Christoph Rinne

Dokumentation und Nachnutzung von Daten – R und SQLite: ein starkes Team

Abstract This article discusses the objectives and possibilities of supplementary data for scientific publications from the perspective of reusability, interoperability and reproducibility (FAIR). To begin with, the widely used formats of tabular data with layout or text documents are briefly introduced. These are contrasted with a concept based on SQL-based data, including data definition language (DDL), in combination with the scripting language R. This concept is illustrated using three specific examples of freely available datasets, with slight variations in focus and differing levels of complexity.

Zusammenfassung Der Beitrag behandelt die Ziele und Möglichkeiten supplementer Daten wissenschaftlicher Beiträge unter der Perspektive der Nachnutzbarkeit, Interoperabilität und Reproduzierbarkeit (FAIR). Einleitend werden knapp die weitverbreiteten Formate der Tabelle mit Layout und des Textdokumentes vorgestellt. Diesen wird ein Konzept gegenübergestellt, das auf SQL-basierten Daten einschließlich der Strukturanweisungen (DDL) in Kombination mit der Skriptsprache R basiert. Dieses Konzept wird anhand von drei konkreten Beispielen frei verfügbarer Datensätze veranschaulicht, die sich in ihrem Schwerpunkt und ihrem Komplexitätsgrad leicht unterscheiden.

Keywords FAIR, SQL, Skriptsprache, Markdown

Rinne, Christoph (2026): Dokumentation und Nachnutzung von Daten – R und SQLite ein starkes Team, in: Bolder-Boos, Marion – Hageneuer, Sebastian – Pantelidis, Georg (Hrsg.), *Digitale Kompetenzen in den archäologischen Disziplinen. Perspektiven für die Lehre*, Heidelberg: Propylaeum, S. 141–152, [online] https://doi.org/10.11588/propylaeum.1773.c26534.

Einführung

Ein fachspezifisch adäquater Umgang mit Forschungsdaten wird von der Deutschen Forschungsgemeinschaft (DFG) schon sehr lange gefordert (DFG 2015) und schließt ganz wesentlich die Nachnutzung der grundlegenden und generierten Daten durch Dritte mit ein.[1] Seitdem sind zahlreiche Initiativen auch im Bereich der Archäologischen Wissenschaften in Deutschland gestartet worden, so z. B. das Forschungsdatenzentrum des DAI IANUS[2]. Die jüngste und umfassendste Initiative ist die Gründung der Nationalen Forschungsdaten Infrastruktur (NFDI) mit einzelnen Konsortien, u. a. für Geistes- und Sozialwissenschaften[3]. In einem dieser Konsortien sind mit NFDI-4Objects auch die Deutschen Archäologien mit Denkmalämtern, Universitäten und außeruniversitären Forschungsinstitutionen vertreten.[4] FAIR als Akronym für *findable*, *accessible*, *interoperable*, *reproducible* oder auch nur *reproducible research* sind in diesem Kontext oft wiederkehrende Schlagworte, die ein umfassendes, neues Verständnis von wissenschaftlicher Arbeitsweise kondensieren (u. a. Marwick 2017; Jacobsen et al. 2019). Diese Begriffe sind unmittelbar verständlich und benennen die grundlegenden Anforderungen und den wesentlichen Gedanken, die wir für unserer eigene Arbeitsweise übernehmen sollten.

In dem Vortrag und dem hieraus entstandenen Text geht es darum, diese vorgenannten, überwiegend abstrakt gehaltenen Modelle zu konkretisieren. Mit der Datenhaltung, der Datenverarbeitung und Dokumentation zu Datenbeständen wird nur ein kleiner, aber auch sehr zentraler Bereich herausgegriffen. Die hier vorgestellte Arbeitsweise wird mit Fallbeispielen konkret dargestellt. Diese entstammen der eigenen Arbeit und sind nur ein möglicher Weg, Dritten den Zugang zu den eigenen Daten zu erleichtern. Die hierbei verwendeten Dateiformate und Programmiersprachen sind Markdown (md, rmd), R und SQL. Eine zentrale Software ist *RStudio*[5] von posit, die alternativ auch durch *Quarto*[6] ersetzt werden kann.

