Sophie C. Schmidt

Den Überblick behalten – Organisation von Daten, Analysecode und Text in Paketen

Abstract In a time of increasingly computer-assisted research, it is important to create research processes that are as reproducible as possible. On the one hand, this allows researchers to keep track of their own procedures, and on the other hand it already prepares data and methods for publication. This contribution introduces the concept of the executable document and illustrates its use via an example of implementing statistical analyses in the R scripting language within a *Quarto* document. The R package *rrtools* by Ben Marwick et al. facilitates the creation of a shareable "folder package" of directly executable analysis scripts. It specifies a standardised folder structure for storing various files and guides users through the creation of important metadata. The exemplified workflow promotes open, reproducible and replicable science.

Zusammenfassung In der zunehmend computergestützten Forschung gilt es, Arbeitsschritte möglichst reproduzierbar zu gestalten. So kann einerseits der eigene Überblick über die Prozesse behalten und andererseits die Veröffentlichung von Daten und Methoden vorbereitet werden. Dieser Beitrag stellt das Konzept des ausführbaren Dokuments vor und erläutert seinen Nutzen am Beispiel der Umsetzung statistischer Analysen in der Skriptsprache R innerhalb eines *Quarto*-Dokuments. Das R-Paket *rrtools* von Ben Marwick u. a. erleichtert die Erstellung eines teilbaren „Ordnerpakets" von direkt ausführbaren Analyseskripten. Es gibt eine standardisierte Ordnerstruktur für die Ablage diverser Dateien vor und führt die Nutzer durch die Erstellung wichtiger Metadaten. Damit wird der offenen, reproduzierbaren und replizierbaren Wissenschaft Vorschub geleistet.

Keywords Offene Wissenschaft, Skriptsprache R, ausführbare Dokumente, Reproduzierbarkeit

Schmidt, Sophie C. (2026): Den Überblick behalten – Organisation von Daten, Analysecode und Text in Paketen, in: Bolder-Boos, Marion – Hageneuer, Sebastian – Pantelidis, Georg (Hrsg.), *Digitale Kompetenzen in den archäologischen Disziplinen. Perspektiven für die Lehre*, Heidelberg: Propylaeum, S. 129–139, [online] https://doi.org/10.11588/propylaeum.1773.c26533.

Einleitung

Forschung ist ein langwieriger, iterativer Prozess, in dem mit unterschiedlichen Datentypen, Zwischenergebnissen, Textbausteinen, Code-Schnipseln, Grafiken und Quellen hantiert werden muss, ohne den Überblick zu verlieren. Aus diesen Einzelteilen werden Narrative geschmiedet, die in der Regel als Artikel oder Bücher veröffentlicht werden. Dabei ist die Offenlegung dieses wirren Prozesses mit der zunehmenden Quantifizierung und der Einbindung komplexerer computergestützter Methoden eher schwieriger geworden. Während die Forschenden in den 1960er und 70er Jahren häufig noch rein visuell arbeiteten, ergo zum Nachvollziehen der Ergebnisbildung v. a. ein Katalog notwendig war, setzte in den 1970ern eine bis heute andauernde wissenschaftstheoretische Phase ein, die computergestützte Datenverarbeitung und statistische Methoden als modern und wissenschaftlich ansieht. Mit der zunehmenden Digitalisierung sind ganz neue Probleme verbunden, derer sich Forschende bewusst sein sollten.

Eines dieser Probleme ist vom Prinzip nur, „den Überblick zu behalten“. Dieser Überblick sollte nicht nur während des Schreibprozesses des einen Artikels gegeben, sondern auch nach mehreren Jahren herstellbar sein. Es gilt, möglichst reproduzierbar zu arbeiten, d. h., die Informationen so vorzuhalten, dass das zukünftige Selbst und andere Menschen mit den gleichen Daten und Methoden zum gleichen Ergebnis kommen (The Turing Way Community 2025) **(Abb. 1)**. Wenn Analysen reproduzierbar zur Verfügung stehen, sind die Wege zur Replizierbarkeit auch erleichtert, weil die Methodik auf eine Art und Weise bereitgestellt wird, dass sie genauso mit anderen Daten nachvollzogen werden kann.

