Maria Shinoto

Basiskompetenz „Text“: Neue Möglichkeiten archäologischer Arbeit

Abstract Computer work is based on text, and archaeology conveys knowledge based on text. Both types of “text” differ but can be elegantly combined digitally. New ways of working with “text” ensure verifiable, future-proof work with new possibilities for exchange and publication. This article introduces the principles of text formats and Markdown, version control and efficient collaboration, literature management, publication in different formats, and the latest tools (apps and services like *Quarto*). This overview attempts to clearly separate the fundamentally valid, timeless aspects from the tools that are subject to constant change; it is intended to be a time-independent introduction.

Zusammenfassung Computerarbeit basiert auf Text, und die Archäologie vermittelt Wissen auf der Grundlage von Text. Beide Arten von „Text“ unterscheiden sich, lassen sich aber digital elegant kombinieren. Neue Wege im Umgang mit „Text“ sorgen für überprüfbare, zukunftssichere Arbeit mit neuen Möglichkeiten des Austauschs und der Veröffentlichung. In diesem Beitrag werden die Grundlagen von Textformaten und Markdown, Versionskontrolle und effiziente Zusammenarbeit, Literaturverwaltung, die Veröffentlichung eines Textes in verschiedenen Formaten sowie die neuesten Tools (Apps und Dienste wie *Quarto*) vorgestellt. Diese Übersicht versucht, die grundsätzlich gültigen, zeitlosen Aspekte und die dem ständigen Wandel unterworfenen Werkzeuge klar zu trennen und einen möglichst zeitunabhängigen Einstieg in die Werkzeuge zu ermöglichen.

Keywords Markup, Markdown, Versionierung, Literaturverwaltung, Nachprüfbarkeit, Plattformunabhängiges Arbeiten, Kooperationen

Shinoto, Maria (2025): Basiskompetenz „Text“: Neue Möglichkeiten archäologischer Arbeit, in: Bolder-Boos, Marion – Hageneuer, Sebastian – Pantelidis, Georg (Hrsg.), *Digitale Methoden des Lernens und Lehrens in der Archäologie*, Heidelberg: Propylaeum, S. 95–108, [online] https://doi.org/10.11588/propylaeum.1572.c23243.

Einführung – alles ist Text

Computertechnologie ändert sich in rasantem Tempo. Was heute aktuell und faszinierend ist, ist in wenigen Monaten schon veraltet. Aber es gibt Grundlagen, die sich nicht ändern. Eine dieser Grundlagen ist das Verarbeiten und Speichern von Informationen und Anweisungen (Programmen) als Text, das heißt: unmittelbar als Schriftzeichen. Zwar ist die Maschine nicht in der Lage, Text direkt zu verstehen, aber die Kommunikation mit der Maschine geschieht seit über 70 Jahren mit Schriftzeichen, also Text. Die Regeln wurden über die letzten fast 40 Jahre zunehmend in Unicode standardisiert, und so können wir davon ausgehen, dass wir hierüber – konkret: UTF-8[1] – jetzt und in Zukunft direkt mit allen Computern, unabhängig vom Betriebssystem und unabhängig von der verwendeten Sprache problemlos kommunizieren können.

Das Großartige an der Arbeit mit Schriftzeichen – von jetzt an als Text bezeichnet – ist, dass sie vollkommen ausreichen, um unsere Gedanken, Publikationen, Zeichnungen und Diagramme sowie Daten zu speichern und zu verarbeiten. Wie wir dieses nachhaltige, reproduzierbare Arbeiten gestalten können, und welche bislang nicht geahnten Vorteile für die archäologische Forschung und die denkmalpflegerische Arbeit sich daraus ergeben, das soll im Folgenden vorgestellt werden.

Dateien enthalten Informationen und Metainformationen

Allseits bekannt sind Office-Programme, die eine Textverarbeitung, Tabellenkalkulation, ein Präsentationsprogramm und optional Datenbankprogramme oder Notizprogramme enthalten können. Dateien, die in diesen Programmen erstellt werden, werden mit einer Dateiendung wie .docx für *Word*-Dateien oder .xlsx für *Excel*-Dateien gespeichert. Hierdurch weiß der Computer, mit welchem Programm die Datei geöffnet werden kann, und das Programm weiß, wie die Daten gespeichert sind. Denn die Daten solcher Programme sind „proprietär", das heißt, ohne bestimmte Programme sind die Daten nutzlos – und es ist nicht einmal klar, welche zusätzlichen Daten in den Dateien abgespeichert sind.

Solche zusätzlichen Daten sind „Metadaten", die über die eigentlichen Inhalte hinausgehen und Namen der Autor*innen, benutztes Computersystem, Bearbeitungsablauf und anderes umfassen können. Sie sind nützlich, aber potenziell auch gefährlich, wenn sie nicht vollständig von den Nutzer*innen kontrollierbar sind.

