraction"
)
```

```
[1] -0.6513191 -0.6060906 -0.1946089  0.0000000  0.0000000
0.0000000  0.0000000
[8]  0.4977908  1.3122407
```

Das Ergebnis veranschaulicht die Selektionseigenschaft von Lasso: Gemäß DGP (6.19) sind die Variablen X_3 bis X_7 *irrelevante* Prädiktoren für Y; ihre wahren Koeffizienten sind 0. In der kreuzvalidierten Lasso-Schätzung erreicht die Regularisierung, dass die Koeffizienten der Variablen X_4 bis X_7 tatsächlich mit 0 geschätzt werden. Wir schätzen für das mit CV bestimmte λ also ein

leicht überspezifiziertes Modell mit den Regressoren X_1, X_2, X_3, X_8 und X_9. Beachte, dass die Lasso-Schätzung einen Kompromiss impliziert: Die Varianz der Schätzung ist geringer als die des KQ-Schätzers im Modell mit allen Variablen.[16] Aufgrund der Regularisierung sind die mit Lasso geschätzten Koeffizienten der relevanten Variablen jedoch in Richtung 0 verzerrt.

Einen positiven Effekt dieses Kompromisses beobachten wir anhand des mittleren Vorhersagefehlers für Daten, die *nicht* zur Berechnung des Schätzers verwendet wurden. Wir vergleichen den Vorhersagefehler nachfolgend anhand eines solchen simulierten Test-Datensatzes mit 25 neuen Beobachtungen. Den Vorhersagefehler bestimmen wir als MSE zwischen den vorhergesagten und den tatsächlichen Ausprägungen für Y.

```
# Test-Datensatz erstellen
set.seed(4321)
new_X <- matrix(rnorm(N * 9), ncol = 9)
new_Y <- new_X %*% beta_v + rnorm(N)

# Lasso: Vorhersage von new_Y für Test-Datensatz
Y_predict_lars <- predict(
  object = fit_lars,
  s = fraction_opt,
  type = "fit",
  mode = "fraction",
  newx = new_X
)$fit

# Lasso: MSE für Test-Datensatz berechnen
mean((Y_predict_lars - new_Y)^2)
```

```
[1] 1.419817
```

Wir schätzen nun das große Modell mit allen 9 Variablen mit KQ und berechnen ebenfalls den MSE der Prognosen für den Test-Datensatz.

```
# KQ-Schätzung des großen Modells durchführen
KQ_fit <- lm(Y ~ X - 1)

# Test-Datensatz für predict.lm() formatieren
```

[16] Wegen $N = 25$ verbleiben bei der KQ-Schätzung mit 9 Regressoren nur 16 Freiheitsgrade.

```r
new_X <- as.data.frame(new_X)
colnames(new_X) <- paste0("X", 1:9)

# KQ: Vorhersage von new_Y für Test-Datensatz
Y_predict_KQ <- predict(
  object = KQ_fit,
  newdata = new_X
)

# KQ: MSE für Test-Datensatz berechnen
mean((Y_predict_KQ - new_Y)^2)
```

```
[1] 9.851932
```

Offenbar führt die Lasso-Schätzung zu einem deutlich geringeren MSE der Vorhersage von `Y` für den Test-Datensatz als die KQ-Schätzung und damit zu einer höheren Vorhersagegüte. Das "sparsame" mit Lasso-Regression geschätzte Modell ist dem "großen" mit KQ geschätztem Modell in dieser Hinsicht also überlegen.

Key Facts zu Lasso-Regression

- Lasso-Regression bestraft die Verlustfunktion des KQ-Schätzers mit der ℓ_1-Norm der Koeffizienten.
- Neben Koeffizientenschätzung mit Shrinkage in Richtung 0 kann der Lasso-Schätzer Variablenselektion durchführen: Regressionskoeffizienten können exakt mit 0 geschätzt und so ein "sparsames", leichter zu interpretierendes Modell gewählt werden.
- Wie bei Ridge Regression impliziert die Wahl von λ einen Bias-Variance-Tradeoff, der für Vorhersagen nützlich ist: Für größere λ wird mehr Verzerrung induziert und möglicherweise relevante Variablen mit kleinen Koeffizienten aus dem Modell entfernt. Ein solches sparsames Modell kann eine höhere Prognosegüte haben als ein komplexes, unregularisiertes Modell.
- Der Lasso-Schätzer $\widehat{\boldsymbol{\beta}}_\lambda^L$ ist *nicht* erwartungstreu.
- Lasso Regression kann bspw. mit dem LARS-Algorithmus (Paket `lars`) oder mit `glmnet` berechnet werden.

6.3 Vergleich von Lasso- und Ridge-Regression mit Simulation

In diesem Kapitel illustrieren wir Vor- und Nachteile von Lasso- und Ridge-Regression in Prognose-Anwendungen anhand von Monte-Carlo-Simulationen. Wir betrachten hierbei datenerzeugende Prozesse, die sich hinsichtlich der Anzahl relevanter Variablen sowie der Korrelation dieser Variablen unterscheiden.

Die grundlegende Vorschrift für die Simulationen ist

$$Y_i = \sum_{j=1}^{k=40} \beta_j X_{i,j} + u_i, \quad u_i \overset{u.i.v.}{\sim} N(0,1), \quad i = 1, \ldots, 100,$$

wobei die Regressoren X_j eine Varianz von 1 haben und aus einer multivariaten Normalverteilung mit Korrelation

$$\rho \in (0, 0.5, 0.8)$$

gezogen werden.

Für die Koeffizienten $\boldsymbol{\beta}$ unterscheiden wir zwei Szenarien. In Szenario A ist

$$\boldsymbol{\beta} = (1, \ldots, 1)',$$

d.h. alle Variablen sind relevant und haben denselben Einfluss auf Y. In Szenario B erzeugen wir $\boldsymbol{\beta}$ einmalig vorab so, dass

$$\beta_j = \begin{cases} 1, & \text{mit Wsk. } p \\ 0, & \text{mit Wsk. } 1-p, \end{cases}$$

d.h. nur eine Teilmenge der Variablen beeinflusst Y jeweils mit demselben Effekt $\beta_j = 1$. Die übrigen Variablen sind irrelevant.

Wir schätzen und validieren die Modelle mit `glmnet()`.

6.3.1 Prognosegüte in diversen Szenarien

```
# Simulationsparameter definieren
rho <- c(0, 0.5, 0.8)   # Korrelation
k <- 40                 # Anz. Regressoren
```

```
N <- 100                # Anz. Beobachtungen
n_sim <- 100            # Anz. Simulationen
```

Damit der Code für die Simulation möglichst wenig repetitiv ist, definieren wir eine Funktion `cv.glmnet_MSE()`, die unter Angabe der Daten `X` und `Y`, des Trainingssets `train` sowie des Parameters `alpha` den gewünschten regularisierten Schätzer under Verwendung von Cross Validation anpasst und den Testset-MSE zurückgibt.

