Kurze Einführung in Quarto

Eine Arbeitsumgebung zum Coden in R und Schreiben in Markdown

Frank Siegmund

2023-01-09

Table of contents

Einleitung

Mein nächstes größeren Buchprojekt werde ich mit Quarto schreiben, der Mitte 2022 von "posit" lancierten Schreibumgebung. Also arbeite ich mich ein und beginne mit einer ersten Fingerübung. Da Quarto noch relativ neu ist, gibt es m. W. bislang keine deutschsprachige Einführung jenseits von YouTube. Damit ist meine Schreibmotivation für das Folgende umrissen.

Warum Quarto?

Quarto ist eine Schreibumgebung, die auf der weit verbreiteten Auszeichnungssprache Markdown beruht. Sie zielt darauf ab, Markdown-Dokumente erzeugen und pflegen zu können, die Text und Code eng vereinen. Neben R-Code kann in Quarto insbes. auch Python-Code eingebettet werden. Quarto ist somit sehr ähnlich zu RMarkdown (und dessen Derivaten wie Bookdown etc.), im Grunde seine leistungsfähigere Weiterentwicklung. Die Entwickler von RStudio, das als Nutzeroberfläche und Entwicklungsumgebung für R sehr weit verbreitet ist, entwickeln auch Quarto und zielen auf eine enge Interaktion von RStudio und Quarto. Im Unterschied zu RMarkdown, das nur als Teil (Paket) von RStudio betrieben werden kann, ist Quarto jedoch auch als autonomes Programm aufrufbar. Doch in dieser Einführung wollen wir Quarto nicht autonom nutzen, sondern unter RStudio einsetzen und gehen in den Beispielen davon aus, dass der eingebettete Code R-Code ist resp. sein soll.

Warum Markdown?

Markdown ist eine "Auszeichungssprache" (wie z.B. auch LateX) und steht für eine andere Art des Schreibens. Schreiben bedeutet heute für Viele: MS-Word, LO-Writer, Textmaker o.ä., d.h. Programme, die im "WYSIWYG" arbeiten, "what you see is what you get". Schreibende arbeiten an den Inhalten und sind stets parallel auch die Gestalter ihres Textes, d.h. achten auf Schriftgröße und -art, auf Zeilenabstände, Blocksatz und Trennungen, usw. Die Idee von Auszeichnungssprachen wie LaTex oder Markdown ist: erst auf den Inhalt fokussieren und auf das Schreiben, die Gestaltung folgt später. Folglich werden Markdown-Text nur sparsam mit Auszeichnungen versehen, im Sinne "hier Überschrift 2. Grades", "hier kursiv" usw.

In der Praxis hat sich das 2004 ziemlich puristisch eingeführte Markdown weiterentwickelt und bereichert. Die meisten Markdown-Editoren bieten heute u.a. Fußnotenverwaltung, Abbildungsintegration und -verwaltung und die Interaktion mit einer Literaturverwaltung wie z.B. Zotero. Aber die Ausgabe als gestaltetes Produkt erfolgt stets erst nach dem Schreiben. Wobei Markdown-Dokumente - auch das ist ein Ziel dieses Ansatzes - wahlweise in unterschiedliche Ausgabeformate exportiert werden können. Eine Ausgabe als HTML, PDF oder *.doxc sind Standard, Quarto erzeugt aber auch Ausgaben z.B. als *.odt, *.pptx u.a.

Quarto & Alternativen

Wen die Idee von Markdown interessiert, aber wenig mit R oder Python zu tun hat, schaue sich statt Quarto eher einen der gängigen guten Markdown-Editoren an. Mir persönlich z.B. gefallen "Obsidian" und "Zettlr" sehr, beide in ihrer Ausrichtung leicht unterschiedlich. Für das Verfassen eines größeren, code-losen Buches würde ich heute "Zettlr" benutzen. Als Notizsystem und Zettelkasten ist "Obsidian" sehr geschmeidig, man kommt mit geringem Lernaufwand schnell in eine produktive Phase. RMarkdown und Quarto kommen da auf's Spielfeld, wo Text und Code eng zusammenhängen, z.B. beim Verfassen von Lehrbüchern für

R, angewandte Statistik oder von Forschungstexten, die stark mit Daten und Statistik argumentieren. Wenn neben dem eigentlichen Text auch Code und Daten, die im Sinne reproduzierbarer Forschung mitpubliziert werden sollen, sind Quarto oder RMarkdown die Instrumente der Wahl.

