R Kurs Unterlagen

Anna-Lena Schubert, Jan Goettmann, Jose Carlos Garcia Alanis, Meike Steinhilber, Cordula Hu

2021-10-13

Inhaltsverzeichnis

Kapitel 1

Über dieses Buch

TEXT

Kapitel 2

Einführung

(Anna-Lena)

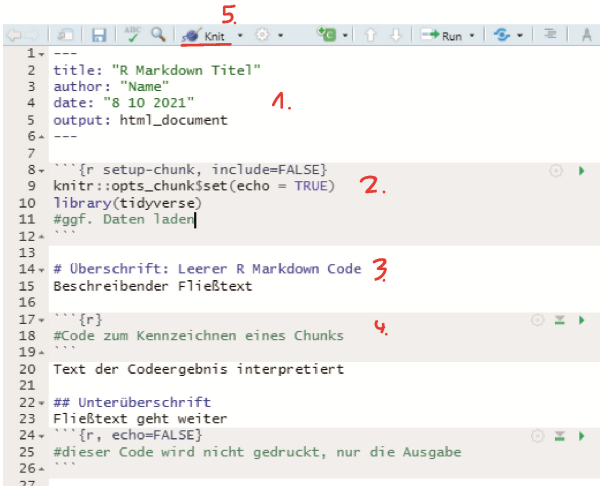
Kapitel 3

Datenstruktur

Thema	Inhalte
RMarkdown	*Titel, Chunks, knitten*
Hilfe	*help-Fenster, ?, #was passiert hier*
Werte, Vektoren & Listen	*chr, num, log, c(), list(), typeof(), coercion, Abruf von Elementen, list(list())*
Workspace	*rm(), Besen*
Berechnungen	*mit Values, Vektoren, Funktionen, z-Standardisierung*
Matrizen	*matrix(), Indizierung*
tidy Daten	*Zeilen: Beobachtungen, Spalten: Variablen*
tidyverse	*Installation und library (package)*
data.frame & tibble	*Unterschiede, as.data.frame(), as_tibble(), $, [], Zugriff auf Elemente, Reihennamen, Faktoren*
Daten laden & speichern	*Import per klick, read./_, sep=, dec=, .xlsx, .svs, write_csv()*
Daten anschauen	*View(), head(), str(), count()*

3.1 RMarkdown

Das R Markdown Skript ist ein besonderes Dateiformat für R Skripte. Es enthält Fließtext und eingebetteten R Code:




```
---
title: "R Markdown Titel"
author: "Name"
date: "8 10 2021"
output: html_document
---

```{r setup-chunk, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
library(tidyverse)
#ggf. Daten laden
```

Überschrift: Leerer R Markdown Code
Beschreibender Fließtext

```{r}
#Code zum Kennzeichnen eines Chunks
```
Text der Codeergebnis interpretiert

Unterüberschrift
Fließtext geht weiter
```{r, echo=FALSE}
#dieser Code wird nicht gedruckt, nur die Ausgabe
```
```

`Knittet` man dies Skript mit dem Wollknäul Button (5.) in der oberen Leiste, integriert es den ausgeführten Code mit dem Fließtext und druckt ein übersichtliches Dokument (html, pdf, txt oder doc). Das ist praktisch um z.B. Auswertungsergebnisse zu präsentieren.

1. Im Header werden Titel und Dokumententyp für das Ausgabe-Dokument festgelegt
2. Die Code Blöcke (`Chunks`) sind mit je drei rückwärts gestellten Hochkommata (`Backticks`) am Anfang und Ende des Chunks eingerahmt. Werden sie vom R Markdown Skript als solche erkannt, wird auch die Hintergrundfarbe automatisch abgeändert. Im ersten chunk sollten `globale Chunk Optionen festgelegt`, alle notwendigen `Packages geladen` und die `Daten eingelesen` werden.
3. Den Fließtext kann man mit Überschriften (`#`) und Unterüberschriften (`##`) strukturieren, im Code kennzeichnet `#` Kommentare
4. Zu Beginn eines Chunks muss man innerhalb einer geschwungenen Klammer spezifizieren(“‘{…}):

- Es ist möglich Code von anderen Programmiersprachen (z.B. Python oder TeX) einzubetten, standard ist `r`
- (optional) Nach einem Leerzeichen: Einzigartiger Chunk-Name
- (optional) Nach einem Komma: Befehle, um die Ausgabe des Chunks in das neue Dokument zu steuern:
  - `include = FALSE` Weder Code noch Ergebnis erscheinen
  - `echo = FALSE` Nur das Code-Ergebnis erscheint
  - `message = FALSE` Nachrichten zum Code erscheinen nicht
  - `warning = FALSE` Warnungenzum Code erscheinen nicht
  - `fig.cap = "..."` Hiermit lassen sich Grafiken beschriften

3.2 Hilfe

Sie merken, dass die Befehle und Funktionen zum Teil sehr spezifisch und Sie sich kaum alles behalten können. Am wichtigsten ist die Reihenfolge und Vollständigkeit der Zeichen: vergessen Sie ein Komma, ein Backtick oder eine Klammer zu, dann kann R den Code schon nicht interpretieren. Zum Glück erkennt R Studio das oft und weist einen darauf während des Codens mit einem `roten x` neben der Zeilennummer hin. Andernfalls dürfen Sie versuchen, die Fehlermeldung beim Ausführen zu verstehen.