```
# allg. Funktion für Testset-MSE nach CV
cv.glmnet_MSE <- function(X, Y, train, alpha) {

  # Modell mit glmnet schätzen; lambda per CV bestimmen
  fit_cv <- cv.glmnet(
    x = X[train,],
    y =Y[train],
    alpha = alpha
  )

  # Vorhersagen treffen
  Y_pred <- predict(
    object = fit_cv,
    s = fit_cv$lambda.min,
    newx = X[-train,])

  return(
    # Testset-MSE berechnen
    mean(
      (Y[-train] - Y_pred)^2
      )
  )
}
```

Wir initialisieren zunächst Matrizen, in welche die MSEs aus den 100 Simulationsdurchläufen reihenweise geschrieben werden. `lasso_mse` und `ridge_mse` haben je eine Spalte für jede Korrelation in `rho`

```
# Matrizen für simulierte MSEs initialisieren...
lasso_mse <- matrix(
  data = NA,
  nrow = n_sim,
  ncol = length(rho)
)
ridge_mse <- lasso_mse

# ... und benennen
colnames(lasso_mse) <- paste0("Kor=", rho)
colnames(ridge_mse) <- colnames(lasso_mse)
```

Für die Simulation iterieren wir mit `purrr::walk` über den Vektor `rho` sowie über die Laufvariable `1:n_sim`. Beide Schleifen nutzen den Syntax für anonyme Funktionen:

```
# Die anonyme Funktion
function(x) return(x)
# ist äquivalent definiert als
\(x) return(x)
```

In jeden Simulationsdurchlauf erzeugen wir den Datensatz entsprechend der obigen Vorschrift, teilen die Daten auf und berechnen MSEs für Lasso- und Ridge-Regression mit `cv.glmnet_MSE()`.

Szenario A

```
# Koeffizienten-Vektor definieren
beta <- rep(1, k)
```

```
library(mvtnorm)
library(tidyverse)

set.seed(1234)

# Simulation durchführen
walk(1:length(rho), \(j) {
```

```r
# Korrelationsmatrix definieren
Sigma <- matrix(
  data = rho[j],
  nrow = k,
  ncol = k
)
diag(Sigma) <- 1

walk(1:n_sim, \(i) {

# Daten simulieren
X <- rmvnorm(
  n = N,
  mean = rep(0, k),
  sigma = Sigma
)
Y <- X %*% beta + rnorm(N)

# Trainingsdaten definieren
ID_train <- sample(
  x = c(1:N),
  size = N/2
)

# Modelle mit CV schätzen und MSEs berechnen
# Ridge-Regression
ridge_mse[i, j] <<- cv.glmnet_MSE(
  X = X,
  Y = Y,
  train = ID_train,
  alpha = 0
)

# Lasso-Regression
lasso_mse[i, j] <<- cv.glmnet_MSE(
  X = X,
  Y = Y,
  train = ID_train,
```

```
    alpha = 1
  )

  })

})
```

Beachte, dass hier der Super-Assignment-Operator `<<-` genutzt wird, damit `walk` die Matrizen `ridge_mse` und `lasso_mse` in der globalen Umgebung überschreibt.[17]

Wir berechnen jeweils den mittleren MSEs, sammeln die Ergebnisse in einer `tibble()` und nutzen `gt()` für die tabellarische Darstellung.

```
library(gt)

# Ergebnisse tabellarisch darstellen
tibble(
  Methode = c(
    "Lasso-Regression",
    "Ridge-Regression"
  ),
) %>%
  bind_cols(
    bind_rows(
      colMeans(lasso_mse),
      colMeans(ridge_mse)
    )
  ) %>%
  gt() %>%
  tabopts
```

Tabelle 6.1: Durchschnittliche Testset-MSEs für Setting A

Methode	Kor=0	Kor=0.5	Kor=0.8
Lasso-Regression	7.17	10.398	7.581
Ridge-Regression	4.841	1.615	1.517

[17] Dies folgt aus der Definition von `walk`. `<-` bewirkt hier lediglich Assignment in der Funktionsumgebung.

Tabelle 6.1 zeigt, dass Ridge-Regression gegenüber Lasso-Regression für jede der drei betrachteten Korrelationen überlegen ist. Insbesondere bei stärker korrelierten Regressoren ist Ridge vorteilhaft.

Für Szenario B überschreiben wir `beta` nach Multiplikation mit einem zufälligen binären Vektor, sodass einige der Koeffizienten 0 und die zugehörigen Variablen irrelevant für Y sind.

Szenario B

```
# Wsk. für Relevanz einer Variable
p <- .3

# Koeffizienten-Vektor definieren
set.seed(123)
beta <- beta * sample(
  x = 0:1,
  size = k,
  replace = T,
  prob = c(1-p, p)
)

# Koeffizienten prüfen
head(beta, n = 10)
```

```
[1] 0 1 0 1 1 0 0 1 0 0
```

Eine wiederholung der Simulation für die modifizierten Koeffizienten `beta` und liefert folgende tabellarische Auswertung.

Tabelle 6.2: Durchschnittliche Testset-MSEs für Szenario B

Methode	Kor=0	Kor=0.5	Kor=0.8
Lasso	2.51	2.143	1.923
Ridge	3.331	2.562	2.014

Die Ergebnisse in Tabelle 6.2 zeigen, dass Ridge-Regression in Szenario B bis auf den Fall unkorrelierter Regressoren etwas schlechter abschneidet als in Szenario A. Die hohe Anzahl irrelevanter Variablen verbessert die Leistung von Lasso deutlich: Hier ist es plausibel, dass Lasso aufgrund der Thresholding-Eigenschaft die Koeffizienten einiger irrelevanten Variablen häufig exakt 0 setzt und damit ein sparsameres Modell schätzt als Ridge. Entsprechend erzielt

Lasso in diesem Szenario insbesondere für $\rho = 0$ genauere Vorhersagen als Ridge Regression.

6.3.2 Visualisierung des Bias-Variance-Tradeoffs bei Prognosen

Für ein besseres Verständnis, wie sich der Regularisierungsparameter λ auf den Bias-Variance-Tradeoff bei Prognosen mit Ridge- und Lasso-Regression auswirkt, vergleichen wir für beide Methoden nachfolgend die Abhängigkeit des MSEs der Prognose $\widehat{Y}_0$ für den Wert Y_0 der abhängigen Variable eines Datenpunkts anhand seiner Regressoren $\boldsymbol{X}_0'$, wobei

$$\text{MSE}(\widehat{Y}_0) = \text{Bias}(\widehat{Y}_0)^2 + \text{Var}(\widehat{Y}_0) + \text{Var}(Y_0) \tag{6.20}$$

Beachte, dass $\text{Var}(Y_0)$ die durch den datenerzeugenden Prozess (und damit unvermeidbare) Varianz von Y_0 ist, wohingegen $\text{Bias}(\widehat{Y}_0)^2$ und $\text{Var}(\widehat{Y}_0)$ von dem verwendeten Schätzer für $\widehat{Y}_0$ abhängt.

Für die Simulation betrachten wir erneut Szenario A aus Kapitel 6.3.1 mit 50 Beobachtungen für ein Modell mit 40 unkorrelierten Regressoren. Wir legen zunächst die Simulationsparameter fest und erzeugen den vorherzusagenden Datenpunkt (`X_0`, `Y_0`).