Als Teil des zweiten Workshops des DiKopA-Projektes, der auf die Teilnahme von Studierenden ausgerichtet war, wurde einleitend auch der eigene Weg zu diesem Thema als knappe Vorstellung des Autors integriert. Dieser Aspekt soll hier als stark verkürzter Rückblick auf die Anfänge von online verfügbaren Daten in der Archäologie eingefügt werden, wozu die Radiokarbondatenbank RADON[7] und das

1 https://www.dfg.de/resource/blob/172112/leitlinien-forschungsdaten.pdf [zugegriffen am 05.10.2025].

2 https://ianus-fdz.de [zugegriffen am 05.10.2025].

3 https://www.nfdi.de [zugegriffen am 05.10.2025].

4 https://www.nfdi4objects.net [zugegriffen am 05.10.2025].

5 https://posit.co/download/rstudio-desktop [zugegriffen am 05.10.2025].

6 https://quarto.org [zugegriffen am 05.10.2025].

7 https://radonb.ufg.uni-kiel.de [zugegriffen am 05.10.2025].

Online-Journal Jungsteinsite, heute Journal of Neolithic Archaeology (JNA)[8], ohne Zweifel zählen. Der Initiator und damalige Herausgeber Dirk Raetzel-Fabian formuliert im Editorial im November 1999 folgenden, nach meinem Verständnis wegweisenden Satz[9]:

> *„Ob und wann Unmittelbarkeit, Schnelligkeit, vor allem aber allgemeine Zugänglichkeit von Information und der Vorteil, Daten in einer problemlos weiter zu verarbeitenden Form übermitteln zu können, als zentrale Werte dieser Technologie [gemeint ist das Internet (Anm. des Autors)] auch in der deutschen Archäologie intensiver als bisher genutzt werden, bleibt offen – erste Ansätze zeichnen sich immerhin bereits ab.“*

Datenquellen

Seit vielen Jahren werden für wissenschaftliche Publikationen online ergänzende Daten digital zur Verfügung gestellt. Sehr oft erfolgt dies in den Formaten von *MS Excel* (*.xlsx), *MS Word* (*.docx) oder als Portable Document File (*.pdf). Damit ist die Lesbarkeit auf vielen Systemen gegeben, doch die effiziente Nachnutzbarkeit ist bisweilen stark eingeschränkt. Dies betrifft auch Tabellen im *Excel*-Format, die vielfach zahlreiche Register und ergänzendes Layout oder aber redundante Informationen in wiederkehrenden Spalten beinhalten. Dies ist umso ärgerlicher, als der Umfang analysierter Daten in der Regel beachtlich ist und die lesende Auswertung eher die Ausnahme darstellt. Selbst bei Tabellen in *Excel* müssen für jedes Registerblatt ggf. vorhandene Tabellenbeschriftungen in der ersten Zeile, erläuternde Spaltenüberschriften mit Überlänge oder Sonderzeichen, Zeilenumbrüche in Zellen, aggregierende Zwischenüberschriften oder aggregierte Zellen entfernt werden, um eine effiziente Nachnutzung zu ermöglichen.

Deutlich effizienter und schlichter sind textbasierte Formate, z. B. *comma-separated values* (*.csv). Auch sind Erläuterungen und ergänzende Informationen notwendig, dazu gehören mindestens der Zeichensatz (z. B. UTF-8), das Trennzeichen (z. B. ; oder ,), die Verwendung von Anführungszeichen für Text oder auch erweiterte Informationen zu den Spaltennamen und den jeweiligen Datenformaten, z. B. der Maßeinheit. Die Integration von längeren Texten ist bei csv-Dateien nur bedingt möglich und die Dateien werden schnell unübersichtlich. Sollten darüber hinaus Relationen zwischen einzelnen Tabellen bzw. Dateien über Schlüsselfelder bestehen, sollte

8 https://www.jna.uni-kiel.de [zugegriffen am 05.10.2025].

9 http://web.archive.org/web/20000124104440/http://www.jungsteinsite.de:80/editor.htm [zugegriffen am 05.10.2025].

dies ebenfalls dezidiert dokumentiert werden. Es kommt also zu einer Gruppe von Dateien, die in einem Container (z. B. *.zip) aggregiert werden können.