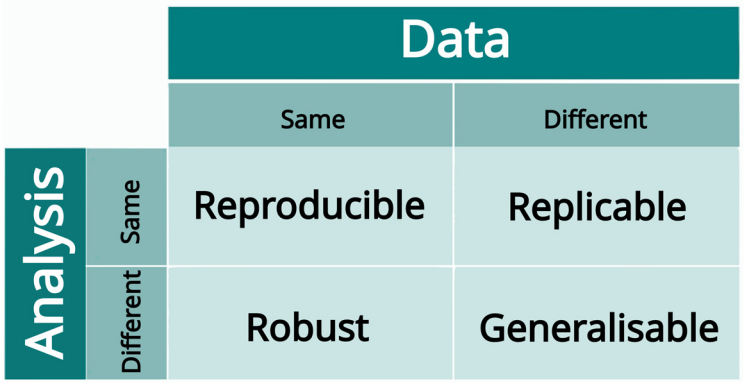

Abb. 1: Definitionen des Projekts „Turing Way“ für reproduzierbare, replizierbare, robuste und generalisierbare Analysen (CC BY 4.0 The Turing Way Community 2025).

Werden Daten so bereitgestellt, dass andere Forscher*innen sie mit anderen Methoden bearbeiten können, wird es einfacher, die Robustheit der Schlussfolgerung zu testen: Werden mit neuen Methoden die gleiche Ergebnisse erzielt, können diese als belastbar angesehen werden. Neben der Nützlichkeit solch eines Vorgehens allgemein, können Fehlerquellen bei der Übernahme fremder Daten vermieden werden, neue

Methoden schneller Verbreitung finden und auch leichter von Kolleg*innen überprüft werden. Damit führen Praktiken der offenen Wissenschaft zu einer Beschleunigung und Verbesserung von wissenschaftlichem Fortschritt (Touchon – McCoy 2016; Marwick et al. 2017).

Eine Möglichkeit in der quantitativen Forschung, die Methodik Kolleg*innen zugänglich zu machen, ist die Publikation von skriptbasierten Abläufen. Skriptbasiert bedeutet, dass entweder die Analyse selbst in einer Programmier- oder Skriptsprache geschrieben wird oder, falls ein *Graphical User Interface* (GUI) genutzt wird, die Analyse als Code ausgegeben werden kann. Eine Möglichkeit ist, Code-Blöcke mit Artikeln zusammen abzudrucken (z. B. Roth 2025). Das ist hilfreich, jedoch in seiner Nützlichkeit beschränkt: Nachnutzer*innen müssen nicht nur das Programm und die benötigten Pakete nachinstallieren, sondern auch zusätzlich den Code abtippen oder über OCR-Verfahren umständlich in digitalen Text umwandeln. Zudem muss die Zuordnung einzelner Codeblöcke zu den jeweiligen Analyseergebnissen oder Grafiken dargelegt werden.

Einfacher ist es, den Analysecode und Daten mithilfe eines „ausführbaren Dokuments“ zur Verfügung zu stellen. Der hier vorgestellte Ansatz basiert zum Großteil auf Arbeiten von Ben Marwick und Kolleg*innen, die die Reproduzierbarkeitsdebatte aus anderen Fächern in die Archäologie getragen haben, und sich mit unterschiedlichen Artikeln für offene Methoden einsetzen (Marwick et al. 2017; Marwick et al. 2018; Marwick – Birch 2018).