Quarto ist noch jung, es wurde im Sommer 2022 auf der RStudio-Konferenz offiziell vorgestellt.[1] Seitdem debattieren an RMarkdown Gewohnte: warum doch bei RMarkdown bleiben, warum mit Quarto beginnen? Wer sich für die Argumente interessiert, lese z.B. folgende Beiträge:

- Alison Hill (4.4.2022): "We don't talk about Quarto. Until now!" (Blog): https://www.apreshill.com/blog/2022-04-we-dont-talk-about-quarto/
- Nicholas Tierney (11.4.2022): "Notes on Changing from RMarkdown/Bookdown to Quarto" (Blog): https://www.njtierney.com/post/2022/04/11/rmd-to-qmd/
- Yihui Xie (29.4.2022): "With Quarto Coming, is R Markdown Going Away? No." (Blog): https://yihui.org/en/2022/04/quarto-r-markdown/
- Sharon Machlis (14.7.2022): "What is Quarto? RStudio quietly rolls out next-generation R Markdown" (InfoWorld): https://www.infoworld.com/article/3666743/what-is-quarto-rstudio-quietly-rolls-out-next-generation-r-markdown.html

Wie Sie sehen: ich habe mich für Quarto entschieden, weil's schon jetzt einfach rund läuft und weil es von posit/RStudio[2] entwickelt wird - womit Nachhaltigkeit gesichert. Nach meiner Einschätzung wird Quarto künftig dynamischer weiterentwickelt und länger gepflegt werden als RMarkdown.

Erste Schritte: Quarto installieren und starten

Zunächst kümmern wir uns um RStudio: prüfen, ob die installierte Version aktuell ist (Quarto braucht RStudio v2022.07 oder jünger). Ggf. updaten oder überhaupt RStudio erstmals installieren. Quelle (dort). Üblicherweise wählt man die Version Desktop, Open Source Edition, "free", für das eigene Betriebssystem. Installation wie heute üblich.

Nun zu Quarto, kostenlos auf der Quarto-Homepage (dort). Ebenfalls Installation wie heute üblich. Quarto legt sich automatisch ins RStudio hinein. RStudio starten. In der obersten Bedienleiste ganz links auf den Reiter "File" gehen, dann "New File" und die Auswahloptionen sichten. Neu findet sich - oben unter R Script - "Quarto Document" und "Quarto Presentation".

[1] Allaire, J. J. (28.7.2022). Announcing Quarto, a new scientific and technical publishing system. https://www.rstudio.com/blog/announcing-quarto-a-new-scientific-and-technical-publishing-system/

[2] "Rstudio" ist der Name der Software, d.h. der Entwicklungsumgebung für R und seit ca. 2021 auch Python. Bis Sommer 2022 war es zugleich der Name der Firma, die RStudio entwickelt. Seit der RStudio-Konferenz 2022 wurde der Unternehmensname geändert in "posit", während die Software ihren Namen RStudio behält. Siehe: <https://www.rstudio.com/blog/rstudio-is-becoming-posit/>.

Quarto Document ist der erste Startpunkt. Während Markdown-Dateien nach dem Punkt mit ".md" bezeichnet werden, RMarkdown-Dateien mit ".rmd", heißen Quarto-Dateien ".qmd". Mit diesem Wissen und diesen Vorbereitungen können wir anfangen und loslegen.[3]

Sobald Sie starten, fragt Quarto ev. nach dem Paket "markdown" und dem Paket "tinytech", die für die Ausgabe benötigt werden. Ggf. bitte installieren.

"hello world" - einfach loslegen mit Quarto

Wie? RStudio starten. File => New File => Quarto Document.

Es öffnet sich ein PopUp-Fenster, das vorschlägt, einen Titel und einen Autor einzutragen. Tun Sie das! - z.B. „Hallo Welt" und „Emil E. Neumann", oder noch besser Ihr Name.

Dann die drei Output-Optionen bestimmen, in welcher Form das Dokument nach dem Schreiben ausgegeben werden soll: als HTML (d.h. wie eine Website), als PDF oder als MS-Word-Dokument. Kein Entscheidungsstress: diese Einstellung kann man immer wieder ändern, und im übrigen auch andere Ausgabeformate bestellen. Ich wähle bevorzugt HTML, weils das m. E. für's Schreiben und Arbeiten am Text am besten geeignet ist.

Als „Engine" ist vermutlich „Knitr" voreingestellt, prima, behalten Sie das bei. Mit all diesen Einstellungen werden wir uns später vertiefend beschäftigen, aber für den Anfang kann man die Voreinstellungen übernehmen. Nun also die Schaltfläche „Create" drücken, … und in ziemlich kurzes Quarto-Dokument wird geöffnet (im linken oberen Fenster von RSTudio).

Gleich über dem Textfeld findet man zwei Steuerungsleisten. Die unterste zeigt links „Source" und „Visual". Wenn man „Source" aktiviert, sieht man den rohen Markdown-Text mit seinen Auszeichnungen, wenn man „Visual" drückt, erhält man das von Textverarbeitungprogrammen gewohnte WYSIWYG-Bild.

Wir klicken auf „Visual" und schreiben oberhalb von „## Quarto" z.B. Folgendes:

Hallo Welt!

machen eine Leerzeile und schreiben dann z.B.:

Neue Studien zum Untergang von Atlantis, von Alfred E. Neumann.