---

1 Unicode https://home.unicode.org [zugegriffen am 20.10.2023].

Dateien dagegen, die nur aus Text bestehen, speichern zunächst einmal nur die eigentlichen Inhalte. Auch sie können aber Metadaten enthalten, meist in getrennten Textblöcken. Solche Metadaten verbergen nichts und werden bewusst von den Nutzer*innen erstellt und gespeichert. Eine andere Form von Metadaten kann direkt in den Text eingefügt werden. Es sind Etiketten (tags), die eine Phrase einklammern können und deren Funktion oder Bedeutung näher bezeichnen, wie in diesen Beispielen für eine Funktion im Text (<header>Einführung</header>) oder eine Bedeutungserklärung (Es war ein <wetter>schöner</wetter> Tag).

Reine Textdateien können die genannten Programme der Office-Suiten ersetzen. Jede Textdatei ist grundsätzlich ohne Dateierweiterung von einem Terminal oder der Command Line auf jedem Computersystem lesbar und kann dort bearbeitet werden. Komfortabler wird das mit Texteditoren, die unten vorgestellt werden.

Analog zu den Dateierweiterungen komplexer Programme können reine Textdateien optional durch Dateierweiterungen charakterisiert werden: Ein erzählender Text kann die Endung .txt erhalten. Diese Endung kann auch für Daten im Textformat verwendet werden. Präsentationen können mit Etiketten (tags) ausgezeichnet und im Browser als .html Datei genutzt werden.

Daten, die einem bestimmten Standard für eine Datenstruktur folgen, werden meist als .csv gespeichert, ein Format, das die Tabellenstruktur betont, indem Datensätze (Reihen) durch ein Steuerzeichen für eine neue Zeile, und Variablen (Zellen oder Spalten) durch Tabulatoren, Kommas oder andere speziell festgelegte Zeichen definiert sind. Solche Tabellen lassen sich in Tabellenkalkulationen oder Datenbanken öffnen und bearbeiten. Auch andere strukturierte Datenformate können mit Dateierweiterungen wie .json und .yml gekennzeichnet werden und sind genauso im Terminal oder in Editoren lesbar.

Computerprogramme sind Text, ebenso Vektorgrafiken und 3D-Daten, die zum Beispiel als .svg, .mermaid oder .vml gespeichert werden. Alle Daten können direkt im Text manipuliert werden. So kann zum Beispiel eine rote Linie in eine grüne mit dem Austausch der Worte „red“ und „green“ geändert werden. Diagramme aus statistischen Anwendungen wie R sind per Text generierte Grafik.

Diese unterschiedlichen Ansätze können in einer Datei vereinigt werden und komplexe, gut formatierte Endprodukte erzeugen. Die folgenden Abschnitte zeigen schrittweise das Potenzial dieses Prinzips.

Eine Vielzahl archäologischer Tätigkeiten kann demnach in einfach verständlicher, zukunftssicherer Form erstellt und gespeichert werden, Open Source[2] spielt dabei eine zentrale Rolle. Formatstandards und komfortable Programme zur Bearbeitung sind nicht notwendig Open Source, aber tatsächlich bilden offene Lizenzen[3] für Programme,

2 Open Source https://de.wikipedia.org/wiki/Open_Source [zugegriffen am 15.01.2025].

3 Open source license https://en.wikipedia.org/wiki/Open-source_license [zugegriffen am 15.01.2025].

Dokumente und Standards eine enge Gemeinschaft mit Prinzipien der Plattformunabhängigkeit und Zukunftssicherheit.

Inhalt und Form werden getrennt

HTML, Markup und CSS

Beim Lesen von Texten haben wir uns daran gewöhnt, dass nicht alles in der gleichen Schrift durchgehend gleich geschrieben wird, sondern dass die optische Gestaltung des Textes weitere Informationen signalisiert: So werden Überschriften oft größer und fett geschrieben, Fremdwörter kursiv, Bedeutendes unterstrichen, oder zwischen einem Kapitel und einem Folgekapitel besteht ein größerer vertikaler Abstand. Solche Signale entsprechen Konventionen, die allgemein verstanden werden und Gültigkeit besitzen. Zudem ändern sich die Signale je nach Medium: Wird eine Überschrift in einer Tageszeitung groß, fett und schwarz gedruckt, so kann sie auf einer Webseite weniger groß, aber in Rot gestaltet sein.

Da ein reiner Text solche Formatierungen nicht enthält, wird gesondert festgelegt, wie bestimmte Textbestandteile gestaltet werden sollen. Oben wurden schon die Etiketten wie <header> oder <wetter> als Metadaten vorgestellt. Mit solchen tags, wie sie ab jetzt genannt werden sollen, wird angezeigt, welche Bestandteile besonders formatiert dargestellt werden sollen: So könnte festgelegt werden, dass ein header in einem gedruckten Text schwarz, fett und in 16 Punkt erscheinen soll, bei der Darstellung auf dem Bildschirm aber in derselben Größe wie der Lauftext, fett und in Rot. Diese Festlegungen folgen heute meist dem CSS-Standard, der entweder in einem Metadatenblock in der Textdatei selber festgehalten wird oder in einer gesonderten Datei, die dann die Dateierweiterung .css trägt und – selbstverständlich – wiederum aus reinem Text besteht. Die Regeln können im CSS-Text jederzeit geändert werden.