Mit Blick auf die Einschränkungen bei textbasierten Dateien für umfangreichere und komplexe Datenbestände sollte in diesem Fall ein Datenbankformat verwendet werden. Auch hier sollte einem weit verbreiteten, vom Betriebssystem (*operating system*, OS) unabhängigen System der Vorzug gegeben werden. Dies ist ohne Zweifel SQLite [10]. Hier sind alle Datentabellen mit ihren Bezügen, aggregierenden Zusammenstellungen (Abfragen bzw. Sichten oder *views*) oder auch ergänzende Metadaten in eigenen Tabellen in einer einzigen Datei zusammengefasst. Diese Datei kann sowohl binär und somit ausschließlich maschinenlesbar oder aber auch als Datei mit SQL-Anweisungen (sog. *dump*) und damit zusätzlich menschenlesbar zur Verfügung gestellt werden. Letzteres bietet für die Archivierung und Nachnutzung enorme Vorteile.

Fallbeispiele

Die folgenden Fallbeispiele nutzen jeweils eine SQLite-Datenbank und *RStudio* mit einer Markdown-Datei, um auf die Daten zuzugreifen und Ausgaben in dem Markdown-Dokument, dem daraus generierten PDF oder auch der HTML-Datei zu präsentieren. Die Kommunikation zwischen der SQL-Datenbank und *RStudio* erfolgt über Anweisungen (Code), die in SQL oder R geschrieben sein können, je nach Kompetenz und Anforderung des Nutzers. Für einen Import der Daten gibt es Alternativen, so z. B. CSV oder *Excel* und zahlreiche weitere Formate. Ein Import produziert aber stets redundante Daten und es fehlt auch die Option, einfach in die Datenquelle hineinzuschreiben. Neben SQLite gibt es weitere Datenbankanbindungen in R[11]. Das *a priori* sehr allgemeine Konzept des interaktiven Arbeitens in *RStudio* bietet zahlreiche Möglichkeiten: Publikationen verfassen, Analyse- oder Erkenntnisprotokoll, Dokumentation eines Datenbestands und Modifikationen, einfache Skripte von Analysen mit den zugehörigen Daten als Supplement einer Publikation oder ein umfassendes HTML mit Erläuterungen, Code und zugehörigen Daten für die Nachvollziehbarkeit von komplexen Berechnungen. Hierzu folgt jeweils ein Fallbeispiel.

Ganz allgemein wird die Rückgabe des Codes, z. B. ein Text, eine Tabelle oder eine Grafik, im Markdown-Dokument zusammen mit der Anweisung angezeigt. Das ganze Dokument kann dann in diverse Formate mit und ohne den Code exportiert werden, so z. B. in ein PDF, eine HTML-Datei oder auch ein *Word*-Dokument. Diese Endprodukte unterscheiden sich dann im Layout und vor allem in der Funktionalität.

---

10 https://sqlite.org [zugegriffen am 05.10.2025].

11 https://solutions.posit.co/connections/db [zugegriffen am 05.10.2025].

Eine HTML-Datei bietet mehr Dynamik, z. B. das Filtern und das Scrollen in Tabellen, während ein PDF eher statisch, aber auch für den Ausdruck geeignet ist **(Abb. 1)**.

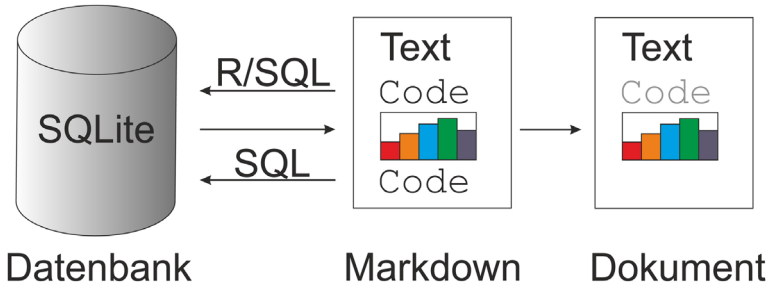

Abb. 1: Modell zur Interaktion zwischen einer SQLite-Datenbank, dem dynamischen Markdown-Dokument und dem hieraus generierten, statischen Dokument.

Datendokumentation: Fundorte an der Schlei

Das Ziel ist in diesem Fall eine knappe, einleitende Dokumentation zur Struktur und zum Umfang der Daten für den ggf. interessierten Nutzer. Die Daten sind frei zugänglich (Rinne – Nakoinz 2025). Der zugehörige Download bietet sieben Dateien, von denen nur zwei (Rmd, sqlite) hier benötigt werden. Um diese unmittelbar zu nutzen, sollten beide in einem Ordner liegen und *RStudio* durch einen Doppelklick auf die Rmd-Datei gestartet werden, da hierdurch dieser Ordner zum Arbeitsverzeichnis für *RStudio* wird.[12]

Tab. 1: Die im Download bereitgestellten Dateien.