Das ausführbare Dokument

„Ausführbar“ heißen Dokumente, die wie ein kleines Programm/ein Skript agieren, sodass dabei ein Artikel entsteht, der Text, Bilder und Bibliographie enthält. Man hält damit den Text sowie den Code, den man für die Erstellung der Grafiken und Analyseergebnisse benötigt, zusammen in einem Dokument vor. Dies erleichtert die Auffindbarkeit und Zuordnung von Ergebnissen zum jeweiligen Code.

Grundlagen für diese ausführbaren Dokumente, auf die im Folgenden eingegangen werden sollen, sind:

1. Markdown
2. *Quarto*
3. Code-Blöcke
4. der Umwandlungsprozess und
5. Pakete.

Diese werden nachfolgend vorgestellt und im Anschluss ein R-Paket beschrieben, das die Erstellung ausführbarer Dokumente und Pakete erleichtert.

Markdown

Markdown (md abgekürzt und als Dateiendung) ist die Grundlage für den Textteil des Dokuments. Es handelt sich um eine Auszeichnungssprache, die von John Gruber 2004 entwickelt wurde: Indem in einer simplen Textdatei einer bestimmten Syntax gefolgt wird, kann die Datei von einem Programm im Hintergrund in formatierte Dateien (z. B. *Word*, html, pptx oder pdf) umgewandelt werden.

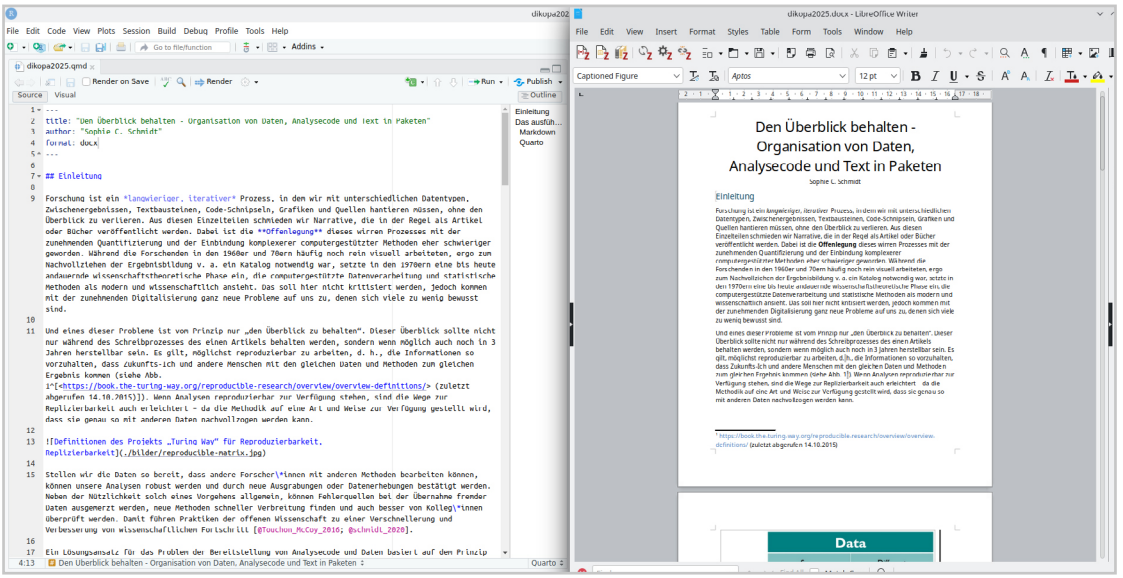

Abb. 2: Beispiel für die Syntax von Markdown in einem *Quarto*-Dokument und der ausgeführten Datei (CC BY 4.0 S. C. Schmidt).

Zum Beispiel lassen sich Überschriften mit dem Rautensymbol markieren (hierarchisch gestaffelt – eine Raute # ist die höchste Ebene, ## zeichnet die Ebene darunter aus, ### die dritte Ebene), Listen mit einem einfach Minus-Symbol und Text über Sternchensymbole *kursiv* und **fett** drucken (siehe **Abb. 2**). Fußnoten, Bilder, Links, Rückverweise auf Überschriften (mithilfe von „Ankern“) genauso wie auf Bilder (über *label*) und Literaturverweise lassen sich ebenso einbinden.[1] Damit entstehen Dokumente, die menschen- und maschinenlesbar sind.