Mit der Trennung von Inhalt und Form gewinnt man einmal eine größere Klarheit im Text, wie die Auszeichnung <wetter> für das Wort „schön“ zeigt, aber auch eine große Flexibilität in der Gestaltung des Dokuments.

Das verbreitetste Textformat, das dieses Prinzip nutzt, ist HTML (Hypertext Markup Language), das für die Darstellung von Webseiten genutzt wird. Es zeichnete sich von Beginn an dadurch aus, dass Textbestandteile mit anderen verlinkt werden können, was heute selbstverständlich und auch in anderen Formaten wie PDF Standard ist. Durch die Links können zudem Fotos oder Scans im Pixelformat – eines der wenigen Dateiformate, in denen die Darstellung der Informationen als Text nicht sinnvoll ist – verlinkt werden. So werden bei der Darstellung im Browser aus formatierten Texten illustrierte Texte.

Texte, in denen tags für die nähere Beschreibung von Textbestandteilen genutzt werden, sind auf Deutsch als „Auszeichnungssprachen“, auf Englisch als „Markup

Languages“ bekannt.[4] HTML ist die am weitesten verbreitete Markup Language, und sie besitzt eine kleine Anzahl Standard-tags. Daneben werden zahlreiche weitere Metainformationen gespeichert, die man im eigenen Browser einsehen kann, wenn man die Quellcode-Ansicht einstellt. Es zeigt sich, dass mit solch stark angereichertem Markup ein einfacher Text schnell unleserlich werden kann. Er ist zudem mühselig zu schreiben.

Markdown und Pandoc

Vor diesem Hintergrund hat John Gruber 2004 das Markup-Format zu einem Markdown-Format abgespeckt (Gruber 2004). Er wollte damit das Schreiben von HTML-Seiten vereinfachen, aber seitdem hat Markdown alle Bereiche der Arbeit mit Computern und Softwareentwicklung erobert und auch in die wissenschaftliche Arbeit Eingang gefunden. Dateien in Markdown sind nach wie vor reiner Text, als Dateierweiterung hat sich .md weitgehend durchgesetzt.

Markdown ist in wenigen Minuten gelernt, und Texte in Markdown schreiben sich fließend: Man kümmert sich nur um die Inhalte, es gibt keine Sorgen um Seitenränder, Zeilenabstände, Fonts und anderes. Formatiert wird grundsätzlich automatisiert, standardisiert und flexibel – parallel zur Erstellung der Inhalte oder im Anschluss.

Die Möglichkeit, Markdown nicht nur in HTML, sondern auch in andere Formate wie .pdf, .docx, .pptx konvertieren zu können, wurde früh erkannt; für die Arbeit mit wissenschaftlichen Texten mussten nur noch Fußnoten und klassische Literaturverzeichnisse eingebunden werden. Durchgesetzt hat sich das Projekt *Pandoc* des Philosophen John MacFarlane (MacFarlane 2022), das 2006 begann und mittlerweile von einer fast unüberschaubaren Entwicklergemeinde in Open Source unterstützt wird und in weitere Open Source-Projekte eingebunden ist. *Pandoc* ist sehr konservativ, was Änderungen gegenüber dem ursprünglichen Markdown betrifft, und so werden *Pandoc*-Dateien austauschbar mit Markdown genutzt und tragen meist die Dateierweiterung .md. Eine wichtige Ergänzung ist der Metadatenblock zu Beginn eines *Pandoc*-Dokuments im YAML-Format.[5]

Sowohl für Markdown als auch für *Pandoc* gilt, dass sie aus zwei Komponenten bestehen: (1) Regeln für die Formatierung eines Textes, (2) Scripts für die Konvertierung von Markdown in HTML und von *Pandoc*-Markdown in verschiedene Formate. Die Scripts interessieren hier nicht weiter; es sind Werkzeuge, die jederzeit neu für verschiedene Plattformen geschrieben werden können oder gar manuell simuliert werden können, während die Regeln für die Formatierung der Texte Bestand haben.

---

4 Auszeichnungssprache
https://de.wikipedia.org/w/index.php?title=Auszeichnungssprache&oldid=239588499 [zugegriffen am 26.02.2024].

5 YAML https://yaml.org [zugegriffen am 15.01.2025].

Die Umwandlung des Ausgangstextes in HTML oder andere Formate funktioniert im Terminal oder der Command Line, mittlerweile bieten aber die meisten Editoren eine graphische Schnittstelle, die oft während der Texterzeugung schon das transformierte Dokument anzeigt.