Datei	kB	Inhalt
metadata.txt	1	Metadaten zu den Daten
nakoinz.o_2005_schlei.v00.html	1000	Dokumentation als HTML
nakoinz.o_2005_schlei.v00.pdf	351	Dokumentation als PDF
nakoinz.o_2005_schlei.v00.Rmd	14	R-Markdown-Datei für *RStudio*
nakoinz.o_2005_schlei.v00.sqlite	188	Datenbank (binär)
nakoinz.o_2005_schlei.v00.sqlite.sql	140	Datenbank als SQL-Anweisungen
readme.txt	2	Allgemeine Information zum Datenbestand

12 Alle folgenden Beschreibungen beziehen sich auf die Arbeit unter dem OS *Windows*.

Die Daten entstammen einer Publikation von Oliver Nakoinz (Nakoinz 2005) und wurden im Rahmen einer Übung zur Archäoinformatik aus einem Text mit regulären Ausdrücken strukturiert zerlegt und in einer SQLite Datenbank mittels SQL-Anweisungen reorganisiert. Der Download bietet auch die Version als SQL-Anweisungen (SQL), die nochmals deutlich kleiner, menschenlesbar und ggf. für den Import in andere Formate manipulierbar ist. Leider können weder *RStudio* noch der Datamanager von *QGIS* multiple SQL-Anweisungen verarbeiten (`beginn transaction; … commit;`). Die Grundlegende Anweisung für den Import an das Programm *sql3.exe* ist aber sehr einfach: `sqlite3 <ZielDB> <SQL-Anweisung>: sqlite3 sqlite3nakoinz.sqlite ".read nakoinz.o_2005_schlei.v00.sqlite.sql"`.

Für die Anbindung von Datenbanken muss in *RStudio* zuerst das benötigte Paket ggf. installiert und dann geladen werden. Danach wird der Zugriff auf die Datenbank einer Variablen zugewiesen. Diese Codeblöcke werden mit einer spezifischen Zeichenkombination markiert und erhalten in geschweiften Klammern spezifische Parameter, als erstes mindestens den Verweis auf die Programmiersprache und dann ggf. eine Bezeichnung **(Codeblock 1)**.

Codeblock 1

````
```{r setup}
require(RSQLite) || install.packages("RSQLite")
dbn <- dbConnect(RSQLite::SQLite(),
                 dbname = "./nakoinz.o_2005_schlei.v00.sqlite")
```
````

Nach der Definition des Datenbankzugriffs kann auf die Datenbank auch mit SQL zugegriffen werden. Ein Beispiel aus dem Vortrag liest die Tabellennamen aus der Datenbankdefinition aus und ergänzt in der ersten Spalte einen Zähler **(Codeblock 2)**. Auf ähnliche Weise kann auch die jeweilige Tabellendefinition abgefragt werden. In diesem Fall erfolgt die Anfrage in der Programmiersprache R mit der Übergabe einer SQL-Anweisung (`dbGetQuery()`). Von der resultierenden Tabelle wird auf die erste Zeile gefiltert und nachfolgend werden die komplexen `(\r\n)` durch einfache Zeilenumbrüche `(\n)` ersetzt. Abschließend wird das Ergebnis, die SQL-Anweisung der Tabellendefinition, als gut lesbarer Text mit `cat()` angezeigt **(Codeblock 3)**.

Codeblock 2

```
```{sql Tables in the sqlite database, connection = dbn}
select row_number() over (order by name) as 'nr',
      name  as 'table name' from sqlite_master
      where type = 'table' order by 1;
```
```

Codeblock 3

```
```{r Mit R Tabellenstrukturen auslesen}
sql<-dbGetQuery(dbn, "select sql from sqlite_master
                    where type = 'table' and
                          name = 'fundpunkte';")[,1]
cat(gsub('\r\n', '  \n', sql))
```
```