```r
# Parameter festlegen
set.seed(1234)
n <- 200 # Anz. Iterationen
N <- 50  # Anz. Beobachtungen
k <- 40  # Anz. Variablen

# Korrelationsmatrix definieren
Sigma <- diag(k) # Diagonalmatrix
beta <- rep(x = 1, k)

# Prognose-Ziel vorab zufällig generieren:

# Regressoren
X_0 <- rmvnorm(
  n = 1,
  mean = rep(x = 0, k)
)
```

```
# Abh. Variable
Y_0 <- X_0 %*% beta + rnorm(n = 1) %>%
  as.vector()
```

Anhand der Simulationsergebnisse wollen wir die von der verwendeten Schätzfunktion abhängigen Komponenten von (6.20) untersuchen. Wir initialisieren hierzu die Listen `ridge_fits` und `lasso_fits`, in die unsere Simulationsergebnisse geschrieben werden.

```
# Listen für Simulationsergebnisse initialisieren
ridge_fits <- list()
lasso_fits <- list()
```

Weiterhin definieren wir separate λ-Sequenzen für Lasso- und Ridge-Schätzer.[18]

```
# Lambda-Sequenzen festlegen
lambdas_r <- seq(.25, 2.5, length.out = 100)
lambdas_l <- seq(.05, 0.5, length.out = 100)
```

Für die Simulation iterieren wir mit `walk()` über simulierte Datensätze und schreiben jeweils den vollständigen Output von `glmnet()` in die zuvor definierten Listen `ridge_fits` und `lasso_fits`.

```
# Simulation
walk(1:n, \(i) {

  # Daten simulieren
  X <- rmvnorm(
    n = N,
    mean = rep(0, k),
    sigma = Sigma
  )
  Y <- X %*% beta + rnorm(n = N, sd = 5)

  # Modelle mit glmnet schätzen
```

[18] Die Sequenzen haben wir in Abhängigkeit des DGP so gewählt, dass die Abhängigkeit der Prognosegüte von λ gut visualisiert werden kann.

```
  # Ridge-Regression
  ridge_fits[[i]] <<- glmnet(
    x = X,
    y = Y,
    alpha = 0,
    intercept = F
  )
  # Lasso-Regression
  lasso_fits[[i]] <<- glmnet(
    x = X,
    y = Y,
    alpha = 1,
    intercept = F
  )

})
```

Wir nutzen Funktionen aus `purrr` und `dplyr`, um über die in den Simulationsdurchläufen angepassten Modelle zu iterieren. Mit `predict()` erhalten wir Punktvorhersagen für `Y_0` für jedes λ der zuvor definierten λ-Sequenzen. Beachte, dass `map()` jeweils eine Liste mit 200 Punktvorhersagen für jedes der 100 zurückgibt. Mit `list_rbind()` können wir die Ergebnisse komfortabel jeweils in einer `tibble` sammeln.

```
# Prognosen für Ridge-Regression
pred_r <- map(
  .x = ridge_fits,
  .f = ~ as_tibble(
    predict(
      object = .,
      s = lambdas_r,
      newx = X_0
    )
  )
) %>%
  list_rbind()

# Prognosen für Lasso-Regression
```

```
pred_l <- map(
  .x = lasso_fits,
  .f = ~ as_tibble(
    predict(
      object = .,
      s = lambdas_l,
      newx = X_0)
    )
) %>%
  list_rbind()
```

Für die statistische Auswertung berechnen wir jeweils $\text{MSE}(\widehat{Y}_0)$, $\text{Bias}(\widehat{Y}_0)^2$ und $\text{Var}(\widehat{Y}_0)$ und führen die Ergebnisse mit `pivot_longer()` in ein langes Format `sim_data_r` über. Wir berechnen weiterhin mit `MSE_min_r` das λ, für das wir über die Simulationsdurchläufe durchschnittlich den geringsten MSE beobachten.

Ridge-Regression

```
# Ergebnisse für Ridge-Regression zusammenfassen
sim_data_r <- tibble(

  lambda = lambdas_r,

  "MSE" = map_dbl(
    .x = pred_r,
    .f = ~ mean((.x - Y_0)^2)
  ),

  "Bias^2" = map_dbl(
    .x = pred_r,
    .f = ~ (mean(.x) - Y_0)^2
  ),

  "Varianz" = map_dbl(
    .x = pred_r,
    .f = ~ var(.x)
  )
) %>%
```

```
  pivot_longer(
    cols = -lambda,
    values_to = "Wert",
    names_to = "Statistik"
  )

# Lambda bei MSE-Minimum bestimmen
MSE_min_r <- sim_data_r %>%
  filter(
    Statistik == "MSE",
    Wert == min(Wert)
  )
```

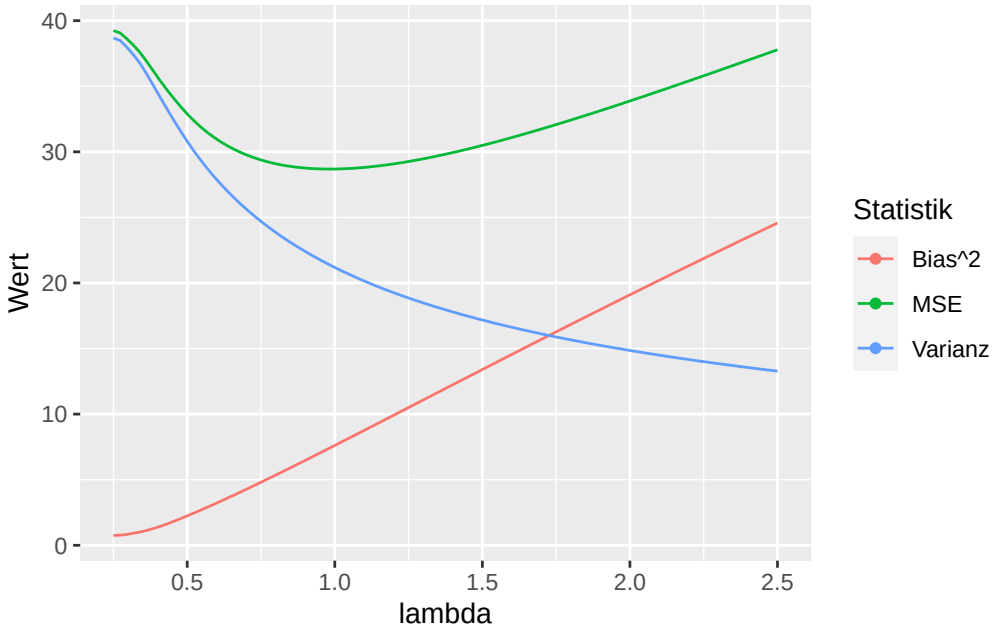
Lasso-Regression