Es gibt einen weiteren Vorteil: Das Versionskontrollsystem *Git*[2] arbeitet in einem Markdown-basierten Dokument zeilenbasiert, in einem *Word*-Dokument jedoch nur

1 Siehe z. B. https://www.markdownguide.org/cheat-sheet oder https://rstudio.github.io/cheatsheets/quarto.pdf [zugegriffen am 01.04.2026].

2 Siehe Maria Shinoto in diesem Band.

auf Ebene des gesamten Dokuments. Markdown-Dateien lassen sich damit kleinteiliger versionieren als *Word*-Dokumente und Veränderungen zwischen Versionen sind leichter zu erfassen.
Markdown kommt in unterschiedlichen Kontexten vor und damit in leicht unterschiedlichen „Geschmacksrichtungen“, die die Markdown-Syntax etwas erweitern oder abwandeln. Für das Schreiben wissenschaftlicher Dokumente hat sich inzwischen *Quarto* etabliert.

Quarto

Die Abkürzung und Endung eines *Quarto*-Dokuments ist qmd. Das qmd-Dokument benötigt eine gewisse Struktur, die der umwandelnden Software Anweisungen gibt. Das Dokument beginnt mit einem sogenannten „YAML-Header“. Er beginnt und endet mit drei Minuszeichen in einer eigenen Zeile. Die Struktur in dem YAML-Header kann leicht anders aussehen, je nachdem welche Konvertierungssoftware man benutzt, hier ist es für *Quarto* dargelegt. In dem Header werden Anweisungen und Metadaten zu dem Dokument abgelegt (siehe **Abb. 2**):

- Der Titel wird hinter `title:` in Hochkommas geschrieben
- und mit `format:` wird spezifiziert, in was für eine Art Dokument das qmd-Dokument verwandelt werden soll. Es gibt „html“, „docx“ oder „pdf“ und tatsächlich noch einige andere Versionen, deren Erläuterung hier zu weit führen würde. Man kann auch auf eine docx-Datei als Vorlage („template“) verweisen: Diese Textdatei bestimmt dann das „Design“, in das die Markdown-Datei konvertiert wird.
- Die Literaturverwaltung läuft über `bibliography:` Hier wird auf eine bib-Datei, also die Literaturdatenbank, verwiesen. Mit labels aus dieser Datenbank wird auf die einzelnen Literaturhinweise verlinkt.
- Die *citation style language* (csl)-Datei definiert die Formatierung der Bibliographie (sowohl im Text als auch für das Literaturverzeichnis). csl-Dateien werden inzwischen für viele unterschiedliche Formatierungsvorgaben zur Verfügung gestellt. Sie lassen sich im Style-Repositorium von *Zotero* finden.[3]

Die Nützlichkeit dieser Art des ausführbaren Dokuments liegt aber nicht nur darin, einen Text zu formatieren, sondern v. a. darin, dass die Analyse innerhalb des gleichen Dokuments abgelegt werden kann.

3 Siehe https://www.zotero.org/styles [zugegriffen am 14.10.2025].

Code-Blöcke („Code Chunks“)

Durch das Markieren von Bereichen mit drei Hochkommas (`) am Anfang und am Ende entsteht ein *Code Chunk* oder Code-Block: In den geschweiften Klammern wird angegeben, in welcher Programmiersprache der Code geschrieben ist und dann lässt sich damit und dort z. B. eine Grafik erstellen oder ein Wert berechnen.

Die Grafik kann eine Bildunterschrift erhalten und ein *label*, welches genutzt werden kann, um auf diese Grafik zu verweisen. Der Wert lässt sich als Variable im Text aufrufen.

