

Selektion, Manipulation und Transformation

Hannah-Marie Büttner

Table of contents

1 Selektion, Manipulation und Transformation von Daten	1
1.1 Die Tidyverse-Schreibweise und der Pipe-Operator %>%	3
1.2 Daten-Selektion und Subset-Bildung	3
1.2.1 Auswahl von Spalten	4
1.2.2 Filtern von Fällen	5
1.3 Manipulation und Transformation von Daten	5
1.3.1 Neue Spalten erstellen	6
1.3.2 Variablen umbenennen	6
1.4 Gruppieren von Daten und Aggregation	6
1.5 Umgang mit realen Datensätzen und fehlenden Werten	7
1.6 Erstellung von Indizes	8

Wenn wir mit Daten arbeiten liegen diese in den seltesten Fällen in der von uns benötigten Form vor. Entsprechend wichtig ist es für uns die Fähigkeit zu besitzen Daten in die für uns gewünschte Form zu bringen. Damit beschäftigen wir uns in diesem Abschnitt.

Video

<https://nc.uni-bremen.de/index.php/s/rCEXdktAKeA9mzR/download/%234%20Selektion,%20Manipulation%20und%20Transformation%20von%20Daten.mp4>

1 Selektion, Manipulation und Transformation von Daten

In diesem Kapitel werden wir uns mit den Schritten der Datenbearbeitung befassen, die essentiell für die quantitative Datenanalyse sind. Wir werden lernen, wie wir Daten gezielt auswählen, verändern und transformieren können, um die für uns relevanten Informationen zu gewinnen. Dabei werden wir das umfassende “tidyverse”-Universum nutzen, insbesondere