```
# Ergebnisse zusammenfassen
sim_data_l <- tibble(

  lambda = lambdas_l,

  "MSE" = map_dbl(
    .x = pred_l,
    .f = ~ mean((. - Y_0)^2)
  ),

  "Bias^2" = map_dbl(
    .x = pred_l,
    .f = ~ (mean(.) - Y_0)^2
  ),

  "Varianz" = map_dbl(
    .x = pred_l,
    .f = ~ var(.)
  )
) %>%
  pivot_longer(
    cols = -lambda,
    values_to = "Wert",
```

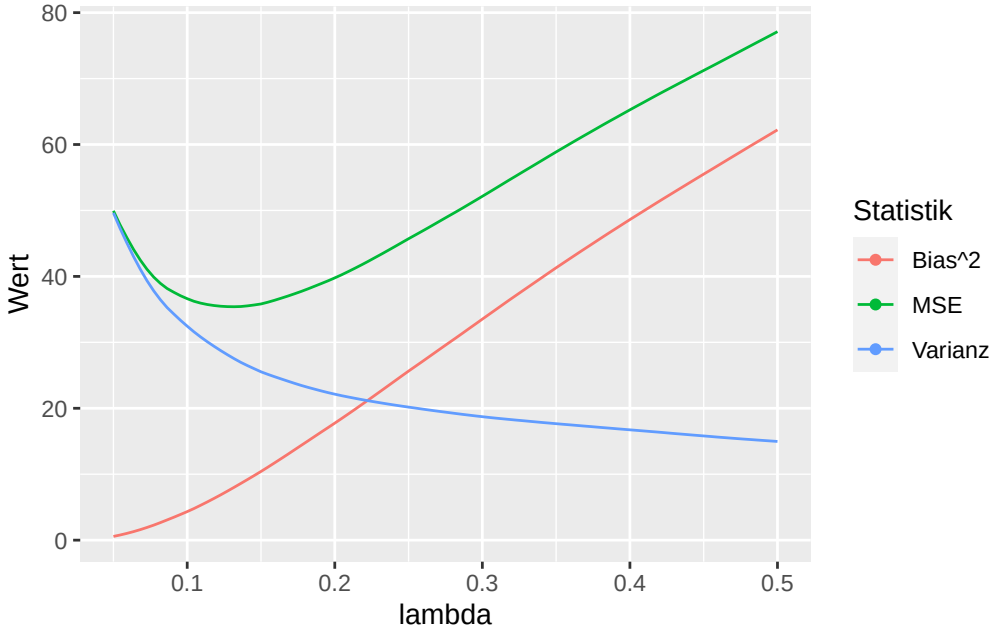
```
    names_to = "Statistik"
  )

# Lambda bei MSE-Minimum bestimmen
MSE_min_l <- sim_data_l %>%
  filter(
    Statistik == "MSE",
    Wert == min(Wert)
  )
```

Die Datensätze im langen Format, `sim_data_r` und `sim_data_l`, werden nun für die Visualisierung der Ergebnisse mit `ggplo2` genutzt.

```
# MSE, Bias^2 und Varianz gegen Lambda plotten

# Ridge-Regression
sim_data_r %>%
  ggplot(
    mapping = aes(
      x = lambda,
      y = Wert,
      color = Statistik
    )
  ) +
  geom_line() +
  geom_point(data = MSE_min_r)