Der Vorteil liegt auf der Hand: Sollte sich etwas in der Datenbasis oder der Analyse verändern, können die berechneten Werte und Grafiken innerhalb des Textes automatisch angepasst werden. Auch dies reduziert eine Fehlerquelle. Zudem ist dies genau der Teil, der Kolleg*innen eventuell besonders interessiert: Wie genau die Analyse berechnet wurde, kann mithilfe dieses Dokuments geteilt werden und benötigt weniger erklärende Worte als eine Analyse ohne publizierten Code.

Umwandlungsprozess der qmd-Datei

Das gewünschte Dokument entsteht dann erst „auf Knopfdruck“ bzw. Eingabeaufforderung: Als ersten Schritt werden die Code-Blöcke ausgeführt und die qmd–Datei mithilfe von *Jupyter* (bei Python-Code) oder *knitr* (bei R-Code) in eine Markdown-Datei umgewandelt – dabei entstehen schon alle Grafiken, die jetzt in der md-Datei nur noch als Bilder eingebunden werden. Diese Markdown-Datei wird schließlich mithilfe von *pandoc* in das gewünschte Format transformiert, also pdf, *Word* (docx), html, oder ein anderes (siehe **Abb. 3**)[4]. In dem Beispiel **Abb. 2** ist die Datei in eine *Word*-Datei verwandelt worden.

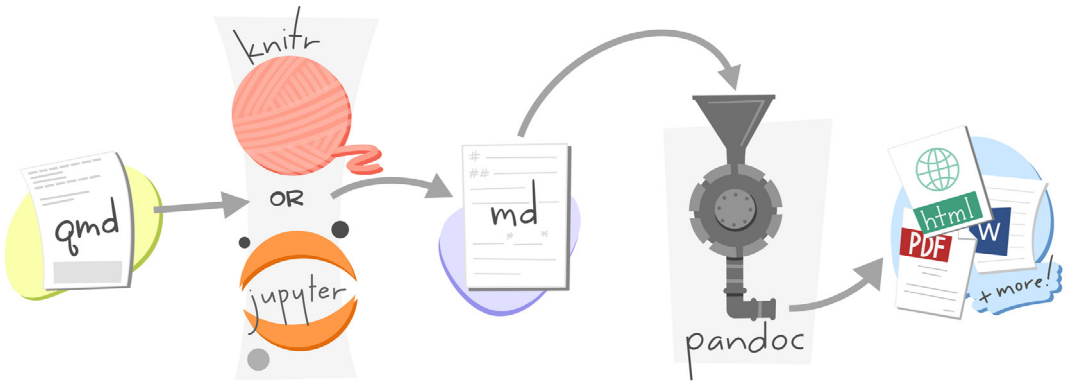

Abb. 3: Umwandlungsprozess der QMD in das gewünschte Dokument (Illustration CC BY 4.0 A. Horst für *RStudio*).

4 Illustration von Allison Horst, für *RStudio*, zu finden auf https://allisonhorst.com [zugegriffen am 14.10.2025].

Der Gestaltungsfreiraum ist sehr groß. Es lassen sich im YAML-Header diverse Variablen setzen, für Bilder z. B., wo sie abgelegt oder mit wie viel DPI sie gespeichert werden. Für Bücher oder längere Arbeiten können in einem übergeordneten Dokument mehrere Kapitel als einzelne qmd-Dateien zusammengeführt werden. Über eine Programmierschleife lassen sich auch unterschiedlich parametrisierte Ausgaben berechnen, d. h. aus einem qmd-Dokument werden mehrere PDFs, in denen z. B. die zugrundeliegenden Datensätze anders gefiltert wurden. Damit können automatisch mehrere Berichte aus einem Dokument und einem Datensatz erstellt werden.