Genug! Oberhalb des Schreibfensters, in der zweituntersten Steuerleiste sehen Sie einen hellblauen Pfeil mit „Render". Draufklicken. Ein PopUp-Menü wünscht, dass Sie die Datei sichern. Tun Sie dies (wir kommen auf das Thema „wohin" gleich noch einmal zurück). Also z.B.

[3] Keine Sorge wegen dieser scheinbar verschiedenen Dateiformate. Die Markdown-Dialekte sind weitgehend kompatibel. Man ist also nicht in einem Format (und einer Software) gefangen, sondern kann Quarto-Dokumente auch mit anderen Markdown-Editoren lesen und bearbeiten. Ich habe das nicht rundum getestet - auch weil die Entwicklung sehr dynamisch ist und Aussagen schnell veralten können - aber ein Lesen von *.qmd-Dateien mit dem weit verbreiteten (und empfehlenswerten) Editor Notepad++ war problemlos möglich.

„MsHalloWelt” und „Save”. Jetzt passiert einige Sekunden lang scheinbar nichts, doch tatsächlich ist Quarto dabei, den Text zu „rendern” (nei RMarkdown hieß das „knitten”/stricken) - auch das werden wir noch vertiefen. Im RStudio-Fenster unten links taucht neben dem gewohnten Reiter „Console” (das ist die R-Console) ein weiterer Reiter „Background Jobs” auf: in diesem Fenster wird das Rendern protokolliert. Langweilig, wenn alles wie üblich rund läuft, hilfreich, wenn nicht, weil man dann dort den Verlauf nachvollziehen kann und ein Fehlerprotokoll bekommt, d.h. die Klemmstelle identifizieren kann.

Doch während wir noch kurz in dieses Protokoll schauen, hat sich in Ihrem Webbrowser ein neuer Tab geöffnet, Ihr Manuskript „Hallo Welt”. Ein Anfang ist gemacht! Ich rege an, dass Sie nun parallel zum Lesen dieser Einführung mit diesem Text arbeiten - so sinnfrei er auch sein mag. Learning by doing.

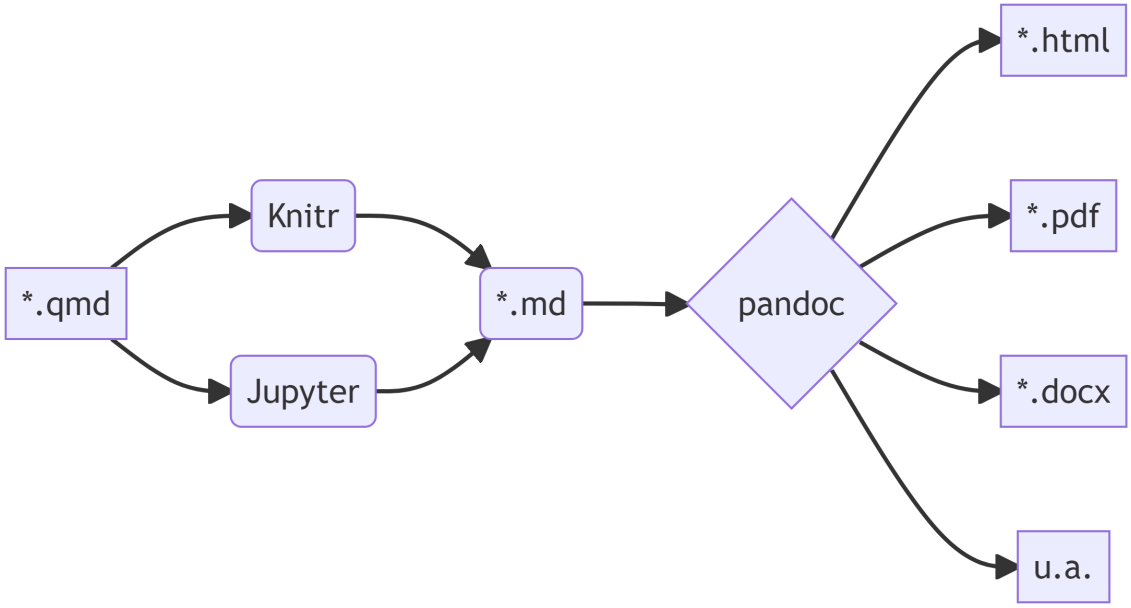

Figure 1: Demonstration, dass Quarto mit dem integrierten Tool “Mermaid” auch Ablaufdiagramme / Flow Charts erzeugt. Gleichzeitig verdeutlicht das Diagramm, was beim Rendern geschieht: die *.qmd-Arbeitsdatei wird via “Knitr” (oder “Jupyter”) in eine Markdown-Datei gewandelt, diese an “Pandoc” übergeben, und Pandoc spielt dann das im sog. YAML-Header bestellte Format aus.

Das Flußdiagramm

… wurde mit der Markdown-Komponente “mermaid” erstellt. Eine gründliche Dokumentation findet man dort; einen interaktiven Editor, anhand dessen man das Programm intutiv und anschaulich lernt dort.