# Lasso-Regression
sim_data_l %>%
  ggplot(
    mapping = aes(
      x = lambda,
      y = Wert,
      color = Statistik
    )
  ) +
  geom_line() +
  geom_point(data = MSE_min_l)
```



(a) Ridge Regression



(b) Lasso Regression

Abbildung 6.9: Simulierte MSE-Komponenten in Abhängigkeit von Lambda

Anhand von Abbildung 6.9 lässt sich der Bias-Variance-Tradeoff bei der Vorhersage von Y_0 gut erkennen: Bereits für kleine λ erzielen beide Methode eine deutliche Reduktion des MSE. Dies wir durch etwas zusätzlichen Bias, aber eine überproportionale Verringerung der Varianz erreicht. Der erkennbare funktionale Zusammenhang zeigt, dass der MSE eine konvexe Funktion von λ ist. Damit existieren optimale λ mit minimalem MSE (grüne Punkte), die wir mit Cross Validation schätzen können.

6.4 Inferenz für Treatment-Effekt-Schätzung mit vielen Variablen

In empirischen Studien des Effekts einer Behandlungsvariable B auf eine Outcome-Variable Y steht häufig eine Vielzahl potentieller Kontrollvariablen zur Verfügung. Häufig ist unklar, welche Variablen in das Modell aufgenommen werden sollten, um das Risiko einer verzerrten Schätzung durch ausgelassene Variablen zu vermindern und gleichzeitig eine Schätzung mit geringer Varianz zu gewährleisten. Ist der Beobachtungsumfang N relativ zur Variablenanzahl k groß, so kann die KQ-Schätzung einer langen Regression (ein Modell mit allen k Kontrollvariablen) gute Ergebnisse liefern. In der Praxis liegt diese wünschenswerte Situation jedoch oft nicht vor und es ist $k \lesssim N$ oder sogar $k > N$. Dann ist eine KQ-Schätzung des Behandlungseffekts anhand aller k Variablen mit hoher Varianz behaftet bzw. gar nicht möglich.[19] Ein weiteres Szenario ist $k(N) > N$, d.h. die Anzahl der Regressoren kann mit dem Beobachtungsumfang wachsen.[20] Lasso-Verfahren können dann hilfreich sein, um Determinanten von Y *und* B zu identifizieren und damit eine Menge an Kontrollvariablen zu selektieren, für die eine erwartungstreue und konsistente Schätzung des interessierenden Effekts wahrscheinlich ist.

Betrachte zunächst das Modell mit allen Kontrollvariablen X_j,

$$Y_i = \beta_0 + \alpha_0 B_i + \sum_{j=1}^{k} \beta_j X_{i,j} + u_i, \tag{6.21}$$

wobei einige $\beta_j = 0$ sind und wir annehmen, dass B lediglich mit ein paar der X_j korrelliert. Die Shrinkage der geschätzten Koeffizienten aus einer naiven Lasso-Regression von (6.21) führt grundsätzlich zu einer verzerrten Schätzung

[19] Beachte, dass der KQ-Schätzer bei $k > N$ nicht lösbar ist.

[20] Dieses Szenario wird unter Bedingungen bzgl. der Wachstumsrate und der Größe der Koeffizienten betrachet, s. (Belloni und Chernozhukov 2013).

des Behandlungseffekts α_0 und damit zu ungültiger Inferenz.[21]

Die Verzerrung von geschätzten Koeffizienten kann vermieden werden, indem Lasso lediglich zur Selektion von Kontrollvariablen verwendet wird. Dabei wird mit einer Lasso-Regression von Y auf die X_j eine Teilmenge von Regressoren $\mathcal{S}$ selektiert und der Treatment-Effekt anschließend mit der KQ-Schätzung von

$$Y_i = \beta_0 + \alpha_0 B_i + \sum_{j \in \mathcal{S}} \beta_j X_{i,j} + e_i, \tag{6.22}$$

basierend auf der Selektion $\mathcal{S}$ berechnet wird.[22] Ein solcher *Post-Lasso-Selection-Schätzer* (Belloni und Chernozhukov 2013) ist jedoch im Allgemeinen und insbesondere in hoch-dimensionalen Settings nicht konsistent für α_0 und nicht asymptotisch normalverteilt, da weiterhin die Gefahr einer verzerrten Schätzung durch in $\mathcal{S}$ ausgelassene Variablen besteht, die mit B korrelieren: Lasso selektiert Variablen X_j, die "gut" Y erklären. Dabei kann nicht ausgeschlossen werden, das ein Modell gewählt wird, dass relevante Determinanten von B auslässt. Selbst wenn wir ein mit Lasso gewähltes Modell mit KQ (d.h. ohne Shrinkage) schätzen, würde α_0 verzerrt geschätzt!

Belloni, Chernozhukov, und Hansen (2014) schlagen ein alternatives Verfahren vor, dass auf Selektion der Determinanten X_j von Y und B basiert. Dieses Verfahren wird als *Post-Double Selection* bezeichnet und kann wiefolgt implementiert werden:

Post-Double-Selection-Schätzer

1. Bestimme die Determinanten X_j von Y mit Lasso-Regression und bezeichne die Menge der selektierten Variablen als $\mathcal{S}_Y$.
2. Bestimme die Determinanten X_j von B mit Lasso-Regression und bezeichne die Menge der selektierten Variablen als $\mathcal{S}_B$.
3. Bestimme die Schnittmenge $\mathcal{S}_{YB} = \mathcal{S}_Y \cap \mathcal{S}_B$. Schätze den Treatment-Effekt als $\widehat{\alpha}_0$ in der KQ-Regression

$$Y_i = \beta_0 + \alpha_0 B_i + \sum_{j \in \mathcal{S}_{YB}} \beta_j X_{i,j} + v_i. \tag{6.23}$$

Belloni, Chernozhukov, und Hansen (2014) zeigen, dass $\widehat{\alpha}_0$ aus diesem Verfahren ein asymptotisch normalverteiler Schätzer für α_0 ist und herkömmliche t-Tests und Konfidenzintervalle gültige Inferenz erlauben.

[21] Hahn u. a. (2018) geben eine ausführliche Erläuterung dieser Problematik.
[22] Solche Verfahren werden *Post-Selection-Schätzer* genannt.

Wir illustrieren die in diesem Abschnitt betrachteten Schätzer nun anhand simulierter Daten mit R. Die fiktive Problemstellung ist die Schätzung eines wahren Treatment-Effekts $\alpha_0 = 2$, wenn so viele potenzielle Kontrollvariablen vorliegen, dass der KQ-Schätzer gerade noch berechnet werden kann, aber aufgrund hoher Varianz unzuverlässig ist. Hierzu erzeugen wir Y gemäß der Vorschrift

$$Y_i = \alpha_0 B_i + \sum_{j=1}^{k_Y} \beta_j^Y X_{i,j}^Y + \sum_{l=1}^{k_{YB}} \beta_l^{YB} X_{i,l}^{YB} + u_i,$$

$$\beta_j^{YB} \overset{u.i.v}{\sim} N(10,1), \quad \beta_j^Y \overset{u.i.v}{\sim} U(0,1), \quad u_i \overset{u.i.v}{\sim} N(0,1).$$

$$i = 1, \ldots, 550$$

Die Behandlungsvariable B_i entspricht der Vorschrift

$$B_i = \sum_{l=1}^{k_{YB}} \beta_l^{YB} X_{i,l}^{YB} + e_i,$$

$$\beta_j^{YB} \overset{u.i.v}{\sim} N(2, 0.2), \quad e_i \overset{u.i.v}{\sim} N(0,1).$$

Wir wählen $k_{YB} = k_Y = 25$. Zusätzlich zu B, den Determinanten von Y *und* B (X^{YB}) sowie den Variablen, die ausschließlich Y beeinflussen (X^Y) gibt es $k_U = 499$ Variablen X^U, die weder Y noch B beeinflussen und damit irrelevant für die Schätzung des Behandlungseffekts sind. Wir haben also $N = 550$ Beobachtungen und insgesamt $k = 1 + k_Y + k_{YB} + k_U = 550$ potenzielle Kontrollvariablen von denen $k_{YB} = 25$ für eine unverzerrte Schätzung von α_0 relevant sind.

Der nachstehende Code generiert die Daten gemäß der Vorschrift.

```
library(mvtnorm)
library(tidyverse)
set.seed(4321)

n <- 550      # Beobachtungen
p_Y <- 25     # Determinanten Y
p_B <- 25     # Determinanten B *und* Y
```

```
p_U <- 499    # irrelevante Variablen

# Variablen generieren
XB <- rmvnorm(n = n, sigma = diag(p_B))
XU <- rmvnorm(n = n, sigma = diag(p_U))
XY <- rmvnorm(n = n, sigma = diag(p_Y))

# Stetige Behandlungsvariable erzeugen
B <- XB %*% rnorm(p_B, 2, sd = .2) + rnorm(n)

# Abh. Variable erzeugen, Behandlungseffekt (ATE) ist 2
Y <- 2 * B +
  XB %*% rnorm(p_B, mean = 10) +
  XY %*% runif(p_Y) +
  rnorm(n)

# Variablen in tibble sammeln
X <- cbind(B, XB, XU, XY) %>%
  as_tibble()

# Namen zuweisen
colnames(X) <- c(
  "B",
  paste0("XB", 1:p_B),
  paste0("XU", 1:p_U),
  paste0("XY", 1:p_Y)
)
```

Wünschenswert wäre die KQ-Schätzung des wahren Modells. Diese ergibt eine Schätzung nahe des wahren Treatment-Effekts $\alpha_0 = 2$. Unter realen Bedingungen wäre diese Regression jedoch nicht implementierbar, weil die relevanten Kovariablen `XB` unbekannt sind.