Um auf die Reproduzierbarkeit zurückzukommen: Jede Person, die dieses qmd-Dokument und nicht nur das fertig umgewandelte Dokument bekommt, kann damit nachvollziehen, welche Berechnungen genutzt wurden, um auf den besprochenen Wert zu kommen oder eine Grafik zu erstellen. Mithilfe der zugrundeliegenden Daten und dieses Dokuments sollte jede*r in der Lage sein, genau die gleichen Ergebnisse zu erzielen.

Nun müssen aber diese Daten und ihre Ablage besprochen werden. Um die Daten gemeinsam mit dem qmd-Dokument abzulegen, lohnt es sich, ein Paket zu erstellen.

Pakete schnüren

Ein Paket ist nicht viel mehr als ein Ordner mit Unterordnern für Rohdaten und bearbeitete Dateien, Abbildungen und Grafiken sowie den zusätzlichen Dateien, die für die Umwandlung der qmd notwendig sind, wie die csl- und bib-Datei.

Wenn innerhalb der qmd dann nur relative Pfade verwendet werden, d. h. Pfade, die den Weg von der qmd-Datei zu der gewünschten Datei beschreiben und nicht von der Partition „home/“ oder „C:“ aus, dann ist mit der Übergabe des Pakets an jemand anderes oder der Übertragung auf einen neuen Computer alles an einem Ort, um die Analyse erfolgreich durchzuführen.

Um diesen Prozess für R-Nutzer*innen zu vereinfachen und zu automatisieren, haben Ben Marwick und Kolleg*innen das *rrtools*-Paket[5] entwickelt (Schmidt – Marwick 2020), das die Nutzer*innen mit vier zentralen Funktionen durch die Entwicklung eines Pakets für reproduzierbare Analysen führt. Ein R-Paket ist letztendlich eine kleine Erweiterung der grundlegenden Funktionalitäten der Skriptsprache R (R Core Team 2023). Es gibt sehr viele dieser Erweiterungen, die über das *Comprehensive R Archive Network* (CRAN)[6], häufig auch über *GitHub* oder andere Plattformen, bereitgestellt werden. Diese Pakete sind immer in einer bestimmten Art und Weise aufgebaut, um problemlos von anderen Nutzer*innen installiert werden zu können.

---

5 *rr* für *reproducible research*, https://github.com/benmarwick/rrtools [zugegriffen am 01.04.2026].

6 https://cran.r-project.org [zugegriffen am 14.10.2025].

Das R-Paket rrtools

Indem die Funktionen in *rrtools*, die im Folgenden erklärt werden, den Vorgaben für die Entwicklung von R-Paketen folgen, ist das entstehende Paket bei anderen Nutzer*innen installierbar. Damit können sie die für das Paket oder die Analyse entwickelten Funktionen nutzen und das Abhängigkeitsmanagement sollte geklärt sein. Abhängigkeitsmanagement bedeutet, dass, wenn in einem Paket auf andere Erweiterungen (Pakete) zurückgegriffen wird, diese deklariert und ebenfalls installiert werden müssen.

rrtools bietet auch das automatische Erstellen einer MIT-Lizenz an. Es ist vorgesehen, dass das Paket mit *Git* versioniert und über *GitHub* synchronisiert wird. Es führt Nutzer*innen durch die Erstellung einer Docker- und einer renv-Umgebung. Darauf wird hier nicht im Detail eingegangen. Es dient dem Abspeichern der jeweils genutzten Paketversion von importierten Paketen, damit der Code auch nach Updates oder auf fremden Systemen mit anderen Paketversionen läuft.

Hier wird kurz die Funktionsweise erklärt:

Erstellung des Pakets

Mit der grundlegenden Funktion `use_compendium` wird ein Paket angelegt. *RStudio* öffnet ein neues Fenster, in dem erklärt wird, was diese Funktion gerade geleistet hat: Es hat Dokumente erstellt, wie die DESCRIPTION-Datei, welche das Paket definiert, eine NAMESPACE-Datei, die RProj-Datei und noch einige mehr. Als letztes erklärt es dann „Next, you need to do“ und führt damit durch die Funktionsweise der einzelnen Dateien und notwendigen Schritte.