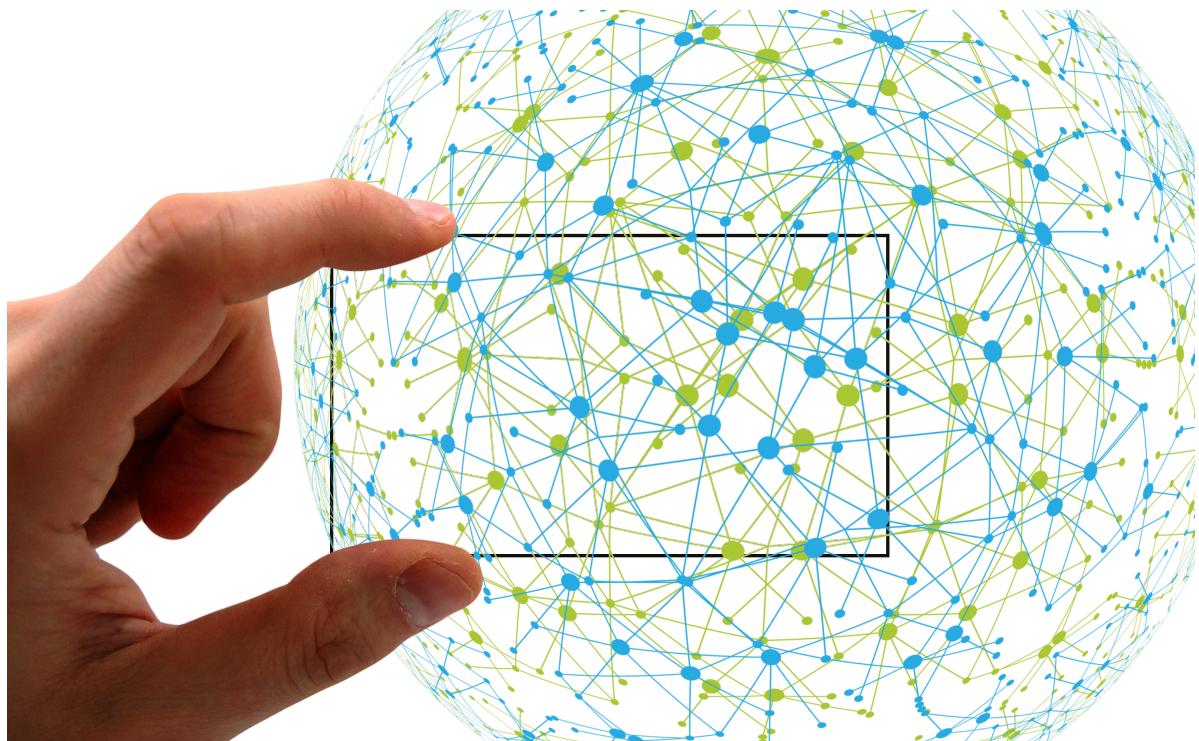


Figure 1: Bild von Gerd Altmann auf Pixabay

das leistungsstarke R-Paket “dplyr”, das speziell für diese Art von Datenmanipulation entwickelt wurde. Bevor wir mit der Selektion, Manipulation und Transformation unserer Daten starten können, müssen wir die nötigen Pakete und unseren Datensatz laden. Dazu laden wir zunächst unseren Paketmanager mit dem Befehl `library(pacman)`. Anschließend können wir mit `p_load()` die Libraries laden, die wir in diesem Kapitel benötigen. Diese sind “haven”, um unseren Datensatz in dem entsprechenden Dateiformat (Endung .dta) öffnen zu können und “tidyverse” und “dplyr”, um Daten auszuwählen, zu manipulieren und zu transformieren:

```
if(!require("pacman")) {install.packages("pacman");library(pacman)}
p_load(haven,tidyr,dplyr)
daten <- haven::read_dta("Datensatz/Allbus_2021.dta")
```

1.1 Die Tidyverse-Schreibweise und der Pipe-Operator %>%

In der tidyverse-Logik verwenden wir eine bestimmte Schreibweise, die es uns ermöglicht, viele Funktionen zu kombinieren und unseren Code trotzdem einfach verständlich und leserlich zu halten. Dabei ist der Pipe-Operator `%>%` eines der markantesten Merkmale des Tidyverse und dient dazu, komplexe Datenverarbeitungsketten auf eine lesbarere und effizientere Weise zu erstellen. Die Pipe ermöglicht es, die Ergebnisse einer vorherigen Operation als Eingabe für die nächste Operation zu verwenden, ohne explizit Zwischenvariablen erstellen zu müssen. Dies fördert nicht nur die Lesbarkeit des Codes, sondern verringert auch die Wahrscheinlichkeit von Fehlerquellen.

Die grundlegende Logik der Pipe `%>%` ist wie folgt:

```
daten %>% operation_1 %>% operation_2 %>% operation_3
```

Hier ist `daten` der Ausgangspunkt (Datensatz oder Objekt), auf dem verschiedene Operationen nacheinander ausgeführt werden. Jede Operation erzeugt eine modifizierte Version des vorherigen Objekts, die dann als Eingabe für die nächste Operation verwendet wird. Dies erzeugt eine Kette von Operationen, die von links nach rechts ausgeführt werden.

1.2 Daten-Selektion und Subset-Bildung

Die Datenanalyse beginnt oft damit, die relevanten Informationen aus einem umfangreichen Datensatz auszuwählen. Hierbei können wir uns auf bestimmte Variablen (Spalten im Datensatz) konzentrieren, die für unsere Fragestellung von Interesse sind. Das “dplyr”-Paket in R bietet uns eine einfache und mächtige Möglichkeit, diese Daten auszuwählen.

1.2.1 Auswahl von Spalten

Die Funktion `select()` ermöglicht es uns, Spalten basierend auf ihren Namen auszuwählen. Dies ist besonders hilfreich, wenn wir nur bestimmte Aspekte unserer Daten benötigen. Wollen wir beispielsweise aus unserem ALLBUS-Datensatz ein Subset ziehen, dass nur das Alter der Befragten enthält und die Variablen, die mit Vertrauen in Institutionen zu tun haben, können wir dies wie folgt machen:

```
vertrauen_institutionen <- daten %>%
  select(age, pt01, pt02, pt03, pt04, pt06, pt07, pt08, pt09, pt10, pt11, pt12, pt14, pt15, ...)
```