*.docx oder *.odt statt HTML

Bitte gehen Sie noch einmal in Ihre Hallo-Welt-Datei. Ersetzen Sie oben in der Zeile “format: html” das “html” durch “docx”, oder “odt” - je nachdem, welche Office Suite Sie nutzen. Dann wieder Rendern. Ein paar Sekunden Geduld. Dann erscheint ihr Text im vertrauten MS-Word oder LO-Writer. Nicht nur das: Quarto legt auch eine entsprechende Datei an, die Sie in dem Ordner finden, in dem auch die Hallo-Welt-qmd-Datei liegt. So einfach ist es, statt HTML andere Formate auszugeben. So kann man, wenn man mit anderen zusammenarbeitet, entweder gemeinsam mit Quarto arbeiten, oder eben Texte für die Partner z.B. als *.docx ausgeben.

Nicht minder interessant: Ersetzen Sie bei format: htlm / docx / odt durch “revealjs” und Rendern Ihre Datei. Es wird eine Präsentation ausgegeben (mit den Pfeiltasten rechts/links kann man weiterblättern oder auch zurückblättern). Dies nur als Vorschau, was möglich ist. Selbstverständlich kann man die Folien weitaus mehr gestalten als wir es in diesem Startbeispiel getan haben. Wer jetzt schon mehr wissen will: (dort). Statt nach HTML können Präsentationen auch als ppt oder PDF ausgegeben werden.

Warnung betr. PDF rendern

Meine Quarto-Installation machte problemlos Ausgaben nach HTML. Als ich erstmals ein pdf bestellte, gab’s Hänger, weil TinyTex nicht vorhanden sei, es aber lt. RStudio durchaus war. Da es zu diesem Thema diverse Fragen und Hilfen im WWW gab, scheint dies ein bekannter Quarto-Mangel zu sein. Am Ende ließ sich der Hänger reparieren, und zwar mit:

- tinytex::install_tinytex() # bestellt eine (erneute) Installation von tinytex; ca. 140 MB, dauert!
- tinytex::tinytex_root() # should return where the installation took place
- tinytex::is_tinytex() # will confirm that it is a tinytex, “TRUE”
- tinytex::tlmgr_path(“add”) # should run the command to add in PATH.

… wobei Zeile 1 und 4 match-entscheidend sind. Danach und nach einem völligen Neustart des Systems (nicht **R**, sondern Windows) und sehr sehr viel Geduld beim anschließenden Rendern, weil noch gefühlt ca. 1/2 Stunde lang Pakete nachinstalliert wurden, lief’s.

Dateinamen, Ordner usw.

Quarto-Dateien enden auf *.qmd. Es sind Markdown-Dateien, d.h. sie können auch mit anderen Markdown-Editoren gelesen werden, wie z.B. Notepad++. Unsere Hallo-Welt-Übungsdatei ist eventuell nicht kostbar. Sollte aber daraus die Projektskizze zu Ihrem neuen wissenschaftlichen Aufsatz anwachsen und dann zum gediegenen Buch, ist es nützlich, von Anbeginn Ordnung zu halten. Meine Empfehlung: 1 Projekt = 1 Ordner auf Ihrem PC/Laptop, und zugleich ein RStudio-Projekt. Warum? Man kann in RStudio "Projekte" anlegen, bei denen alle Dateien in 1 Ordner versammelt sind. Vorteil: man kommt nicht durcheinander mit seinen verschiedenen Projekten: alles, was zum Projekt "Einführung in Quarto" gehört, liegt beisammen in einem Ordner, und der Vortrag "Einführung in die Atlantis-Kunde" mit all seinen Grafiken und Fotos fein sortiert in einem anderen Ordner. Das Ganze macht RStudio für Sie, sofern Sie ein Projekt anlegen: in RStudio ganz oben rechts ist die entsprechende Schaltfläche, "Projekt anlegen", usw. - wie dort beschrieben. Mit RStudio kann man zwischen seinen verschiedenen Projekten geschmeidig hin- und herschalten. Wie gesagt: hilft ungemein, in der Fülle Ordnung zu halten. Es ist wie im sonstigen Leben: wenn man gleich mit solch einer Struktur anfängt, braucht man hinterher kein Chaos aufräumen ;-)

Wie geht es weiter? ...und wo gibt es Hilfe?

Da wir den grundsätzlichen Ablauf an einem kleinen Beispiel kennen gelernt haben, kann das Folgende systematischer aufgebaut werden. Ziel: ein kleines Quarto-Lexikon der wichtigsten Dinge anlegen, in dem man das, was man sucht, meist findet. Wer's "alles" wissen will: Quarto ist gut dokumentiert! - weiteres findet man hier https://quarto.org/docs/get-started/hello/rstudio.html - vor allem unter "Guide" and "Reference".

Im Juli 2022 recht aktiv mit vielen nützlichen Hinweisen war das Blog "A Quarto tip a day" der Statistikerin Mine Cetinkaya-Rundel, hier: https://mine-cetinkaya-rundel.github.io/quarto-tip-a-day/.