Readme

Unter anderem wird der/die Benutzer*in aufgefordert mit use_readme_qmd, eine README und andere Dateien als Vorlagen zu erstellen, die dann noch modifiziert werden sollen. Eine README ist bei *GitHub* und anderen Systemen die erste Datei, die angezeigt wird und allgemein darüber informiert, wofür dieser Ordner da ist und welche Dateien sich in ihm befinden.[7]

Ordnerstruktur

Die Kernfunktion, die die Ordnerstruktur aufbaut, ist `use_analysis`. Hiermit werden die Ordner und Vorlagen angelegt, die für ein Projekt genutzt werden können. Auch hier führen Hinweise, was als nächstes getan werden sollte, durch das Vorgehen. Es gibt zudem einen Warnhinweis, dass die Daten automatisch mit *Git* versioniert werden. Wenn man also nicht möchte, dass auch Daten in *GitHub* hochgeladen werden,

7 *read me* – Die Aufforderung, diese Datei als erstes zu lesen, steckt im Namen.

müsste der data-Ordner zum *gitignore* hinzugefügt werden. *Gitignore* ist eine einfache Textdatei, in der aufgelistet wird, welche Dateien in dem durch *Git* versionierten Ordner NICHT mit *Git* verfolgt werden sollen.

Lizenzieren

Die letzte Funktion, die hier noch erwähnt sein soll, ist `use_mit_licence`. Wenn Material online geteilt wird, ist eine Lizenzierung wichtig, die klärt, wie die geteilten Sachen zitiert werden müssen, aber auch, wie sie genutzt werden dürfen (Marwick – Birch 2018, 6). Als Vorlage ist hier die MIT-Lizenz eingestellt, da die einerseits sehr permissiv ist, andererseits keine Haftung für eventuell fehlerhaften Code übernommen wird.

Zusammenfassung

So entsteht ein Paket, das den Code und Text zusammenhält und alle notwendigen Dateien in einer sinnvollen Struktur ablegt. Die Nutzung von *Git* ermöglicht eine Versionskontrolle, die Anleitung für das Anlegen von Metadaten und der Lizenz sowie dem Abhängigkeitsmanagement ein gut mit anderen teilbares Paket, das auf unterschiedliche Arten und Weisen zur Verfügung gestellt werden kann. Zu empfehlen ist es, das Paket in einem vertrauenswürdigen Repositorium abzulegen, und die dort generierte persistente URL des Pakets in dem Artikel, der veröffentlicht wird, mit abzudrucken. Es ist nicht nachhaltig, das Paket nur auf eigenen Webseiten, *GitHub*[8], academia oder researchgate hochzuladen, da diese Anbieter keine dauerhafte Bereitstellung garantieren. Geeignet sind Repositorien wie zenodo.org, Open Science Framework (OSF) oder Repositorien von Universitäten, die eine persistente Ablage versichern.

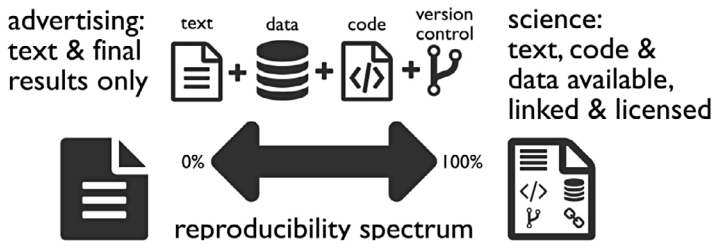

Abb. 4: Stufen der Reproduzierbarkeit (CC BY Ben Marwick et al. 2017).

