3. Chunk-Optionen

Kontrollieren, was im Dokument erscheint

Dr. Paul Schmidt

Im vorherigen Kapitel haben wir ein einfaches Quarto-Dokument mit R-Code-Chunks erstellt. Standardmäßig zeigt Quarto sowohl den Code als auch seine Ausgabe. Aber in einem professionellen Report will man oft mehr Kontrolle — den Code verstecken, Warnungen unterdrücken oder nur bestimmte Ausgaben zeigen. Hier kommen Chunk-Optionen ins Spiel.

Grundlagen der Chunk-Optionen

Chunk-Optionen werden am Anfang eines Code-Chunks mit der `#|`-Syntax angegeben (gelesen als “Hash-Pipe”). Jede Option steht in einer eigenen Zeile:

````
```{r}
#| label: my-chunk
#| echo: false
#| message: false

library(tidyverse)
```
````

Diese Syntax ist spezifisch für Quarto. Wer zuvor R Markdown verwendet hat, kennt vielleicht den alten Stil `{r echo=FALSE}` — das funktioniert immer noch, aber der neue `#|`-Stil ist übersichtlicher und besser lesbar.

Die wichtigsten Chunk-Optionen

Hier sind die Chunk-Optionen, die man am häufigsten verwenden wird:

echo — Code zeigen oder verstecken

Die `echo`-Option kontrolliert, ob der Code selbst in der Ausgabe erscheint.

Einstellung	Code sichtbar?	Ausgabe sichtbar?
`echo: true` (Standard)	Ja	Ja
`echo: false`	Nein	Ja

Anwendungsfall: In einem Report für nicht-technische Leser will man typischerweise `echo: false` — sie interessieren sich für Ergebnisse, nicht für Code.

```
[1] 38.82397
```

Der Code, der diese Ausgabe erzeugt hat, wurde mit `#| echo: false` versteckt.

eval — Code ausführen oder überspringen

Die `eval`-Option kontrolliert, ob der Code überhaupt ausgeführt wird.

Einstellung	Code läuft?	Ausgabe erzeugt?
`eval: true` (Standard)	Ja	Ja
`eval: false`	Nein	Nein

Anwendungsfall: Beispielcode zeigen ohne ihn auszuführen, oder eine langsame Berechnung vorübergehend deaktivieren.

```
```{r}
#| eval: false
# Dieser Code wird angezeigt, aber nicht ausgeführt
sehr_langsame_berechnung()
```
```

include — Die Radikaloption

Die `include`-Option ist eine Abkürzung, die alles versteckt — Code, Ausgabe, Nachrichten, Warnungen.

Einstellung	Irgendetwas sichtbar?
`include: true` (Standard)	Ja
`include: false`	Nein (aber Code läuft trotzdem!)

Anwendungsfall: Setup-Chunks, die Pakete laden oder Daten vorbereiten. Der Code läuft, aber nichts überfrachtet das Dokument.

Der obige Chunk lief still. Wir können sein Ergebnis verwenden: 84.

message und warning — Benachrichtigungen unterdrücken

R-Funktionen erzeugen oft Nachrichten und Warnungen. Diese sind während der Entwicklung hilfreich, aber in einem finalen Report ablenkend.

```
library(scales)
```

```
Attache Paket: 'scales'
```

```
Das folgende Objekt ist maskiert 'package:purrr':

    discard
```

```
Das folgende Objekt ist maskiert 'package:readr':

    col_factor
```

Um diese zu unterdrücken:

```
```{r}
#| message: false
#| warning: false
library(scales)
```
```

> **Tipp**
>
> Für das Laden von Paketen empfehle ich, immer `message: false` zu verwenden. Die "Attaching package"-Nachrichten sind in einem Report selten nützlich.

output — Ausgabeformat kontrollieren

Die `output`-Option hat mehrere Einstellungen, aber die nützlichste ist `asis`:

Einstellung	Verhalten
`output: true` (Standard)	Normale Ausgabe
`output: false`	Ausgabe verstecken
`output: asis`	Ausgabe als rohes Markdown/HTML behandeln

Anwendungsfall: `output: asis` ist essentiell bei Paketen wie flextable, die formatierte Ausgaben erzeugen. Das werden wir in Kapitel 5 ausgiebig verwenden.