```
# KQ: Wahres Modell schätzen
lm(Y ~ B + XB - 1)$coefficients["B"]
```

```
       B
1.937031
```

Wir schätzen daher zunächst die "lange" Regression mit allen k verfügbaren Variablen mit KQ. Beachte, dass der KQ-Schätzer für α_0 zwar implementierbar und erwartungstreu ist, jedoch eine hohe Varianz aufweist. Wegen $k = N = 550$ erhalten wir eine perfekte Anpassung an die Daten und können mangels Freiheitsgraden keine Hypothesentests durchführen.

```
# KQ: Lange Regression schätzen
lm(Y ~ . - 1, data = X)$coefficients["B"]
```

```
       B
3.079497
```

Die KQ-Schätzung von α_0 anhand der langen Regression weicht deutlich vom wahren Wert $\alpha_0 = 2$ ab.

Eine "kurze" KQ-Regression nur mit der Behandlungsvariable B führt wegen Korrelation mit den ausgelassenen Determinanten in `XB` zu einer deutlich verzerrten Schätzung.

```
# KQ: Kurze Regression
lm(Y ~ B - 1)$coefficients["B"]
```

```
       B
6.716837
```

Die Methoden von Belloni und Chernozhukov (2013) und Belloni, Chernozhukov, und Hansen (2014) sind im R-Paket `hdm` implementiert. Mit den Funktionen `hrm::rlasso()` und `hdm::rlassoEffect` kann Lasso-Regression sowie Post- und Double-Post-Selection durchgeführt werden.[23]

Wir berechnen zunächst den naiven Lasso-Schätzer in einem Modell mit allen Variablen.

```
library(hdm)

# Naiver Post-Lasso-Schätzer
lasso <- rlasso(
  x = X,
  y = Y,
```

[23] Diese Funktionen ermitteln ein optimales λ mit dem in Belloni u. a. (2012) vorgeschlagenen Algorithmus.

```
  intercept = F,
  post = F
)

# Koeffizientenschätzer auslesen
lasso$coefficients["B"]
```

```
       B
6.368456
```

Auch dieser Schätzer ist deutlich verzerrt. Problematisch ist hier nicht nur die Shrinkage auf $\widehat{\alpha}_0$, sondern die Selektion der Variablen in `XB`:

```
# Welche Variablen in XB selektiert Lasso *nicht*?
nselektiert <- which(lasso$coef[1:26] == 0)   # ID

# Namen auslesen
names(lasso$coef[1:26])[nselektiert]
```

```
[1] "XB8"  "XB10" "XB16" "XB18" "XB20"
```

Durch das Auslassen dieser Determinanten von Y und B leidet der Lasso-Schätzer unter OVB.

Als nächstes berechnen wir den Post-Lasso-Selection-Schätzer.

```
# Post-Lasso-Selection-Schätzer berechnen
p_lasso <- rlasso(
  x = X,
  y = Y,
  intercept = F,
  post = T
)

# Schätzung für alpha_0
p_lasso$coef["B"]
```

```
       B
6.362409
```

Die Ähnlichkeit der Post-Lasso-Schätzung von α_0 zur Lasso-Schätzung zeigt deutlich, dass die Verzerrung des Lasso-Schätzers überwiegend durch ausgelassene Variablen anstatt durch Shrinkage verursacht wird.

Mit `rlassoEffect()` können wir den Post-Double-Selection-Schätzer berechnen.

```
# Post-Double-Selection-Schätzer
pds_lasso <- rlassoEffect(
  x = X %>%
    dplyr::select(-B) %>%
    as.matrix(),
  y = Y,
  d = B,
  method = "double selection"
)

# Schnittmenge der selektierten Determinanten
# von Y und B
(
  S_BY <- names(
    which(pds_lasso$selection.index)
  )
)
```

```
 [1] "XB1"   "XB2"   "XB3"   "XB4"   "XB5"   "XB6"   "XB7"
 "XB8"   "XB9"
[10] "XB10"  "XB11"  "XB12"  "XB13"  "XB14"  "XB15"  "XB16"
"XB17"  "XB18"
[19] "XB19"  "XB20"  "XB21"  "XB22"  "XB23"  "XB24"  "XB25"
"XU209" "XU241"
[28] "XU295" "XY3"   "XY7"   "XY8"   "XY12"  "XY13"  "XY15"
"XY16"  "XY19"
[37] "XY23"
```

Double Selection führt ebenfalls zu einem Post-Lasso-KQ-Schätzer mit allen 25 relevaten Variablen in `XB`. Wir selektieren allerdings deutlich weniger irrelevante Variablen aus `XU` als mit Single Selection und dennoch einige Determinanten von Y aus `XY`. Double Selection führt also zu einer unverzerrten Schätzen mit geringerer Varianz. Mit `summary()` erhalten wir gültige Inferenz bzgl. des Treatment-Effekts.

```r
summary(pds_lasso)
```

```
[1] "Estimates and significance testing of the effect of target
variables"
   Estimate. Std. Error t value Pr(>|t|)
d1   1.94977    0.07127   27.36   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Der Post-Double-Selection-Schätzer liefert unter den betrachteten Verfahren die beste Schätzung von α_0 und erlaubt gülstige statistische Inferenz. Der geschätzte Effekt ist hoch-signifikant.

Key Facts zum Post-Double-Selection-Schätzer

- Durch die sorgfältige Auswahl von Variablen, die mit Behandlung- und Outcome-Variable zusammenhängen, ermöglicht die Double-Selection eine bessere Kontrolle über das Risiko ausgelassender Variablen in Beobachtungsstudien und ermöglicht gültige (asymptotisch normale) Inferenz.
- Der Post-Double-Selection-Schätzer besteht aus drei Regressionen:
  1. Es werden Variablen mit Lasso selektiert, welche die *Behandlungs-Variable* erklären.
  2. Es werden Variablen mit Lasso selektiert, welche die *Outcome-Variable* erklären.
  3. Der Post-Double-Selection-Schätzer ist der KQ-Schätzer in einer Regression, die für die Schnittmenge der ausgewählten Variablen kontrolliert.
- Dank der Selektion mit Lasso kann der Schätzer auch bei hochdimensionalen Daten ($k > n$) angewendet werden.
- Post-Double-Selection-Schätzer für Behandlungseffekte sind im R-Paket `hdm` implementiert.