Terminal, Texteditoren und integrierte Entwicklungsumgebungen

Text, Markdown, *Pandoc* – alle Formate lassen sich unabhängig von bestimmten Programmen bearbeiten, solange diese Programme Text editieren können. Dabei gibt es eine Stufenleiter der Komplexität von Editoren, die zunehmend weitere, auf bestimmte Aufgaben hin spezialisierte Funktionen bieten.

Es beginnt mit einfachen Terminalprogrammen wie *nano*[6], von denen sich grundsätzlich eines oder mehrere auf jedem Computer befindet. Es folgen reine Texteditoren, also Programme, mit denen sich Textdateien erstellen, öffnen, bearbeiten und speichern lassen; für Windows wäre *Notepad++* zu nennen[7], für Mac OS zum Beispiel der *CotEditor*[8]; es gibt zahlreiche Open Source-Lösungen, die oft plattformunabhängig arbeiten.

Meist bieten Texteditoren die Möglichkeit, die Encodierung der Texte mit einem Klick zu ändern. Heutzutage wird Text fast ausschließlich in UTF-8 encodiert, aber es gibt noch ältere Fälle und Exoten, die nach einem der zahlreichen älteren Standards codiert gespeichert sind, die oft nur auf eine bestimmte Sprache angewendet wurden, wie ASCII für amerikanisches Englisch oder Shift-JIS und EUC für Japanisch. Öffnet man heute solche Dateien in einem Texteditor, der normalerweise UTF-8-Dateien liest, so sieht man einen Buchstabensalat, der erst zu einem sinnvollen Text wird, wenn man im Texteditor die passende Encodierung wählt.

Dazu bieten Texteditoren Werkzeuge, die das Arbeiten mit Text erleichtern, und zwar oft mit einer bestimmten Anwendung im Blick. So haben einige Texteditoren Werkzeuge für das Erstellen von Webseiten und bieten Funktionen für das Schreiben von HTML und CSS, sie bieten Vorschauen, wie HTML oder Markdown in Browsern aussehen würde. Andere Editoren sind explizite Programmiertools, die den Umgang mit einer oder mehreren Programmiersprachen vereinfachen. Am oberen Ende der Skala stehen „Integrierte Entwicklungsumgebungen“ (IDEs)[9], die das komfortable Arbeiten mit unterschiedlichen Formaten erlauben und oft zusätzliche Funktionen wie eine integrierte Versionskontrolle oder die Konvertierung von Markdown-Dokumenten in PDFs und andere Formate mithilfe des eingebundenen *Pandoc* bieten.

---

6 *nano* https://www.nano-editor.org [zugegriffen am 15.01.2025].

7 *Notepad++* https://notepad-plus-plus.org [zugegriffen am 15.01.2025].

8 *CotEditor* https://coteditor.com [zugegriffen am 15.01.2025].

9 Integrierte Entwicklungsumgebung https://de.wikipedia.org/wiki/Integrierte_Entwicklungsumgebung [zugegriffen am 15.01.2025].

Der Übergang von einem einfachen Texteditor zu einer IDE ist fließend; für die eigene Arbeit kann man sich ein Werkzeug aussuchen und es weiter anpassen. An dem Ergebnis ändert die Wahl des Werkzeugs – also des Editors oder der IDE – nichts: Es werden plattformunabhängige, zukunftssichere Textdateien erstellt. Die Textdateien können informative Texte enthalten, aber auch Programme, Daten, programmatisch erstellte Zeichnungen und Diagramme oder 3D-Modelle, um nur die wichtigsten Formen zu nennen, die für unsere archäologischen Anwendungen von Bedeutung sein können.

Im Laufe der Arbeit mit Markdown und *Pandoc* entwickelt jede*r Vorlieben, als Einstieg können neben den genannten Texteditoren zwei IDEs im Jahr 2025 empfohlen werden: *VSCode*[10] und *RStudio*[11]. *RStudio* ist Open Source und kann frei heruntergeladen werden; bei *VSCode* sieht es etwas komplizierter aus: Es gibt Versionen, die vollkommen mit Open Source-Lizenzen versehen sind und eine, in die Microsoft ein paar nicht offene Elemente eingefügt hat, vor allem die telemetrische Überwachung. Diese Version kann ebenfalls frei heruntergeladen werden und wird am häufigsten genutzt. Sie ist nicht nur für Neulinge trotz der Einschränkung empfehlenswert. *VSCode* hat ein hervorragendes, flexibles Interface und ist ausgesprochen vielseitig und stabil.

RStudio wurde ursprünglich für das Schreiben von R-Code entwickelt, das Interface wirkt heute etwas in die Jahre gekommen. Dennoch ist diese stabile Anwendung mit hoch innovativen Entwickler*innen im Hintergrund rundum empfehlenswert. Eine vollkommene Neuentwicklung von *RStudio* als IDE auf der Basis von *VSCode* ist schon weit gediehen und ebenfalls nutzbar.