Für eine möglichst einfache und schlanke Gestaltung der Datenbank und der Daten werden die enthaltenen Fundpunkte als *well-known-text* (WKT) mit ergänzender Information zum Koordinatensystem (srid) zur Verfügung gestellt. Jedes GIS kann aus diesen Angaben wieder eine binäre Geometrie erzeugen. Zudem muss in dem Textfeld der WKT-Geometrie nicht nach dem Geometrietyp unterschieden werden, denn dies ist Bestandteil des Geometrietextes, z. B. POINT() **(Codeblock 4)**. Je nach Bedarf oder Vorliebe können die Daten der Datenbank sowohl in SQL als auch in R gefiltert, aggregiert oder anderweitig prozessiert werden. In einem weiteren Beispiel zur Dokumentation des Datenbestandes werden die vorhandenen Sachinformationen (hier *tag*) aggregiert und quantifiziert **(Codeblock 5)**. Vorformulierte Abfragen über die Daten werden in der SQL-Datenbank als Sichten (*views*) gespeichert und können wie die Tabellendefinitionen aus der Datenbankdefinition abgefragt und dann ausgeführt werden. Hierfür muss lediglich die einleitende SQL-Anweisung für das Erstellen einer Sicht entfernt und die verbleibende Anweisung ausgeführt werden **(Codeblock 6)**.

Codeblock 4

```
```{sql Die Daten in fundpunkte, connection = dbn}
select * from fundpunkte limit 4;
```
```

Codeblock 5

```
```{r Tabellierung der Daten in Tabelle infos}
dbGetQuery(dbn, "select nr, tag, wert from infos;") %>%
  group_by(., tag) %>%
  summarise(count = n()) %>%
  knitr::kable()
```
```

Codeblock 6

```
```{r}
sql<-dbGetQuery(dbn, "select sql from sqlite_master
                where type = 'view' and name = 'Tabelle';")[,1]
sql <- gsub('CREATE VIEW Tabelle as', '', sql)
dbGetQuery(dbn, sql)
```
```

Code als Supplement: Analyse zu neolithischen Häusern

In diesem Fall handelt es sich um eine Kombination aus kommentiertem R-Code und der zugehörigen Datenbank. Auch dies ist als Supplement zu einer Publikation frei verfügbar (Rinne 2022). Es ist eine sehr schlichte Variante, bei der es nur um die Bereitstellung der Daten und des verwendeten Codes für die Analysen und Grafiken im Text geht. Eine einleitende Dokumentation zur Struktur der Datenbank und dem Umfang der Daten wird nicht geboten. Dieses Supplement bietet über den kommentierten, schlichten Code hinaus keinen ergänzenden Service für den Nutzer **(Codeblock 6)**.

Codeblock 7

```
# ================================================
# Housesize & postdimensions of neolithic houses
# ================================================

# encoding of the script: UTF-8, please check correct representation

# Please set the working directory to the folder where the SQLite
# database "houses.sqlite" is stored.
```

```
# required packages
require (RSQLite) || install.packages("RSQLite")
require (ggplot2) || install.packages("ggpot2")
require (grid) || install.packages("grid")
library(RSQLite)
library(ggplot2)
library(grid)

#--------------------
# 1. house sizes lowlands and lower mountain range
#--------------------

con<-dbConnect(SQLite(),"houses.sqlite")

house<-dbReadTable(con, "a_housesizes")

#house <- na.omit(house)

# convert region to factor with abbreviations:
# MG: Mittelgebirgsraum, TL: Nordeuropäisches Tiefland
house$region <- factor(house$region,
                       levels = c("Mittelgebirgsraum",
                                  "Nordeuropäisches Tiefland"),
                       labels=c("MG", "TL"))
table(house$region)
```

Code als Supplement: Adult Mortality

Das folgende Beispiel ist das Supplement zu einem Artikel zur Sterblichkeit in London, bei dem in einem großen Umfang komplexe Berechnungen vorgenommen wurden (Müller-Scheeßel et al. 2024). Eine detaillierte Dokumentation des Codes ist für die Nachvollziehbarkeit der Berechnungen oder auch die Nachnutzung der Daten unabdingbar. Den Anstoß zu dieser deutlich erweiterten Dokumentation bildete das Supplement zu einer Publikation zu Bayes Analysen von Kruschke (Kruschke 2015). Dieses Supplement wird als HTML-Datei zum Download angeboten, was mit seinen dynamischen und interaktiven Optionen die eindeutig zu präferierende Version ist (Kruschke 2020). Leider wird dieses Format als Supplement in vielen Fällen noch abgelehnt und es kann nicht in Cloud-Repositorien wie *Zenodo* als primäres Dokument für die direkte Ansicht ausgewählt werden.