In diesem Beispiel generieren wir einen neuen Datenrahmen, dem wir den Namen `vertrauen_institutionen` zuordnen. Innerhalb der Funktion `select()` geben wir zuerst den Datensatz an, aus dem wir Variablen auswählen wollen. Dieser heißt in unserem Fall `daten`. Darauf folgt nach einem Komma die Auflistung jener Variablen anhand ihrer Namen im Datensatz, die wir auswählen wollen. Ein Blick in das **Codebook** des ALLBUS-Datensatzes zeigt uns, welche Variablen mit dem Vertrauen in Institutionen zu tun haben (STRG+F zur Suche im Dokument nach “Vertrauen in Institutionen”, Auflistung der Variablen in der Tabelle auf Seite xiii). Mit dem Befehl `View()`, oder indem wir auf den generierten Datensatz in unserem Global Environment (RStudio-Schaltfläche 3) klicken, können wir uns das Subset ansehen:

```
View(vertrauen_institutionen)
```

Auf eine einzelne Variable in einem Datensatz können wir mit dem Dollar-Zeichen `\$` zugreifen:

```
vertrauen_institutionen$age
```

```
<labelled<double>[5342]>: ALTER: BEFRAGTE(R)
 [1] 54 53 89 79 62 23 31 57 68 51 57 85 55 26 38 58 54 45
 [19] 49 26 83 48 48 73 62 25 54 54 51 60 49 57 58 58 39 82
 [37] 77 79 22 77 54 50 23 25 65 56 72 52 68 55 31 79 62 67
 [55] 66 23 83 62 41 57 22 38 69 62 48 64 26 73 49 38 40 -32
 [73] 50 57 42 55 31 55 68 91 63 56 77 56 30 58 60 59 51 25
 [91] 62 59 65 50 36 25 44 59 46 44
 [ reached getOption("max.print") -- omitted 5242 entries ]
```

Labels:

value	label
-32	NICHT GENERIERBAR

1.2.2 Filtern von Fällen

Neben der Auswahl von Spalten ist es oft auch erforderlich, Fälle bzw. Zeilen basierend auf bestimmten Bedingungen auszuwählen. Hierfür verwenden wir die Funktion `filter()`. Angenommen wir möchten für unser Subset `vertrauen_institutionen` nur die Daten der Personen auswählen, die sehr großes Vertrauen in das Gesundheitswesen (Variable `pt01`) haben, können wir dies wie folgt machen - aus dem Codebook können wir ableiten, dass "sehr großes Vertrauen" dem Zahlenwert 7 entspricht:

```
vertrauen_institutionen_filter1 <- vertrauen_institutionen %>%
  filter(pt01 == 7)
```

In diesem Beispiel haben wir aus dem Subset `vertrauen_institutionen` erneut ein Subset gezogen, indem wir mit der Funktion `filter()` nur die Zeilen aus dem ursprünglichen Subset ausgewählt haben, in denen die Variable `pt01` (Vertrauen in das Gesundheitswesen) den Wert 7 annimmt. Hier ist es wichtig, ein doppeltes Gleichheitszeichen zu verwenden: `pt01 == 7`. Wir können auch mehrere Filter gleichzeitig anwenden. Zum Beispiel können wir ein Subset ziehen, dass nur die Daten von Personen enthält, die sehr großes Vertrauen in das Gesundheitswesen haben und gar kein Vertrauen in das Bundesverfassungsgericht (`pt02`):

```
vertrauen_institutionen_filter2 <- vertrauen_institutionen %>%
  filter(pt01 == 7 & pt02 == 1)
```

Müssen beide Konditionen zutreffen, damit eine Zeile ausgewählt wird, verbinden wir diese mit dem und-Zeichen `&`. In unserem Global Environment sehen wir, dass dies auf vier Fälle (4 obs.) zutrifft. Wollen wir hingegen alle Fälle auswählen, bei denen entweder ein sehr großes Vertrauen in das Gesundheitswesen oder gar kein Vertrauen in das Bundesverfassungsgericht besteht, trennen wir durch den senkrechten Strich `:`:

```
vertrauen_institutionen_filter3 <- vertrauen_institutionen %>%
  filter(pt01 == 7 | pt02 == 1)
```

In unserem Global Environment sehen wir, dass dies auf 486 Fälle (486 obs.) zutrifft.

1.3 Manipulation und Transformation von Daten

In vielen Datenanalyseprojekten ist es notwendig, Variablen zu bearbeiten, um sie für Analysen oder Visualisierungen vorzubereiten. Dieser Prozess kann das Recodieren von Werten, das Umbenennen von Variablen und die Berechnung neuer Variablen umfassen. Das "dplyr"-Paket ermöglicht uns, diese Aufgaben auf eine intuitive Weise zu erledigen. Zum Beispiel verwenden wir die Funktion `mutate()` für die Transformation von Variablen und das Erstellen neuer Spalten.