Beide IDEs sind bestens für die Arbeit mit Texten im Format *Quarto* geeignet, das eine Weiterentwicklung von Markdown ist und die momentan fortschrittlichste Anwendung; *Quarto* wird unten näher vorgestellt. Zudem bieten sie Werkzeuge für die Arbeit mit Versionskontrolle und zentralen Repos.

---

10 *Visual Studio Code* https://code.visualstudio.com [zugegriffen am 15.01.2025].

11 *RStudio Desktop* https://posit.co/download/rstudio-desktop [zugegriffen am 15.01.2025].

Versionskontrolle ist Grundlage moderner wissenschaftlicher Arbeit

Versionskontrolle ist einer der großen Vorteile des Arbeitens mit Text. Es geht hierbei eben nicht nur um Texte im üblichen Sinne, es geht auch um digitale Textdateien, die, wie in der Einleitung angesprochen, Daten darstellen, Zeichnungen und Diagramme programmatisch erstellen und sie in Textformaten wie .svg speichern.

Solche Dateien stehen nicht von Anfang an als fertiges Ergebnis da, sondern werden langsam entwickelt. Bestimmte Bereiche werden testweise geschrieben und wieder verworfen oder stückweise überarbeitet, und so fort.

Schrittweise „Texterstellung" ist zunächst in der Softwareentwicklung bewusst mit der Erstellung von Programmversionen in geordnete Bahnen gelenkt worden, und spätestens in den 1970ern begannen verschiedene Ansätze einer systematischen Versionskontrolle.[12] Letztlich hat sich ein Open Source-System namens *Git*[13] durchgesetzt, das seit den 2000er Jahren die Software-Welt revolutioniert und auch für die Erstellung wissenschaftlicher Dokumente von hohem Nutzen ist.

In Office-Programmen werden nur temporär Korrekturen oder Kommentare eingetragen und später vollkommen übernommen oder gelöscht. Bei einer Versionskontrolle mit *Git* dagegen hat man die Entwicklung des gesamten Projekts im Blick und kann jederzeit zu früheren oder alternativen Versionen wechseln. Die Entstehung eines Dokuments ist nachprüfbar und reproduzierbar.

Git funktioniert auf jedem Computer, es handelt sich um einen unsichtbaren Ordner in einem übergeordneten Ordner, in dem ansonsten alle Projektdateien – Daten, Texte, Abbildungen, Scripte etc. – gespeichert sind; der Projektordner sieht wie eine normale Sammlung von Dateien aus. Ändert man eine Zeile oder mehrere in diesen Dateien, wird das im unsichtbaren *Git*-Ordner erfasst.

Nicht jede Änderung ist eine Version. In sinnvollen Abschnitten, wenn man zum Beispiel ein bestimmtes Fundstück beschrieben hat, kann man sich ansehen, wie die Änderungen aussehen, und wenn man einverstanden ist, kann man diese Änderung „committen", das bedeutet: Man schreibt eine kurze Zeile mit der Charakterisierung der Änderung („füge Beschreibung Fundstück X an") und sichert den neuen Status, ein „Commit". Das geschieht wahlweise im Terminal oder in einer der verschiedenen Open Source-Applikationen oder direkt in einer IDE während des Schreibens. *Git* speichert nun diese Informationen gemeinsam mit den Änderungen, die man jederzeit mit anderen Commits vergleichen kann. Hat man mit mehreren Commits ein bestimmtes Zwischenziel erreicht, kann man dem abschließenden Commit eine Versions-ID geben.

---

12 Versionsverwaltung https://de.wikipedia.org/w/index.php?title=Versionsverwaltung&oldid=226425173 [zugegriffen am 07.03.2024].

13 *Git* https://git-scm.com [zugegriffen am 10.10.2023].

Man kann alternative Textversionen in Verzweigungen speichern und bei Bedarf mit anderen Zweigen vereinen; man kann in der Versionsgeschichte nach bestimmten Begriffen suchen wie zum Beispiel „Fundstück X“. Die Entwicklungsgeschichte mit den hunderten Commits wird dabei nicht nachträglich geändert; man wird Änderungen in einem alten Commit mit einem neuen Commit durchführen müssen. So bleibt der Entwicklungsprozess deutlich, die Nachvollziehbarkeit oder Reproduzierbarkeit des Wissenschaftsprozesses bleibt gewahrt.

Git ist dezentral, das bedeutet, dass man auf mehreren Computern eine Kopie des Repo getrennt speichern und frei benutzen und verändern kann. Irgendwann müssen solche Repos zusammengeführt werden, um ein Projekt gültig abzuschließen oder von Zeit zu Zeit wieder auf einen Nenner zu kommen. Das kann direkt geschehen, oder, wesentlich praktischer, indirekt über ein zentrales Repo, das vorzugsweise über das Internet für alle erreichbar ist.