Die zentrale Dokumentation wird als R-Markdown erstellt und kann sowohl in HTML als auch als PDF exportiert werden. Der Code wird als R-Projekt in zahlreiche Dateien für die einzelnen Aufgaben und Abschnitte der Berechnungen zerlegt und von diesem zentralen Dokument mit `source()` aufgerufen. Die Struktur der Publikation und der rote Faden der Analyse bleiben damit nachvollziehbar. Zudem steht das gesamte R-Projekt auf *Github*[13] zur Verfügung. Ein zentraler Aspekt der Bayes-Analysen sind umfangreiche Tabellen mit statistischen Kenngrößen der zahlreichen, wiederholten Berechnungen, die in diesem Umfang nicht sinnvoll in einem druckbaren PDF dargestellt werden können. Hier sind dynamische Tabellen mit Filteroption und paginierter Darstellung (`DT::datatable`) eine hervorragende Alternative für gute Lesbarkeit bei einem vollständigen Datenbestand. Auch die Option, den Code nur auf Anforderung einzublenden, erhöht die Lesbarkeit und bietet zugleich die unmittelbare Nachvollziehbarkeit **(Abb. 2)**.

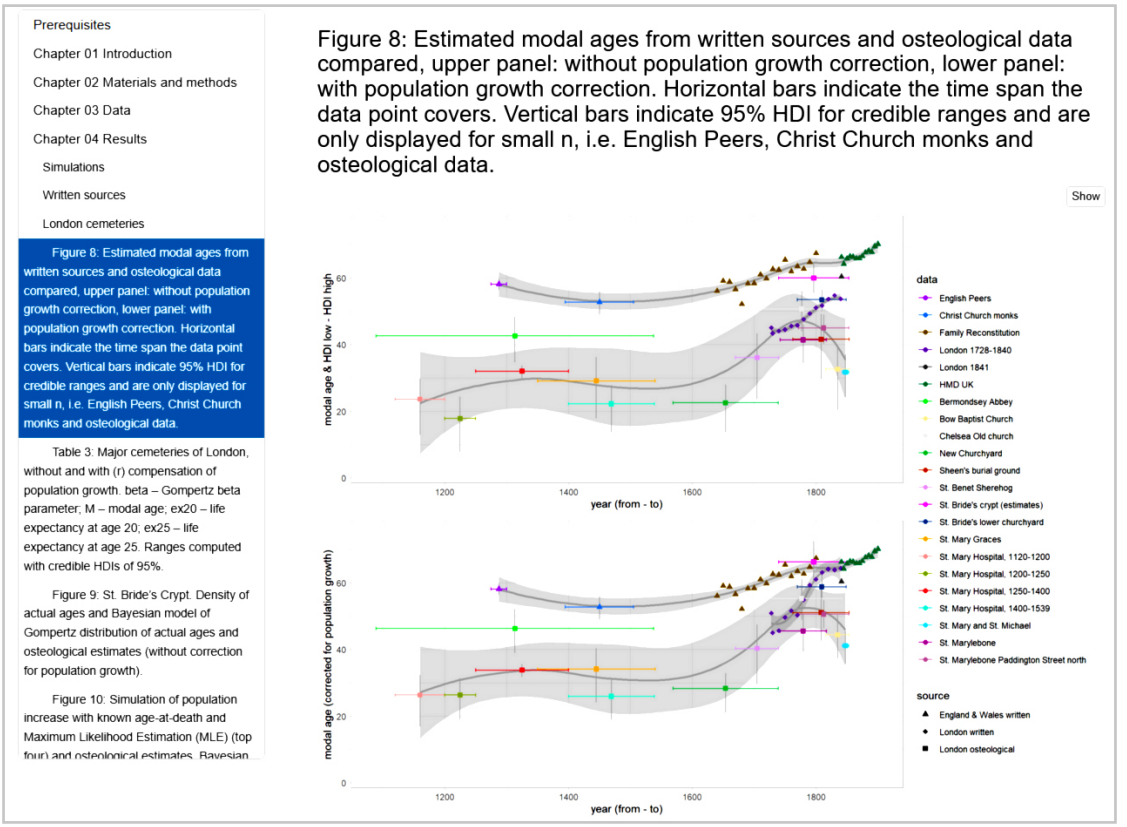

Abb. 2: Ausschnitt aus der HTML-Datei zur Dokumentation der Berechnungen in der Veröffentlichung.

13 https://github.com/ISAAKiel/london_adult_mortality [zugegriffen am 05.10.2025].