6.4.1 Case Study: Makroökonomisches Wachstum

Zur Illustration des Post-Double-Selection Schätzers betrachten wir eine empirische Anwendung bzgl. der Validierung von makroökonomischer Wachstumtheorie. Aus neo-klassischen Ansätzen wie dem Solow-Swan-Modell kann die Hy-

pothese, dass Volkswirtschaften zu einem gemeinsamen Wachstumspfad hin konvergieren, abgeleitet werden. Diese Konvergenzhypothese impliziert die Existenz von Aufholeffekten: Ärmere Volkswirtschaften müssen im mittel schneller Wachsen als die Wirschaft wohlhabender Länder. Die grundlegende Spezifikation eines entsprechenden Regressionsmodells lautet

$$\text{WR}_i = \alpha_0 \text{BIP0}_i + u_i, \tag{6.24}$$

wobei WR_i die Wachstumsrate des Pro-Kopf-BIP in Land i über einen Zeitraum (typischerweise berechnet als Log-Differenz zwischen zwei Perioden) und BIP0_i das (logarithmierte) Pro-Kopf-BIP zu beginn der Referenzperiode ist. Gemäß der Konvergenzhypothese muss $\alpha_0 < 0$ sein: Je wohlhabender eine Volkswirtschaft ist, desto geringer ist das Wirtschaftswachstum.

Um Verzerrung durch ausgelassene Kovariablen zu vermeiden, sollte das Modell (6.24) um länder-spezifische Regressoren $x_{i,j}$, die sowohl das Ausgagnsniveau BIP0 sowie die Wachtumsrate beinflussen, erweitert werden. Zu der großen Menge potentieller Kovariablen gehören makro- und sozio-ökonomische Maße wie bspw. die Investitionstätigkeit des Staates, Offenheit der Volkswirtschaft, das politische Umfeld, das Bildungsniveau, die Demographie usw. Eine bevorzugte Spezifikation ist daher

$$\text{WR}_i = \alpha_0 \text{BIP0}_i + \sum_{j=1}^{k} \beta_j x_{i,j} + u_i, \tag{6.25}$$

wobei α_0 als Behandlungseffekt interpretiert werden kann. Beachte, dass (6.25) eine Regression in der Form von (6.21) ist.

Wir illustrieren die Schätzung von und Inferenz bzgl. α_0 in (6.25) mit Post-Double-Selektion für einen 90 Länder umfassenden Auszug aus dem Datensatz von Barro und Lee (2013), der als Objekt `GrowthData` im R-Paket `hdm` verfügbar ist.[24]

```
# Datensatz in Arbeitsumgebung verfügbar machen
library(hdm)
data(GrowthData)

# Anzahl Beobachtungen und Variablen
dim(GrowthData)
```

```
[1] 90 63
```

[24] Eine ausführliche Beschreibung der Variablen ist hier einsehbar.

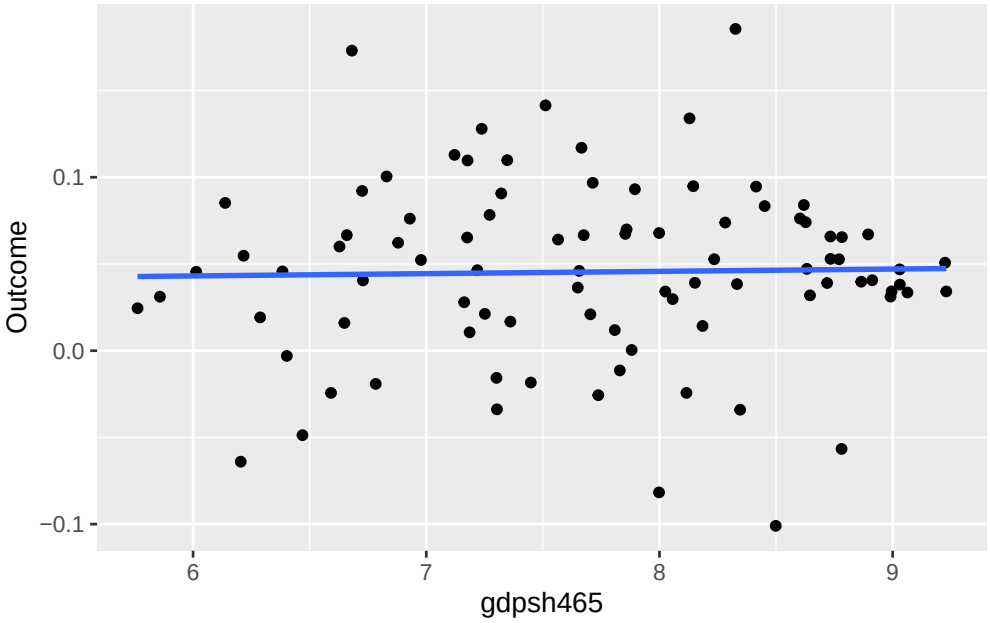


Abbildung 6.10: BIP-Wachstum: Einfache Regression

Die Spalte `Outcome` ist die jeweilige Wachstumsrate des BIP zwischen den Perioden 1965-1975 und 1975-1985 und `gdpsh465` ist das reale Pro-Kopf-BIP im Jahr 1965 zu Preisen von 1980.

Wir führen zunächst eine graphische Analyse hinsichtlich des Modells einfachen Modells (6.24) durch, indem wir `gdpsh465` gegen `Outcome` plotten und die geschätzte Regressionsgerade einzeichnen.

```
# Einfache grafische Analyse mit ggplot2
GrowthData %>%
  ggplot(
    mapping = aes(
      x = gdpsh465,
      y = Outcome
    )
  ) +
  geom_point() +
  geom_smooth(method = "lm", se = F)
```

Abbildung 6.10 zeigt einen geringen positiven geschätzten Effekt $\widehat{\alpha}_0$. Eine Auswertung mit `lm()` ergibt, dass der Effekt α_0 nicht signifikant von 0 verschieden ist.

```
# Einfache Regression durchführen,
# Inferenz für gdpsh465 erhalten
lm(Outcome ~ gdpsh465, data = GrowthData) %>%
  summary() %>%
  coefficients() %>%
  .[2, ]
```

```
   Estimate  Std. Error     t value    Pr(>|t|)
0.001316713 0.006102200 0.215776701 0.829661165
```

Der positive Effekt aus der einfachen Schätzung widerspricht der Konvergenzhypothese. Dieses Ergebnis könnte allerdings durch Auslassen relevanter Kovariablen ungültig sein. Beispielsweise ist es plausibel, dass das Bildungsniveau einer Volkswirtschaft sowohl mit dem BIP korreliert ist als auch die Wachstumsrate beeinflusst. Dann wäre das Bildungsniveau eine relevante Kovariable, deren Auslassen zu einer verzerrten Schätzung von α_0 führt.