Hierfür gibt es Open Source-Plattformen wie *GitHub*[14], *GitLab*[15] und *Bitbucket*[16]. Zur Zeit, als dieser Text geschrieben wird, ist *GitHub* die verbreitetste und damit auch für Neulinge die am einfachsten zugängliche Variante. Hier kann man seine Repos hochladen und von verschiedenen Computern ansteuern oder aber diese mit Kolleg*innen teilen, die auf ihrem Computer eigene Versionen des Projekts bearbeiten und von Zeit zu Zeit diese Versionen mit dem zentralen Repo abgleichen.

Github wie auch die anderen Plattformen bieten an, die Repos privat zu halten – also Einsicht nur sich selber und ausgewählten Kolleg*innen zu gewähren, oder aber sie öffentlich zu stellen, so dass jede*r Einsicht hat und die Projekte „forken“ (kopieren und mit dem Original verbunden halten) und weiter entwickeln kann. So kann man mit Kolleg*innen in aller Welt und allen Disziplinen kommunizieren, es können gemeinsam Daten, Analysen, Texte oder Abbildungen bearbeitet und diskutiert werden.

In der Softwarebranche ist solche Offenheit selbstverständlich, geistiges Eigentum wird durch verschiedene Lizenzen[17] geschützt. Die Offenheit hat in den letzten Jahren zu einer Explosion von Entwicklungen geführt, von denen wir auch in der Archäologie profitieren. Da sich auch unsere Disziplin immer mehr der offenen Forschung verschrieben hat, werden auch archäologisch orientierte Repos zunehmend öffentlich sein.

Zu Beginn ist die Arbeit mit dem kostenlosen Open Source-Programm *Github Desktop*[18] empfehlenswert, und zwar auf dem eigenen Computer und weil es vieles im

---

14 *GitHub* https://github.com [zugegriffen 10.10.2023] und *The DevSecOps*-Platform https://about.gitlab.com [zugegriffen am 10.10.2023].

15 *The DevSecOps*-Platform https://about.gitlab.com [zugegriffen 10.10.2023].

16 *Bitbucket* https://bitbucket.org/product [zugegriffen am 10.10.2023].

17 *Open Source license* https://en.wikipedia.org/wiki/Open-source_license [zugegriffen am 15.01.2025].

18 *GitHub Desktop* https://github.com/apps/desktop [zugegriffen am 15.01.2025].

Umgang mit der Plattform *GitHub* automatisiert und erleichtert. Später werden sich eigene Vorlieben wie Arbeit im Terminal oder direkt in den IDEs entwickeln.

Literaturdatenbanken und Zitierformate sind Text

Seit mehreren Jahrzehnten wird in den Naturwissenschaften vorzugsweise mit einem Satzsystem namens LaTEX gearbeitet. Hier werden wissenschaftliche Texte als Text gesetzt, es ist die gefeierte Alternative zu Office-Anwendungen. Kombiniert wird das Satzprogramm mit einer Literaturverwaltung, die ebenfalls schlichter, nach Regeln strukturierter Text im ASCII-Format ist: Bibtex (Patashnik 2003). Heutzutage hat die UTF-8-fähige Weiterentwicklung Biblatex das alte Bibtex (Kime – Wemheuer – Lehman 2024) abgelöst. Dateien im Biblatex-Format tragen die Endung .bib, während alte Dateien im Bibtex-Format, früher ebenso gekennzeichnet, heute aus Konvention mit der Endung .bibtex gekennzeichnet werden. Die Formate sind fast austauschbar.

Die komplexe Kombination aus LaTEX und Biblatex sollte uns heute nicht mehr interessieren, aber sie spielt hinter der Bühne nach wie vor eine Rolle in der Transformation von Markdown-Dokumenten. Im Zusammenhang mit Literaturverwaltung ist das Biblatex-Format nach wie vor aktuell. Als Textdateien können Biblatex-Daten in Versionskontrolle gegeben werden und im Laufe des Studiums und der wissenschaftlichen Karriere wachsen. Literaturdaten aus Bibtex-Dateien können mit *Pandoc* in die archäologischen Texte integriert werden, so dass die Kombination Markdown/*Pandoc* mit Biblatex ein perfektes akademisches Dokument erzeugen kann.

Das System formatiert sowohl die Zitate im Text als auch die Literaturliste in beliebigen Formaten, je nach Vorgaben der Institution oder der Zeitschrift, immer korrekt und in Sekunden. Die Formatierung nach Vorgaben wird wiederum mit einer Textdatei geregelt, vorherrschend in dem logisch gut nachvollziehbaren CSL-Format mit der Dateiendung .csl[19]. Es gibt mittlerweile etwa zehntausend solcher Formatierungsdateien zum Download[20], auch das DAI bietet solche Dateien an[21].

Die Literaturdatenbank im Format Biblatex und die Formatierungsanweisungen im Format CSL können in jedem Texteditor bearbeitet werden, aber für Biblatex

19 Citation Style Language https://citationstyles.org [zugegriffen am 15.01.2025].

20 *Zotero* Style Repository https://www.zotero.org/styles [zugegriffen am 15.01.2025].