Wer gerne mit Anschauung und Videos übt: Das knapp einstündige Video von R Ladies Freiburg / Julia Müller "Getting to know Quarto" (3.6.2022; 53 min.) empfand ich als sehr anschaulich und hilfreich: https://www.youtube.com/watch?v=shVSmYna3GM. Kürzer, nur das Wichtigste einführend: Lyndon Walker: "Create beautiful documents with Quarto and R" (4.4.2022; 29 min.): https://www.youtube.com/watch?v=y5VcxMOnj3M. Schön systematisch, in Summe aber mehr Zeit kostend: Andy Field: "Quarto tutorials 1 - 5" https://www.youtube.com/watch?v=31Q9ZTZOHIM (23.9.2022, je ca. 10 min.)

Zunächst geht es um das Schreiben mit Markdown: wie man Texte auszeichnet; wie man Listen anlegt; wie man Tabellen anlegt; wie man Links in die Texte einbettet; wie man Bilder in die Texte einfügt; wie man Textblöcke hevorhebt und Randbemerkungen anlegt. Anschließend

lernen den YAML-Header besser kennen, der die Ausgaben steuert. Danach lösen wir das Versprechen des Titels ein und integrieren R-Code in die Texte.

Markdown

Die Einleitung hat klar gemacht, dass wir uns als Einstieg in Quarto zunächst einmal um Markdown kümmern müssen, d.h. das Wie des Schreibens. Die weiteren Themen und Schritte: "chunks" (dt. Stück, Batzen, Klumpen), die R- oder Python-Code-Abschnitte innerhalb der Texte, und dann das "Rendern" der Texte, d.h. ihr Export in Ausgabeformate. Da letzteres vielfältige Möglichkeiten umfasst, die vor allem im sog. YAML-Header gesteuert werden, ist YAML ("yet another markup language") dann noch ein weiteres Thema.

Texte Auszeichnen in Markdown

Die Auszeichnungssprache Markdown kennt einige leicht divergierende Dialekte, daher sei eingangs klargestellt: Quarto basiert auf der Variante "Pandoc-Markdown", deren ausführliches Handbuch man dort findet.

Die einen Text gliedernden Zwischenüberschriften werden mit einem vorangestellten Hashtags "#" ausgezeichnet. Quarto erzeugt bis zu sechs Ebenen, sechs aufeinander folgende Hashtags stehen dann für die 6. Gliederungsebene.

Überschrift 1

Überschrift 2

Überschrift 3

Überschrift 4

Überschrift 5

Überschrift 6

Bei dieser Gelegenheit: Haben Sie schon die kleine Schaltfläche rechts oben über dem Quarto-Schreibfenster entdeckt oder gar gedrückt: "Outline" - dt. Gliederung. Hier tauchen die gesetzten Überschriften als Textgliederung auf. Ich rate sehr dazu, dieses Instrument zu benutzen, weil man damit per Mausklick innerhalb eines Textes sehr schnell zu einer Überschrift springen kann - was bei längeren Dokumenten weitaus schneller geht als Blättern.

Wörter oder Textabschnitte können durch eine Markdown-Auszeichnung kursiv, fett etc. gestellt werden, indem sie am Anfang und am Ende zwischen entsprechende Zeichen einbettet. Ein Sternchen vor und nach dem Text(abschnitt) macht Kursivschrift *abc*, zwei Sternchen machen Fettschrift **abc**, zwei Tilden einen durchgestrichenen Text: "~~abc~~". Zwei einfache Anführungen erzeugen eine Hervorhebung: `abc` (nein, nicht die einfachen geraden Anführungen,

sondern jene, die leicht von links oben nach rechts unten zeigen: “‘”). Hier wiederholend Auszeichnungscode und Wirkung:

kursiv => *kursiv*

fett => **fett**

~~durchgestrichen~~ => ~~durchgestrichen~~

‘hervorgehoben‘ => `hervorgehoben`

Das Hochsetzen von Zeichen geschieht durch das Einbetten zwischen zwei Hütchen (“^”) das Tiefstellen durch das Einbetten zwischen zwei Tilden (“~”). So lassen sich z.B.

^14^C-Daten => ^{14}C-Daten

schreiben, oder Wasser als

H~2~O => H_2O

Puh! - muss ich das alles auswendig lernen, nur um einen einfachen Text schreiben zu können? Nein! Weil Quarto links oben über dem Schreibfenster diese geniale Schaltfläche “Source” und “Visual” hat. Wer - wie hier eingeführt - mit den manuellen Auszeichnungen schreiben will, wählt “Source” und sieht alle Auszeichnungen. Wer’s gerne einfacher und intuitiver haben möchte, wählt “Visual”: Der Text sieht jetzt wie gewohnt aus, die Auszeichnungen sind unsichtbar, aber umgesetzt. Vor allem: über dem Quarto-Schreibfenster findet man nun eine Formatierungsleiste, wo alle Formatierung wie gewohnt über kleine Menüs per Mausklick eingefügt und umsetzt werden können. Das macht das Schreiben mit Quarto sehr einfach. Zugleich eine Möglichkeit, Markdown schrittweise zu lernen: im Modus “Visual” schreiben und später im Modus “Source” anschauen, welche Auszeichnungen man gesetzt hat.