Fazit

Inzwischen stehen eine Vielzahl an Techniken und Methoden zur Verfügung, digitale Repositorien dynamisch zu verbinden und hierdurch deren Nachnutzbarkeit und die Nachvollziehbarkeit des hieraus generierten Erkenntnisgewinns deutlich zu erleichtern. Es ist damit insgesamt viel mehr möglich, als nur Daten in unterschiedlichster Form oder Dateiformaten zum Download bereitzustellen. Die angeführten Beispiele sollen das Potential dieser Arbeitsweise aufzeigen; diese sind keine konkrete Handlungsanweisung oder eine Empfehlung für die Verwendung bestimmter Software. Die über mehrere Jahrzehnte gesammelte Erfahrung des Autors bestätigt aber das kiss-Prinzip: *keep it simple, stupid.*[14] Die alternative Auflösung des Akronyms in *keep it slim & simple* würde ein wichtiges Attribut ergänzen. Textbasierte, maschinenlesbare Formate, für Daten und Anweisungen, sind für das eingangs genannte FAIR-Prinzip sicher eine gute Ausgangsbasis.

ORCID®

Christoph Rinne https://orcid.org/0000-0002-9829-6182

Referenzen

Jacobsen, Annika – de Miranda Azevedo, Ricardo – Juty, Nick – Batista, Dominique – Coles, Simon J. – Cornet, Ronald – Courtot, Mélanie – Crosas, Mercè – Dumontier, Michel – Evelo, Chris T. – Goble, Carole – Guizzardi, Giancarlo – Hansen, Karsten Kryger – Hasnain, Ali – Hettne, Kristina – Heringa, Jaap – Hooft, Rob W. W. – Imming, Melanie – Jeffery, Keith G. – Kaliyaperumal, Rajaram – Kersloot, Martijn G. – Kirkpatrick, Christine R. – Kuhn, Tobias – Labastida, Ignasi – Magagna, Barbara –McQuilton, Peter – Meyers, Natalie – Montesanti, Annalisa – van Reisen, Mirjam – Rocca-Serra, Philippe – Pergl, Robert – Sansone, Susanna-Assunta – da Silva Santos, Luiz Olavo Bonino – Schneider, Juliane – Strawn, George – Thompson, Mark – Waagmeester, Andra – Weigel, Tobias – Wilkinson, Mark D. – Willighagen, Egon L. – Wittenburg, Peter – Roos, Marco – Mons, Barend – Schultes, Erik (2020): *FAIR Principles: Interpretations and Implementation Considerations.* Data Intelligence 2(1–2), 10–29, [online] https://doi.org/10.1162/dint_r_00024.

14 https://de.wikipedia.org/wiki/KISS-Prinzip

Kruschke, John K. (2015): *Doing Bayesian data analysis: a tutorial introduction with R, JAGS, and Stan*, Amsterdam: Academic Press, [online] https://ebookcentral.proquest.com/lib/christianalbrechts/detail.action?docID=5754481 [zugegriffen am 11.11.2021].

Kruschke, John K. (2020): *Bayesian Analysis Reporting Guidelines*, [online] https://osf.io/w7cph [zugegriffen am 23.07.2024].

Marwick, Ben (2017): Computational Reproducibility in Archaeological Research: Basic Principles and a Case Study of Their Implementation, in: *Journal of Archaeological Method and Theory* 24(2), S. 424–450, [online] https://doi.org/10.1007/s10816-015-9272-9.

Müller-Scheeßel, Nils – Rinne, Christoph – Fuchs, Katharina (2024): Adult mortality in the metropolis of London 1100–1850: A Bayesian view based on osteological data, in: Am. J. Biol. Anthropol, [online] https://doi.org/10.1002/ajpa.25025.

Nakoinz, Oliver (2005): Die Schlei: Katalog der submarinen Funde und der Uferfunde. Festgabe Joachim Reichstein, in: *Offa* 59/60, S. 167–218.

Rinne, Christoph (2022): Neolithischer Hausbau. Transformationen architektonischer Kenndaten im diachronen Vergleich, in: *Journal of Neolithic Archaeology* 24, S. 1–13, [online] https://doi.org/10.12766/jna.2022.1.

Rinne, Christoph – Nakoinz, Oliver (2025): *Die Schlei: Katalog der submarinen Funde und der Uferfunde. A SQLite database from the original catalogue*, [online] https://doi.org/10.57892/100-114.