21 *Zotero* Style Repository – Deutsches Archäologisches Institut https://www.zotero.org/styles?q=Deutsches%20Arch%C3%A4ologisches%20Institut [zugegriffen am 15.01.2025].

gibt es eine komfortablere graphische Benutzerschnittstelle namens *JabRef*[22], und für die Bearbeitung von CSL-Dateien bietet sich ein interaktives graphisches Interface im Web an[23].

Es geht aber noch einfacher. Das Open Source-Projekt *Zotero*[24] entwickelt seit langem ein graphisches Benutzerinterface für die Suche von Literatur im Netz, für den Import solcher Informationen in das Programm und für die manuelle Eingabe sowie für die Verwaltung von eventuell damit verbundenen PDFs und eBooks.

Zotero kann Literaturdaten als Biblatex exportieren und bietet Schnittstellen an, mit denen CSL genutzt und modifiziert werden kann. Mit Backups der in *Zotero* verwalteten Daten in Biblatex-Dateien erhält man eine zukunftsfähige Datenbasis, die in Versionskontrolle gegeben werden kann.

Für die tägliche Arbeit mit der Literatur ist das Interface von *Zotero* allerdings unschlagbar gut und sollte Texteditoren und IDEs vorgezogen werden. Als Werkzeug, das trotz seiner großen Verbreitung vielleicht nicht endlos zukunftssicher ist, sei es für die Arbeit heute und in näherer Zukunft empfohlen. Neue Werkzeuge, die mit den zukunftssicher als Text gespeicherten Daten arbeiten können, werden in fernerer Zukunft jetzige Werkzeuge ablösen.

Integration der Komponenten

Wir haben in den letzten Abschnitten Komponenten kennengelernt, die zusammen und auf Basis von rein digitalem Textformat die Erstellung von wissenschaftlichen Arbeiten mit Daten, Abbildungen, Programmcode, eingebettet in den Textfluss, erleichtern und beschleunigen:

1. Markdown/*Pandoc* für die reibungslose Texterfassung – mit Abbildungen, Fußnoten, Links, Metadaten (.md) und Transformation in andere Formate.
2. Literaturdatenbanken im Textformat (.bib), aus denen heraus Zitate und Literaturlisten oder Bibliographien automatisiert in das Hauptdokument eingefügt werden können und Formatierungsregeln im Textformat (.csl).
3. Versionskontrolle (.git), die den schrittweisen und reproduzierbaren Aufbau solcher Dokumente sowie die globale Zusammenarbeit bei der Erstellung erlauben.

---

22 *JabRef* https://www.jabref.org [zugegriffen am 10.03.2024].

23 Editing CSL Styles https://www.zotero.org/support/dev/citation_styles/style_editing_step-by-step [zugegriffen am 15.01.2025].

24 *Zotero* https://www.zotero.org [zugegriffen am 15.01.2025].

Entwicklung hin zu Quarto

Aus diesen Komponenten hat ein im Open Source-Bereich sehr aktives Unternehmen, das auch die IDE *RStudio* entwickelt hat, den Standard *Quarto* entwickelt, 2022 erstmals veröffentlicht und seitdem rasant mit Optionen angereichert.[25] Das zentrale Dokument trägt nun die Dateierweiterung .qmd statt .md oder .Rmd. Alle Komponenten in einem *Quarto*-Projekt bleiben miteinander verbunden und aktualisieren sich selbständig, was gerade für die schrittweise Entwicklung von Daten, Forschung und Berichten hilfreich ist. Häufig zu aktualisierende Berichte lassen sich automatisiert schreiben.

Quarto ist ein zukunftssicherer Textstandard, er ist aus dem R Markdown-Projekt[26] hervorgegangen, das spätestens seit 2015 in der Welt der R-Programmierung großen Anklang gefunden hat. R Markdown vereint die oben genannten Komponenten, ist aber auf Programmblöcke in R beschränkt und benötigt Erweiterungen für viele Funktionen, die letztlich zu einem fragilen System führen. *Quarto* hat dieses Problem grundsätzlich gelöst: Es ist mit Blöcken verschiedener Programmiersprachen, vor allem Python, kombinierbar, und alle Komponenten sind integriert und ohne Erweiterungen in *Quarto* selber nutzbar. *Quarto* kann über ein Plugin in anderen IDEs neben *RStudio* genutzt werden, wie eben auch *VSCode*.

R Markdown und *Quarto* sind nicht aus dem Nichts entstanden; zwei wichtige Komponenten aus der Geschichte sollen kurz vorgestellt werden: (1) Donald Knuth entwickelte das Prinzip des „Literate Programming“ (Knuth 1992). Hierbei wird nicht zuallererst ein Programm geschrieben und bei Bedarf mit Kommentaren versehen, sondern es wird zunächst das Programm in einem Text beschrieben und schrittweise in Codeblöcken in diesem Text umgesetzt und ausgeführt. (2) Sogenannte „Notebooks“ gibt es spätestens seit den 1990er Jahren, sie setzen das Ideal des Literate Programming um, indem sie abwechselnd Textblöcke und Codeblöcke für die Entwicklung von Programmen aufbauen[27] und diese Programme auch ausführen.