1.3.1 Neue Spalten erstellen

Angenommen wir möchten eine neue Variable (Spalte) für unser Subset vertrauen_institutionen erstellen, die den Wert “großes vertrauen” annehmen soll, wenn die Variable pt01 (Vertrauen in das Gesundheitswesen) einen Wert größer vier (> 4) annimmt und “wenig vertrauen”, wenn pt01 einen anderen Wert annimmt, können wir dies folgermaßen erreichen:

```
transformierte_daten <- vertrauen_institutionen %>%
  mutate(vertrauen_gesundheitswesen = ifelse(pt01 > 4, "großes vertrauen", "wenig vertrauen"))
```

In diesem Beispiel haben wir einen neuen Datensatz generiert und ihm den Namen transformierte_daten zugewiesen. Mit den Funktionen `mutate()` und `ifelse()` haben wir basierend auf der Variable `pt01` eine neue Spalte im Datensatz erstellt mit dem Namen `vertrauen_gesundheitswesen`. Innerhalb der Funktion `mutate()` haben wir dafür zunächst unseren Datensatz angegeben, in diesem Fall `vertrauen_institutionen`. Nach einem Komma folgt der Name der neuen Variable (Spalte) mit einem Gleichheitszeichen `=`. In der Funktion `ifelse()` geben wir erst die Kondition an, die wir testen wollen (`pt01 > 4`), gefolgt von einem Komma und dem Wert, den die Variable annehmen soll, sofern die getestete Kondition zutrifft. Dann folgt nach einem erneuten Komma der Wert, den die Variable annehmen soll, falls die Kondition nicht zutrifft.

1.3.2 Variablen umbenennen

Manchmal ist es sinnvoll, Variablennamen zu ändern, um sie verständlicher zu machen oder um Konventionen zu folgen. Die Funktion `rename()` ermöglicht das Umbenennen von Variablen:

```
transformierte_daten <- transformierte_daten %>%
  rename(vertrauen_gesundheit = pt01)
```

Hier ändern wir den Namen der Variable `pt01` im Datensatz `transformierte_daten` zu `vertrauen_gesundheit`.

1.4 Gruppieren von Daten und Aggregation

Eine weitere wichtige Fähigkeit ist das Gruppieren von Fällen basierend auf bestimmten Kategorien. Die Funktion `group_by()` aus dem dplyr-Paket ermöglicht es, Datensätze nach bestimmten Variablen zu gruppieren. Zum Beispiel könnten wir den Datensatz `vertrauen_institutionen` nach dem Alter der Befragten gruppieren und das durchschnittliche Vertrauen in das Gesundheitswesen pro Alter berechnen. Wollen wir den Datensatz `vertrauen_institutionen` nach dem Alter der Befragten gruppieren und das durchschnittliche

Vertrauen in das Gesundheitswesen pro Alter berechnen, kombinieren wir die Befehle `group_by()`, `summarize()` und `mean()` nach der tidyverse-Logik wie folgt:

```
aggregierte_daten <- transformierte_daten %>%
  group_by(age) %>%
  summarize(durchschnitt_vertrauen = mean(vertrauen_gesundheit))
```

Sehen wir uns den Datensatz `aggregierte_daten` mit `View()` oder über unser Global Environment an, können wir feststellen, dass dieser aus zwei Variablen (Spalten) besteht: `age` und `durchschnitt_vertrauen`. Die Spalte `age` enthält für jedes Alter, das im Datensatz `transformierte_daten` vorkommt, eine Zeile. Die Spalte `durchschnitt_vertrauen` gibt den durchschnittlichen Wert für die Variable `vertrauen_gesundheit` für jedes Alter an. Den Durchschnitt haben wir mit der Funktion `mean()` berechnet.

1.5 Umgang mit realen Datensätzen und fehlenden Werten

Bei der Arbeit mit realen Datensätzen ist es häufig der Fall, dass diese nicht perfekt und sauber sind. Es können verschiedene Probleme auftreten, wie fehlende Daten, inkonsistente Codierungen oder unerwartete Werte. In diesem Abschnitt werden wir uns damit beschäftigen, wie Sie solche Herausforderungen bewältigen können.