Damit kann an dieser Stelle darauf verzichtet werden, näher zu erläutern, wie man Bulletpoint-Listen oder nummerierte Listen anlegt, Links und Anker setzt, Bilder / Abbildungen in seinen Text einbettet und Tabellen baut, denn all das ist im Visual-Modus von Quarto über die über dem Schreibfenster sichtbaren Formatierungsoptionen möglich. Die folgende Tabelle:

Spalte 1	Spalte 2	Spalte 3
linksbündiger Text	mittig gesetzt	Zahlen, rechtsbündig
abc	a	1,0
jkl	b	2,0
xyz	c	3,4

… ist nur eingesetzt, damit man im Hin- und Herspringen zwischen “Visual” und “Code” prüfen kann, mit welchem (einfachen) Formatierungsanweisungen die Tabellen gestalten kann.

"Callout-notes": hervorgehobene Textblöcke und Randbemerkungen

Eine Besonderheit bei Quarto sind "Callout-Notes". Detaillierte Erläuterung dazu findet man z.B. (dort). Doch statt langer Vorreden hier ein Beispiel. Nach dem Rendern sieht man, was diese Anweisungen tun.

Es gibt in Quarto fünf verscheidene Arten von Callout-Notes: `note`, `warning`, `important`, `tip`, and `caution`.

Nun ein Block "Hinweis": ::: callout-tip **Hinweis**

Dies ist ein Beispiel für einen Hinweis-Block ("note"). :::

Ein Text mit der praktischen Oprion "Einklappen" / Ausklappen", was naheliegenderweise bei Ausgaben nach HTML gut funktioniert, bei einer Ausgabe nach *.docx oder *.pdf hingegen sinnfrei ist. ::: {.callout-caution collapse="true"} ## Beispiel für einen Textblock mit der Option "mehr", zum Aus- und Einklappen

Dies ist ein Beispiel für einen "gefalteten Text", d.h. einen Textblock, den der Leser/Nutzer ausklappen kann, wenn er sich näher informieren möchte. Als Schreibender kann man eine Voreinstellung setzen: `collapse="true"` hält den Text eingeklappt und der Leser faltet ihn bei Interesse auf; `collapse="false"` macht den Text per Voreinstellung komplett sichtbar, der Leser/Nutzer kann ihn aber einfalten. :::

Abbildungen einbetten

Das Einbetten von Abbildungen ist schneller getan als beschrieben: Im Modus "Visual" die Schaltfläche drücken, die wie ein Bild aussieht. Es poppt ein Menü auf, das alles Nötige beinhalten. Mit "Browse" sucht man das Bild und lädt es; bei "Caption" gibt man ihm eine Bildunterschrift. Beim Ablegen kann man die Zielgröße des Bildes eingeben - fertig. Im Beispiel hier liegt die Bilddatei im gleichen Ordner wie die *.qmd-Datei, aber man kann als Quelle z.B. auch externe URLs angeben.

"Chunks": R-Code integrieren

In diesem Abschnitt sehen wir, wie R-Code in Quarto-Dokumente eingebettet wird und in ihnen oder beim Rendern ausgeführt werden kann. Dazu lesen wir im ersten Schritt einen bei R-Anwendern weithin bekannten Datensatz ein und informieren uns über diese Daten:

```
data("mtcars")
head(mtcars)
```

Figure 2: Abb. 1 Speichermedien aus der Zeit "vor dem USB-Stick". (Foto: Frank Siegmund).

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

Oben, im YAML-Header dieses Textes, haben wir eingestellt "cold-fold: true". Wirkung: beim Schreiben in Quarto sehe ich den vollen Code, aber nach dem Rendern sieht man im Text nur den Hinweis ">Code", aber nicht den Code. Erst wenn man mit der Maus auf diesen Hinweis zeigt und klickt, wird der R-Code ausgeklappt und sichtbar. "code-fold: false" ist das Gegenteil, und die Voreinstellung von Quarto.

Wir ermitteln die Größe des Datensatzes, indem wir mit R nach der Anzahl der Zeilen und Spalten fragen:

```
# Abruf Anzahl der Fälle:
nrow(mtcars)
```

```
[1] 32
```

```
ncol(mtcars) # Abruf Anzahl der Variablen
```

```
[1] 11
```

@Quarto: Beim Schreiben sehen wir den R-Code als sog. “Chunk”. Ein Chunk beginnt und endet mit drei “‘. Dem einleitenden ““‘” folgt ein {r} um zu Deklarieren, dass in diesem Chunk R-Code eingebettet ist - schließlich wäre auch z.B. Python-Code möglich. Es folgt der gewohnte R-Code. Sobald Text als Chunk deklariert ist, werden beim Schreiben in Quarto rechts oben mit zwei grünen Symbolen Optionen eingeblendet, was mit dem R-Code geschehen soll. Der nach rechts gerichtete kleine grüne Pfeil sagt “bitte führe den Code in diesem Chunk aus”. Man kann also wie gewohnt Coden; diesen Code wie gewohnt auch kommentieren (wie im obigen Beispiel, mit “#”); Text rund um den Code-Bock schreiben, wie es hier an dieser Stelle geschieht. Der kleine grüne Pfeil ermöglicht es, seinen Code während des Text-Schreibens schrittweise aufzubauen und zu testen.