Wenn Sie den Namen einer Funktion oder eines Packages nicht direkt erinnern, können Sie den Anfang des Namens im `Chunk` oder in der `Console` eingeben, RStudio bietet einem nach einem kurzen Moment eine Liste möglicher Optionen an, aus der Sie wählen können. Haben Sie eine Funktion gewählt, können sie die `Tab`-Taste drücken und es werden die verschiedenen Funktionsargumente angezeigt, um die Funktion zu spezifizieren, was oft sehr hilfreich ist. Möchten Sie wissen, was eine Funktion macht oder in welcher Reihenfolge die Funktionsargumente eingegeben werden, können Sie `?FUN` in die `Console` eintippen, wobei FUN Platzhalter für den Funktionsnamen ist. Alternativ können Sie im `Help`-tab unten rechts suchen. Die Dokumentation ist oft sehr ausführlich. Die Möglichkeit einschlägige Suchmaschinen im Internet zu verwenden ist fast zu trivial, um sie hier aufzuführen, oft werden Sie dabei auf `StackOverflow` weitergeleitet. Auf Englisch gestellte Fragen oder Probleme führen zu besseren Treffern. Noch trivialer ist es, im Skript des Kurses oder im eigenen Code nachzuschauen. Falls Sie gründlich nachlesen möchten, gibt es auch ganze Bücher, die einem eine Einführung in R geben: z.B. `R Cookbook` oder `R for Data Science`.

3.3 Werte & Vektoren

Datenformate in R sind von einfach zu komplex: `Value`, Vektor, `matrix`, (`array`), `data.frame`,`tibble` und `list`. Die kleinste Objekteinheit in R ist ein `Value`. Es gibt unterschiedliche Typen von `Values`:

1. Text, bzw. Charakter (`chr`), manchmal auch String genannt,
2. (komplexe Zahlen, `cmplx`)
3. Nummer (`num`), bzw.`double`
4. (ganze Zahlen, integer `int` genannt)
5. logische Werte (`logi`), manchmal auch Boolean genannt
6. fehlende Werte (`NA`), `Not Available`

Sie weisen einem Objektnamen einen Wert per `<-` zu (Shortkey:`ALT`&`-`), der Datentyp des `Values` wird automatisch Rkannt.

```
var1 <- "kreativ"   # typ chr
var2 <- 3.5         # typ num
var3 <- TRUE        # typ logi
```

Mit der Funktion `typeof()` können sie sich den Datentypen anzeigen lassen. Vektoren reihen Werten desselben Datentyps auf `c(Wert1, Wert2, ...)`:

```
vec1 <- c(3, 6, 3.4)    # c() kombiniert die Werte zu einem Vektor, der dem Namen zuge
```

Fassen Sie Werte von verschiedenen Typen zu einem Vektor zussammen, werden beide Werte zum Typen mit der kleineren Typenzahl umgewandelt (`coercion`).

```
c("kreativ",3.5)   # ich versuche ein `chr` und eine `num` zu einem Vektor zu kombinie
```

```
## [1] "kreativ" "3.5"
```

3.5 wird in ausgegeben, die Nummer wurde zu Text.

3.3.1 Coercion (Umwandlung von Typen)

Sie können den Datentypen auch per Funktion ändern, z.B. `as.character()`, `as.double()`:

```
as.character(c(1, TRUE, "abc", 4.1627))  # Verändert eine Reihe von Werten zum Typ chr
```

```
## [1] "1"      "TRUE"   "abc"    "4.1627"
```

```
as.double(c(2, TRUE, "abc", 4.1627))    # Verändert die Werte zum Typ double, geht es
```

```
## Warning: NAs durch Umwandlung erzeugt
```

```
## [1] 2.0000     NA     NA 4.1627
```

`Coercion` gibt es auch in Matrizen, Arrays (Mehrdimensionale Matrizen) und in Spaltenvektoren von Datensätzen (data.frames und tibbles). Nur Listen können verschiedene Datentypen und Elemente enthalten `list(Element1, Element2, ...)`. Das geht soweit, dass Listen selbst Listen enthalten können.

3.3.2 Aufruf einzelner Elemente per Index:

Um auf Elemente zuzugreifen, können Sie deren Indexnummer verwenden:

```
vec_4 <- c(1,3,3,7)   #Definition des Vektors
vec_4[2]              #Abruf des zweiten Elements des Vektors
```

```
## [1] 3
```

Das geht sogar in verschachtelten Listen:

```
mylist <- list(list(1,"a"),c(2,3)) # Definiert eine Liste aus Liste & Vector, die je a
mylist[[1]][2]                     # Ruft Element 1 der äußeren Liste: (1,"a"), und da
```