Fehlende Daten sind ein häufiges Problem in Datensätzen. In R werden fehlende Werte oft mit dem Wert NA (Not Available) oder NaN (Not a Number) dargestellt. Ist dies der Fall, können wir fehlende Werte mit der Funktion `drop_na()` entfernen:

```
vertrauen_institutionen_dropna <- transformierte_daten %>%
  drop_na(vertrauen_gesundheit)
```

Je nach Datensatz kann die Codierung fehlender Werte aber stark variieren. Im ALLBUS-Datensatz ist die Variable `pt01` (`vertrauen_gesundheit`) zum Beispiel so codiert, dass fehlende Werte mit einer -9 (keine Angabe), -42 (Datenfehler: Mehrfachnennung) oder -11 (keine Teilnahme an Split A oder B) gekennzeichnet sind. Dadurch wurden durch obenstehenden Code, der `drop_na` verwendet keinerlei Zeilen gelöscht und der Datensatz `vertrauen_institutionen_dropna` entspricht `vertrauen_institutionen`. In diesem Fall müssen wir die spezifischen fehlenden Werte angeben, die wir herausfiltern wollen:

```
missing_codes <- c(-9, -42, -11)

vertrauen_institutionen_dropna <- transformierte_daten %>%
  filter(!vertrauen_gesundheit %in% missing_codes)
```

Hier geben wir zunächst die möglichen Ausprägungen für fehlende Werte an und speichern diese als Vektor missing_codes. Dann wenden wir die Funktion `filter()` auf den Datensatz `vertrauen_institutionen` so an, dass nur Zeilen ausgewählt werden, in denen die Variable keinen der Werte in `missing_codes` annimmt. Das Gegenteil einer Kondition erreichen wir mit einem Ausrufezeichen ! . `vertrauen_gesundheit %in% missing_code` würde alle Fälle filtern, die einen der Werte in `missing_codes` annehmen. Setzen wir ein ! davor, erhalten wir genau den umgekehrten Fall.

Wir können auch die fehlenden Werte aus allen Vertrauensvariablen löschen:

```
vertrauen_institutionen_dropna <- transformierte_daten %>%
  filter(across(vertrauen_gesundheit:pt20, ~ !. %in% missing_codes))
```

In dieser Version wird die Funktion `across()` verwendet, um Operationen auf mehreren Spalten gleichzeitig durchzuführen. Es werden nur die Zeilen beibehalten, in denen keine fehlenden Werte (definiert in `missing_codes`) in den angegebenen Variablen zwischen `vertrauen_gesundheit` und `pt20` vorkommen.

1.6 Erstellung von Indizes

Oft müssen aus vorhandenen Variablen neue abgeleitete Variablen berechnet werden. Dies kann beispielsweise das Berechnen von Indizes oder Skalen sein, um Zusammenfassungen oder Vergleiche zu erleichtern. Wir können zum Beispiel alle Vertrauensvariablen zu einem Index zusammenfassen:

```
vertrauen_institutionen_index <- vertrauen_institutionen_dropna %>%
  mutate(index_vertrauen = rowSums(across(vertrauen_gesundheit:pt20)))
```

Hier haben wir eine neue Spalte `index_vertrauen` erstellt, die die Summe der Vertrauensvariablen enthält. Wir verwenden wieder die Funktion `across()`, um Operationen auf mehreren Spalten gleichzeitig durchzuführen. Für jede Zeile haben wir mit dem Befehl `rowSums()` die Werte für alle Variablen von `vertrauen_gesundheit` bis `pt20` addiert und den resultierenden Wert in die neue Spalte geschrieben. Anstatt der Summe können wir auch den Mittelwert der Variablen berechnen und diesen als Index verwenden, dazu teilen wir die Summe, die wir wie im Beispiel zuvor berechnen, durch die Anzahl an Vertrauensvariablen (20):

```
vertrauen_institutionen_index_2 <- vertrauen_institutionen_dropna %>%
  mutate(index_vertrauen = rowSums(across(vertrauen_gesundheit:pt20))/20)
```