Wir wollen uns in einem Histogramm den Benzin-Verbrauch in Litern/100 km ansehen. Da “mtcars” ein Teil von R ist, ist der Datensatz auch in R dokumentiert, so dass man mit der üblichen Hilfefunktion von R Informationen über ihn abrufen kann. So erfahren wir im “Help” von RStudio (Reiter im Fenster unten rechts), die Variablennamen und deren Bedeutung. Für unseren Darstellungswunsch rechnen wir aus “mpg”: 235/mpg = Liter/100 km.

```r
mtcars$Verbrauch <- (235/mtcars$mpg)
summary(mtcars$Verbrauch)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  6.932  10.307  12.240  12.743  15.236  22.596 
```

… und sind bereit für das gewünschte Histogramm:

```r
hist(mtcars$Verbrauch,
 breaks=seq(4, 26, 2),
 col='steelblue',
 main='Verbrauch in l/100km',
 xlab='l/100km',
 ylab='Anzahl')
```

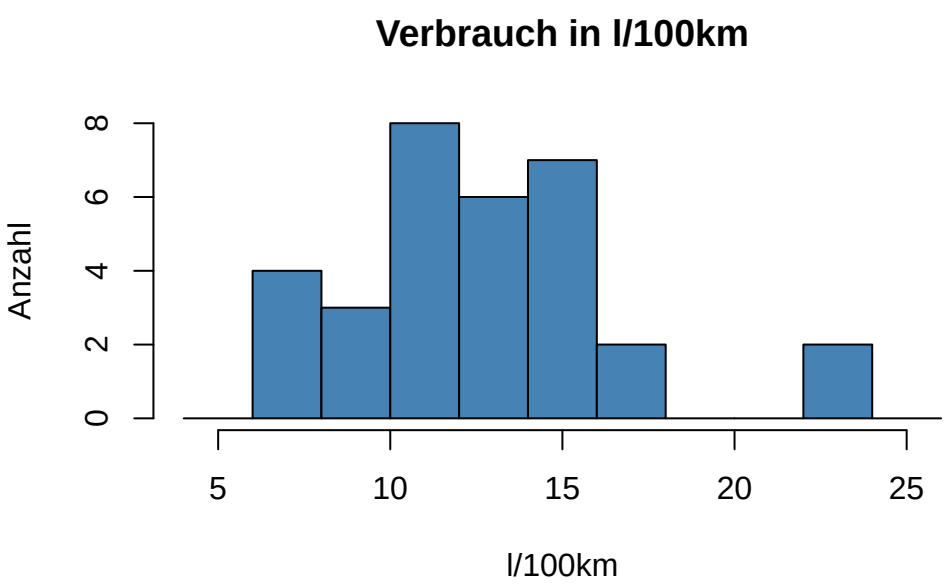

Figure 3: Kraftstoff-Verbrauch in Litern auf 100 km.

Damit ist gezeigt, wie via Quarto R-Code und dessen Ausgabe miteinander verknüpft werden können. Ideal z.B. für Lehrbücher oder Forschungsarbeiten, in denen Code und die Resultate im engen Nebeneinander gezeigt werden sollen.

Nach dem Rendern in eine HTML-Ausgabe erscheint beim "Hoovern" (Ziehen des Mauspfeils über einen Text / eine Fläche) über dem Chunk rechts oben im Chunk eine Notiztafel. Draufklicken: der Code ist kopiert und kann nun per Paste z.B. in Ihre R-Console hineinkopiert werden.

@Quarto: Wir haben bislang die Standardeinstellungen benutzt, was mit dem Chunk zu tun ist: Er soll gezeigt werden und auch der Output soll in dem gerenderten Dokument sichtbar sein. All das kann gesteuert werden, (a) global für das ganze Dokument, und zwar im YAML-Header, und (b) lokal für den jeweiligen Code-Block, womit die globalen Einstellungen im YAML-Header für diesen Chunk überschrieben werden.

Die generellen Einstellungen, die im ganzen Text wirksam sein sollen, kann man manuell setzen oder über ein Menü: im Modus "Source" sitzt über dem Editierfenster neben dem Render-Symbol ein weißes Zahnrad: klicken und auswählen.

Die lokalen Einstellungen, also im Chunk selbst, werden dort mit der Zeichenfolge "#|" eröffnet, z.B. als #| echo: false.