R Markdown Notebooks (.Rmd) deuteten schon die Erweiterung von Notebooks von einem Programmierwerkzeug hin zu einem Kommunikations- und Dokumentationswerkzeug an. Wer Markdown und *Pandoc* kennt, kann R Markdown ohne Probleme in diesen Notebooks nutzen, und wer R Markdown kennt, kann problemlos auf *Quarto* umsteigen und damit PDFs, Zeitschriftenartikel, Webseiten, Semesterarbeiten und größere akademische Werke verfassen, aber auch im Austausch mit Kolleg*innen weltweit Projekte entwickeln.

---

25 *Quarto* https://quarto.org [zugegriffen am 15.01.2025].

26 RMarkdown https://rmarkdown.rstudio.com [zugegriffen am 20.10.2023].

27 Notebook interface https://en.wikipedia.org/wiki/Notebook_interface [zugegriffen am 15.01.2025].

Workflow und Bedeutung für die moderne Archäologie

Die Arbeit mit *Quarto* kann schon im ersten Studiensemester eines Archäolog*innenlebens beginnen. Die gut 13 Jahre, die die Autorin schon mit den Vorläufern und nun *Quarto* arbeitet, zeigen einen enormen „Compound-Effekt“, bei dem alte Projekte nicht einfach verloren gehen, sondern in neue Projekte einfließen können. Ein Workflow kann wie folgt aussehen:

1. Notizen können in *Quarto* als versionskontrolliertes Projekt erstellt und miteinander verlinkt werden. Aus den Notizen können sich Projekte entwickeln; Notizen können als Ausgangsdokumente in neue Projekte transferiert werden.
2. Semester- oder Abschlussarbeiten können von Null auf in einem *Quarto*-Projekt erstellt werden, mit Text, Daten, Analysen, Karten. Hierbei kann ein Repo für bestimmte Personen einsehbar geschaltet werden, Kontrolle durch Lehrende ist innerhalb des Repo einfach über Issues und Kommentarfunktionen möglich; gleichzeitig ist die Arbeit der Lehrenden und Gutachter*innen dokumentiert und kann von den Studierenden systematisch abgearbeitet werden.
3. Forschung kann in *Quarto*-Projekten geschehen, internationale und interdisziplinäre Zusammenarbeit funktioniert auf die gleiche Weise wie bei Arbeiten im Verlauf des Studiums. Solche Projekte können Datensammlungen, Analysen, Zeitschriftenartikel und größere Publikationen in sich vereinen.
4. Artikel für Zeitschriften können in *Quarto* bis zur Abgabe frei geschrieben und im jeweils passenden Format für eine Zeitschrift eingereicht werden. Bei der Abgabe in einer anderen Zeitschrift ist der Aufwand für die Anpassung an neue Formate minimal.
5. Repositorien können selbst als Publikationsformat genutzt werden, zum Beispiel in *GitHub Pages*[28] oder mit *Zenodo*[29].

Solch ein Workflow zeigt das Potenzial der integrierten Arbeit mit Text, Versionskontrolle und Literaturverwaltung für die archäologische Forschung, für Reproduzierbarkeit von Arbeitsschritten und den internationalen Austausch. Hierzu müssten natürlich alle Beteiligten mit dem System und seinen Komponenten vertraut sein. Es ist anzunehmen, dass sich nie alle archäologisch Tätigen mit diesen Standards vertraut machen werden, und dass gerade Kooperationen sich nicht vollkommen mit diesen Standards durchführen lassen. Das sollte einen nicht abhalten, die eigene Forschung systematisch mit den hier vorgestellten Komponenten aufzubauen und das System auch in Kooperationen so weit wie möglich zu nutzen.

---

28 *GitHub Pages* https://pages.github.com [zugegriffen am 15.01.2025].

29 *Zenodo* https://zenodo.org [zugegriffen am 15.01.2025].

ORCID®

Maria Shinoto https://orcid.org/0000-0001-6089-4526

Referenzen

Gruber, John (2004): *Markdown*, [online] https://daringfireball.net/projects/markdown [zugegriffen am 10.10.2023].

Kime, Philip – Wemheuer, Moritz – Lehman, Philipp (2024): *Das biblatex Paket. Das Benutzerhandbuch*, [online] https://tug.ctan.org/info/translations/biblatex/de/biblatex-de-Benutzerhandbuch.tex [zugegriffen am 11.02.2025].

Knuth, Donald E. (1992): *Literate programming*. Stanford, CA: Center for the Study of Language and Information.

MacFarlane, John (2022): *Pandoc: a universal document converter*, [online] https://pandoc.org [zugegriffen am 10.10.2023].

Patashnik, Oren (2003): BibTEX yesterday, today, and tomorrow, in: *TUGboat*, 24, Nr. 1, S. 11–21.