Globale Optionen setzen

Wenn man dieselben Optionen für alle Chunks im Dokument will, kann man sie global im YAML-Header setzen:

```yaml
---
title: "Mein Report"
format: docx
execute:
  echo: false
  warning: false
  message: false
---
```

Nun verstecken alle Chunks standardmäßig Code und unterdrücken Warnungen/ Nachrichten. Man kann diese Einstellungen in einzelnen Chunks immer noch überschreiben.

> **Hinweis**
>
> Für professionelle Reports verwende ich typischerweise diese globalen Einstellungen und zeige Code nur, wenn es didaktisch sinnvoll ist.

Chunk-Labels

Jeder Chunk sollte ein Label haben. Das dient mehreren Zwecken:

1. **Navigation**: RStudio zeigt gelabelte Chunks in der Dokumentübersicht
2. **Debugging**: Fehlermeldungen referenzieren das Chunk-Label
3. **Querverweise**: Man kann Abbildungen und Tabellen über ihr Label referenzieren (Kapitel 7)

```
```{r}
#| label: summary-statistics
```
```

```
mean(adelie$bill_length_mm)
```
```

**Label-Regeln:**

- Nur Kleinbuchstaben, Zahlen und Bindestriche verwenden
- Keine Leerzeichen oder Unterstriche
- Muss innerhalb des Dokuments eindeutig sein
- Für dieses Projekt mit `qrt-` präfixen für Eindeutigkeit über Kapitel hinweg

# Schnellreferenz-Tabelle

| Option | Zweck | Übliche Werte |
|---|---|---|
| `echo` | Code zeigen? | `true`, `false` |
| `eval` | Code ausführen? | `true`, `false` |
| `include` | Irgendetwas zeigen? | `true`, `false` |
| `output` | Ausgabe zeigen? | `true`, `false`, `asis` |
| `message` | Nachrichten zeigen? | `true`, `false` |
| `warning` | Warnungen zeigen? | `true`, `false` |
| `error` | Bei Fehler fortfahren? | `true`, `false` |
| `label` | Chunk-Bezeichner | beliebiger gültiger Name |

Für die vollständige Liste der Optionen siehe die Quarto-Dokumentation zu Code Cells.

# Praktisches Beispiel

So könnte ein typisches Report-Setup aussehen:

````
---
title: "Pinguin-Analyse"
format: docx
execute:
  echo: false
  warning: false
  message: false

```{r}
#| label: setup
#| include: false
library(tidyverse)
library(palmerpenguins)
library(flextable)

adelie <- penguins %>%
  filter(species == "Adelie") %>%
  drop_na()
```

Einleitung

Dieser Report analysiert 146 Adelie-Pinguine.
````
````

````
```{r}
#| label: summary-table
#| output: asis
adelie %>%
  summarise(
    n = n(),
    mean_bill = mean(bill_length_mm),
    sd_bill = sd(bill_length_mm)
  ) %>%
  flextable()
```
````

In diesem Setup:

- Globale Optionen verstecken Code und unterdrücken Nachrichten für alle Chunks
- Der Setup-Chunk verwendet `include: false` um still zu laufen
- Die Summary-Tabelle verwendet `output: asis` für korrektes flextable-Rendering

> Übung: Kontrolliere deine Ausgabe
>
> 1. Nimm das Dokument aus Kapitel 2
> 2. Füge `execute:`-Optionen zum YAML-Header hinzu, um standardmäßig allen Code zu verstecken
> 3. Füge einen Setup-Chunk mit `include: false` für das Laden von Paketen hinzu
> 4. Überprüfe, dass das gerenderte Word-Dokument nur Ergebnisse zeigt, keinen Code

Was kommt als Nächstes

Jetzt wo man kontrollieren kann, was im Dokument erscheint, ist der nächste Schritt, es professionell aussehen zu lassen. In Kapitel 4 lernen wir, wie man Word-Vorlagen verwendet, um einheitliche Formatierung anzuwenden — Schriftarten, Farben, Überschriftenstile und mehr.

Bibliography