8 Wenn das Repositorium auf *Github* liegt, ist es sehr leicht als Version nach *Zenodo* zu veröffentlichen: https://docs.github.com/de/repositories/archiving-a-github-repository/referencing-and-citing-content [zugegriffen am 31.10.2025].

Die Ausführbarkeit des qmd-Dokumentes ermöglicht es Kolleg*innen, die durchgeführten Analysen relativ schnell und einfach nachzuvollziehen. Darin steckt ein großes Potenzial für eine offene und reproduzierbare Wissenschaft (siehe **Abb. 4**).

Das R-Paket, welches diesen Text, die Bilder, csl- und bib-Dateien bereitstellt, befindet sich unter der DOI-Adresse: https://doi.org/10.5281/zenodo.17495645.

ORCID®

Sophie C. Schmidt https://orcid.org/0000-0003-4696-2101

Referenzen

Marwick, Ben – d'Alpoim Guedes, Jade – Barton, C. Michael – Bates, Lynsey A. – Baxter, Michael – Bevan, Andrew – Bollwerk, Elisabeth A. – Bocinsky, R. Kyle – Brughmans, Tom – Carter, Alison K. – Conrad, Cyler – Contreras, Daniel A. – Costa, Stefano – Crema, Enrico R. – Daggett, Adrianne – Davies, Benjamin – Drake, B. Lee – Dye, Thomas S. – France, Phoebe – Fullager, Richard – Giusti, Domenico – Graham, Shawn – Harris, Matthew D. – Hawks, John – Heath, Sebastian – Huffer, Damien – Kansa, Eric C. – Kansa, Sarah Whitcher – Madsen, Mark E. – Melcher, Jennifer – Negre, Joan – Neiman, Fraser D. – Opitz, Rachel – Orton, David C. – Przystupa, Paulina – Raviele, Maria – Riel-Salvatore, Julien – Riris, Philip – Romanowska, Iza – Smith, Jolene – Strupler, Néhémie – Ullah, Isaac I. – Vlack, Hannah G. Van – VanValkenberg, Nathaniel – Watrall, Ethan C. – Webster, Chris – Wells, Joshua – Winters, Judith – Wren, Colin D. (2017): Open Science in Archaeology, in: *SAA Archaeological Record 17(4)*, S. 8–14.

Marwick, Ben – Boettiger, Carl – Mullen, Lincoln (2018): Packaging data analytical work reproducibly using R (and friends), [online] https://doi.org/10.7287/peerj.preprints.3192v2.

Marwick, Ben – Birch, Suzanne E. Pilaar (2018): A Standard for the Scholarly Citation of Archaeological Data as an Incentive to Data Sharing, in: *Advances in Archaeological Practice 6(2)*, S. 125–143, [online] https://doi.org/10.1017/aap.2018.3.

R Core Team (2023): R: A Language and Environment for Statistical Computing, [online] https://www.r-project.org.

Roth, Georg (2025): Not outdated yet improvable – Improving the relative Chronology of the Michelsberg Culture by combining Correspondence Analysis and Principal Curve Estimation, in: Gleser, Ralf – Scharl, Silviane – Seidel, Ute – Strobel, Michael (Hrsg.), *Neue Funde im Osten – Entstehung, Verbreitung und Charakteristik des Phänomens Michelsberg im Lichte neuer Forschungen*, S. 511–536.

Schmidt, Sophie C. – Marwick, Ben (2020): Tool-Driven Revolutions in Archaeological Science, Journal of Computer Applications, in: *Archaeology 3(1)*, S. 18–32, [online] https://doi.org/10.5334/jcaa.29.

The Turing Way Community (2025): *The Turing Way: A handbook for reproducible, ethical and collaborative research*, [online] https://doi.org/10.5281/zenodo.15213042.

Touchon, Justin C. – McCoy, Micheal W. (2016): The mismatch between current statistical practice and doctoral training in ecology, in: *Ecosphere 7(8)*, [online] https://doi.org/10.1002/ecs2.1394.