Folgende Optionen werden oft verwendet:

- echo - zeige den R-Code auch im Output.
- output - zeige das Ergebnis des Codes auch im Output.
- warning - zeige auch die Warnungen von R im Output. In der Regel setzt man dies auf "false".
- error - zeige auch die Fehlermeldungen im Output - was in der Wirkung auch das Rendern an jeder Fehlerstelle abbricht. In der Regel setzt man dies auf "false".
- include - nimm alles, Code wie auch Ergebnisse, beim Rendern aus dem Output heraus.

Die Fülle weiterer Befehle ergibt sich aus den Befehlen, mit denen "knit" gesteuert wird. Eine umfassende Anleitung findet sich z.B. dort.

"YAML-Header": steuert die Ausgabe

Der sog. YAML-Kopf steuert die Ausgabe der Quarto-Datei. Es ist jener Teil einer Datei, die stets am Anfang steht und mit drei Bindestrichen eingeleitete und mit drei Bindestrichen abgeschlossen wird. Dieser Teil ist sehr mächtig, aber auch kompliziert. Doch Quarto macht den Anfang leicht: Wer in RStudio links oben über "File" => New File => Quarto Document geht, erhält eine Schaltfläche eingespiegelt, in denen die wichtigsten Einstellungen für ein übliches Projekt abgefragt werden resp. voreingestellt sind. Wer später komplexere Bedarfe hat, muss sich über YAML-Hilfen tiefer orientieren, z.B. dort.

Der YAML-Header ist Teil des *.qmd-Dokuments, in den gerenderten Fassungen, d.h. der Ausgabe des Dokuments z.B. als HTML oder *.docx, nicht mehr sichtbar. Zur Anschauung daher nachfolgend zwei YAML-Header. Zunächst eine Minimal-Fassung, um später ein einfaches MS-Word-Dokument ausspielen zu können. Da wir an dieser Stelle den YAML-Header nicht als auszuführenden Code verstanden wissen wollen, betten wir ihn hier als nicht-auszuführend ein:

---

title: "Aufsatztitel"

author: "Alfred E. Neumann"

format: docx

editor: visual

Nun jener YAML-Header, der diesem Text zugrundeliegt:

---

title: "Kurze Einführung in Quarto"

subtitle: “Eine Arbeitsumgebung zum Coden in R und Schreiben in Markdown”

author: “Frank Siegmund”

format:

> html:
>
> > self-contained: true
> >
> > toc: true
> >
> > theme: cosmo
> >
> > fig-cap-location: bottom

knitr:

> opts_chunk:
>
> warning: false
>
> message: false

editor: visual

license: CC BY

Hinweis: Die Einrückungen innerhalb des YAML-Headers - hier mit Hilfe von “>” simuliert - erzeuge man stets mit “Feste neue Zeile” & “TAB”, nie mit Leerzeichen.

Lange Dokumente: ein Buch schreiben

Ab einer gewissen Länge wird das Arbeiten in einem Dokument mühsamer; wer ein Buch schreibt, kann und sollte in Quarto (wie ja auch in üblichen Textverarbeitungen) kapitelweise vorgehen und das Buch in mehrere getrennte Dokumente / Kapitel zerlegen. Ein Masterdokument hält dann alles zusammen, d.h. führt die Kapitel in der gesetzten Reihenfolge zusammen. Dies ermöglicht es, in einzelnen Kapiteln zu arbeiten, diese separat zu rendern, aber ggf. auch über das Masterdokument auch das gesamte Buch zu rednern. Das Masterdokument wird via RStudio als Buch-Projekt angelegt und führt in seinem YAML-Header in der Anweisung “chapters:” alle Kapitel / Dokumente auf, die zum Buch zusammengebunden werden sollen. Näheres in der Quarto-Dokumentation, dort und dort.

Weiterführendes

- Die umfassende Dokumentation von Quarto, dort.
- Das Quarto-Paket auf CRAN, inkl. Reference Manual, dort.
- "R Quarto Tutorial – How To Create Interactive Markdown Documents" (R-bloggers, 28.07.2022), dort.
- Blog "A Quarto tip a day": dort. Auch via Twitter: #quartotip
- Mickaël Canouil: "Awesome Quarto" dort - Umfassende Liste von Quarto-Dokumentationen, Hilfen, Beispielen etc.
- Mock, T. (2022). Welcome to Quarto Workshop!YouTube, 2:22 Std. dort /und/ dort
- Rapp, A. (2022). The ultimate guide to starting a Quarto blog. dort.
- Da, wo die Quarto-Dokumentation für den YAML-Header und die Chunks noch Fragen offen lässt, kann ein Blick in Dokumentationen zu R Markdown oder Bookdown hilfreich sein. Ein guter Ansatzpunkt ist: Xie, Y. (2022). bookdown: Authoring Books and Technical Documents with R Markdown, dort.

Viel Freude!

… beim weiteren Entdecken und Arbeiten mit Quarto. Wie gesagt, ich bin ein Lernender. Hinweise zu Fehlern, Verbesserungen oder zum Ausbau dieser Einführung? Gerne! mail@frank-siegmund.de