Eine "lange" Regression mit allen Kovariablen ist zwar möglich, aber problematisch: Das Verhältnis von Beobachtungen (90) zu Regressoren (62) bedeutet eine hohe Unsicherheit der Schätzung.

```
# Inferenz für alpha_0 in langer Regression
summary(
  lm(Outcome ~ . - 1 , data = GrowthData)
  ) %>%
  coefficients() %>%
  .[2, ]
```

```
    Estimate   Std. Error      t value     Pr(>|t|)
-0.009377989  0.029887726 -0.313773911  0.756018518
```

Der geschätzte Koeffizient $\widehat{\alpha}_0$ ist nun zwar negativ, liefert jedoch weiterhin keine Evidenz, dass α_0 von 0 verschieden ist. Ein Vergleich der Standardfehler zeigt aber, dass die KQ-Schätzung aufgrund Berücksichtigung aller potentiellen Kovariablen mit deutlich größerer Varianz behaftet ist als in der einfachen KQ-Regression (6.24)

Post-Double-Selection erlaubt gültige Inferenz bzgl. α_0 nach Schätzung der Menge relevanter Kovariablen. Wir weisen die entsprechenden Variablen R-Objekten zu und berechnen den Schätzer.

```r
# Variablen für Post-Double-Selection vorbereiten

# abh. Variable
y <- GrowthData %>%
  pull(Outcome)

# "Treatment"
d <- GrowthData %>%
  pull(gdpsh465)

# potentielle Regressoren
X <- GrowthData %>%
  dplyr::select(
    -Outcome, -intercept, -gdpsh465
  )
```

```r
# Post-Double-Selection-Schätzer berechnen
Growth_DS <-
  rlassoEffect(
    x = X %>%
      as.matrix(),
    y = y,
    d = d,
    method = "double selection"
)
```

Post-Double-Selection wählt aus der Menge potentieller Kovariablen lediglich sieben Regressoren aus.

```r
# Selektierte Variablen einsehen
# ID
Selektion <- Growth_DS$selection.index

# Namen auslesen
names(
  which(Selektion == T)
)
```

```
[1] "bmp1l"    "freetar"  "hm65"     "sf65"     "lifee065"
"humanf65" "pop6565"
```

Tabelle 6.3 zeigt die Definitionen der ausgewählten Variablen.

Tabelle 6.3: Mit PDS selektierte Variablen aus `GrowthData`. Referenzjahr 1965.

Variable	Beschreibung
bmp1l	Schwarzmarktprämie d. Währung
freetar	Maß für Zollbeschränkungen
hm65	Einschreibungsquote Uni (Männer)
sf65	Beschulungsquote Sekundarstufe (Frauen)
lifee065	Lebenserwartung bei Geburt
humanf65	Durschn. Bildung im Alter 25 (Frauen)
pop6565	Anteil Bevölkerung ü. 65 Jahre

```r
# Gültige Inferenz mit dem Post-Double-Selection-Schätzer
summary(Growth_DS)
```

```
[1] "Estimates and significance testing of the effect of target
variables"
   Estimate. Std. Error t value Pr(>|t|)
d1  -0.05001    0.01579  -3.167  0.00154 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Das Ergebnis der Post-Double-Selection-Schätzung unterstützt die (bedingte) Konvergenzhypothese mit einer signifikanten negativen Schätzung $\widehat{\alpha}_0 \approx -0.05$.

Literatur

Austin, P. 2011. „An Introduction to Propensity Score Methods for Reducing the Effects of Confounding in Observational Studies“. *Multivariate Behavioral Research* 46 (3): 399–424. https://doi.org/10.1080/00273171.2011.568786.

Barro, Robert J., und Jong Wha Lee. 2013. „A new data set of educational attainment in the world, 1950–2010“. *Journal of Development Economics* 104: 184–98. https://doi.org/https://doi.org/10.1016/j.jdeveco.2012.10.001.

Basten, Christoph, und Frank Betz. 2013. „Beyond work ethic: Religion, individual, and political preferences“. *American Economic Journal: Economic Policy* 5 (3): 67–91.

Belloni, Alexandre, Daniel Chen, Victor Chernozhukov, und Christian Hansen. 2012. „Sparse models and methods for optimal instruments with an application to eminent domain“. *Econometrica* 80 (6): 2369–429.

Belloni, Alexandre, und Victor Chernozhukov. 2013. „Least squares after model selection in high-dimensional sparse models“. *Bernoulli*, 521–47.

Belloni, Alexandre, Victor Chernozhukov, und Christian Hansen. 2014. „High-dimensional methods and inference on structural and treatment effects“. *Journal of Economic Perspectives* 28 (2): 29–50.

Cattaneo, Matias D, Michael Jansson, und Xinwei Ma. 2020. „Simple local polynomial density estimators“. *Journal of the American Statistical Association* 115 (531): 1449–55.

Cortez, Paulo, und Alice Maria Gonçalves Silva. 2008. „Using data mining to predict secondary school student performance“.

Efron, Bradley, Trevor Hastie, Iain Johnstone, und Robert Tibshirani. 2004. „Least angle regression“.

Gelman, Andrew, und Guido Imbens. 2019. „Why high-order polynomials should not be used in regression discontinuity designs“. *Journal of Business & Economic Statistics* 37 (3): 447–56.

Hahn, P Richard, Carlos M Carvalho, David Puelz, und Jingyu He. 2018. „Regularization and confounding in linear regression for treatment effect estimation“.

Hájek, J. 1971. „Comment on ‚An essay on the logical foundations of survey sampling‘ by Basu, D“. *Foundations of Statistical Inference* 236.

Hirano, Keisuke, Guido Imbens, und Geert Ridder. 2003. „Efficient Estimation of Average Treatment Effects Using the Estimated Propensity Score.“ *Econometrica* 71 (4): 1161–89. https://doi.org/10.1111/1468-0262.00442.

Hoerl, Arthur E, und Robert W Kennard. 1970. „Ridge regression: Biased estimation for nonorthogonal problems“. *Technometrics* 12 (1): 55–67.

Imbens, G. W., und Thomas Lemieux. 2008. „Regression discontinuity designs: A guide to practice“. *Journal of econometrics* 142 (2): 615–35.

Imbens, Guido, und Karthik Kalyanaraman. 2012. „Optimal bandwidth choice for the regression discontinuity estimator“. *The Review of economic studies* 79 (3): 933–59.

Lee, David S. 2008. „Randomized experiments from non-random selection in US House elections“. *Journal of Econometrics* 142 (2): 675–97.

Love, Thomas. 2004. „Graphical display of covariate balance“. Presentation.

McCrary, Justin. 2008. „Manipulation of the running variable in the regression discontinuity design: A density test“. *Journal of Econometrics* 142 (2): 698–714.

Rosenbaum, Paul R., und Donald R. Rubin. 1983. „The central role of the propensity score in observational studies for causal effects“. *Biometrika* 70 (1): 170–84. https://doi.org/10.1017/cbo9780511810725.016.

Tibshirani, Robert. 1996. „Regression shrinkage and selection via the lasso“. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 58 (1): 267–88.

Weber, Max. 2004. *Die protestantische Ethik und der Geist des Kapitalismus*. Bd. 1614. CH Beck.