Maria Shinoto

Reproduzierbarkeit von der Datenerfassung bis zur Publikation – Theorie und methodisches Vorgehen im Beispiel

Abstract Basic considerations regarding data reproducibility in archaeological research and a guide to the first steps in working with data under version control. This session introduces the necessary concepts and simple, widely used tools (*VS Code*, *GitHub Desktop*, and other *open-source* tools), and walks through a hypothetical minimal example for participants to follow along with. Some steps need to be performed only once – such as installing certain programs. Other steps are repeated throughout the workflow and become routine in the research process – such as creating *commits* and versions. This is supplemented by an open repository containing sample data and comments on all steps in the practical example.

Zusammenfassung Im Beitrag geht es um grundsätzliche Gedanken zur Reproduzierbarkeit von Daten in der archäologischen Forschung und bietet eine Anleitung für die ersten Schritte der Arbeit mit Daten in Versionskontrolle. Es werden die notwendigen Konzepte und einfache, weit verbreitete Werkzeuge (*VS Code, Github Desktop*, andere *Open-Source*-Werkzeuge) vorgestellt sowie ein hypothetisches Minimalbeispiel zum Nacharbeiten durchgespielt. Einige Schritte sind nur einmal notwendig, etwa die Installation bestimmter Programme. Andere Schritte wiederholen sich im Arbeitsablauf und werden zur Routine im Forschungsprozess, wie das Erstellen von *commits* und Versionen. Ergänzend gibt es ein offenes Repositorium mit Beispieldaten und Kommentaren zu allen Arbeitsschritten im Praxisbeispiel.

Keywords Versionskontrolle, Git, Quantitative Forschung, Tutorial, Reproduzierbarkeit

Shinoto, Maria (2026): Reproduzierbarkeit von der Datenerfassung bis zur Publikation – Theorie und methodisches Vorgehen im Beispiel, in: Bolder-Boos, Marion – Hageneuer, Sebastian – Pantelidis, Georg (Hrsg.), *Digitale Kompetenzen in den archäologischen Disziplinen. Perspektiven für die Lehre*, Heidelberg: Propylaeum, S. 105–128, [online] https://doi.org/10.11588/propylaeum.1773.c26532.

Zu Beginn

In diesem Tutorial möchte ich konkret die Werkzeuge und Schritte zeigen, mit denen reproduzierbar gearbeitet werden kann. Das Ziel ist nicht, die Verarbeitung von Daten im Textformat zu erklären, sondern die Leserinnen und Leser in die Lage zu versetzen, einmal den gesamten Prozess des Versionierens durchzugehen. Dies ließe sich gleichermaßen alternativ mit Programmiercode oder akademischem Text zeigen; hier aber werden wir eine kleine Datenbasis aufbauen, wie sie für Seminar- oder Abschlussarbeiten im Studium entstehen und bei Bedarf in größere Datenbanken integriert werden können.

Konzepte, Werkzeuge und Material

In der archäologischen Praxis ist der Umgang mit Daten – von Fund- und Kontextbeschreibungen bis zu Metainformationen – unser tägliches Brot. Schon im Studium lernen wir, Datenblätter anzulegen. Diese Datenblätter sind zumeist Tabellen in Programmen wie *Excel* oder dem *Open-Source*-Gegenstück *LibreOffice Calc* oder in computerbasierten Datenbanken, die von Projekten und Institutionen eingerichtet und unterhalten werden. Wer gerade seine eigene Forschung aufbaut und beginnt, Daten zu sammeln, wird sich Gedanken darüber machen, wie und mit welchen Werkzeugen die eigenen Daten gespeichert und bearbeitet werden sollen.

Hier soll ein einfacher, praktikabler und gleichzeitig theoretisch und methodisch solider Weg vorgeschlagen werden, der die Entwicklung von einfachen Lösungen hin zu professionellen Datenbanken offenhält.

Datenbanken – Relationales Datenmodell – Tabellen

Daten werden in Datenbanken verwaltet – was aber ist eine Datenbank? Es sind nicht nur große Client-Server-Lösungen: Auch die systematische, regelbasierte Erfassung auf Karteikarten oder in Tabellen auf Papier führt zu einer Datenbank. Digitale Datenbanken sind erheblich flexibler, und das in den 1970er Jahren entwickelte Prinzip der digitalen relationalen Datenbanken, basierend auf dem relationalen Datenmodell (Codd 1970), ist fast synonym geworden für „Datenbanken“. Das Prinzip hinter Datenbanken sowie das relationale Datenmodell benötigen allerdings keine digitalen Werkzeuge oder SQL-basierte oder in *Client* und *Server* getrennte Datenbanken, sondern es kann sowohl auf Papier als auch in einfachen textbasierten Dateien dargestellt werden. – Das relationale Datenmodell wird bei Hernandez (Hernandez 2020) gut verständlich erklärt.

Prinzipiell stellt ein relationales Datenmodell komplexe Situationen in getrennten Tabellen dar: So kann, wie in unserem Beispiel unten, eine Sammlung von Beobachtungen an Keramik aufgeteilt werden in Tabellen mit unterschiedlichen Themen; hier sei der einfachste Fall angenommen: (1) Eine Tabelle beschreibt die Keramik, (2) eine weitere Tabelle beschreibt den Fundkontext **(s. Abb. 7a)**.

Das Modell wird in einem *Entity-Relationship-Diagramm* (ERD) dargestellt. Die getrennten Tabellen können manuell oder automatisiert durch das Suchen nach den IDs und die Verknüpfung der Informationen unterschiedlicher Tabellen miteinander verbunden werden. Außerhalb spezifischer relationaler Datenbanken bieten sich die Möglichkeiten von Analyseprogrammen wie R an, um Tabellen aus Tabellenkalkulationen oder Textdateien flexibel zu verknüpfen.

Wichtig für die Nachprüfbarkeit *(Reproducibility)* ist, ein möglichst weit verbreitetes und stabiles Dateiformat zu wählen, das auch langfristig lesbar und nicht von bestimmten proprietären Softwareversionen abhängig ist. Es gibt verschiedene Lösungsansätze, aber der einfachste ist sicher der, Daten aus einer Tabellenkalkulation in einfachen Text (.csv) zu exportieren und diese Texte in die Versionskontrolle zu geben. Dieser Weg soll hier theoretisch und konkret vorgestellt werden.

Versionen – Standards – Öffentlichkeit

Wenn wir unsere Daten und die Forschung mit unseren Daten reproduzierbar gestalten wollen, müssen wir sie (1) öffentlich zugänglich und (2) allgemein verständlich gestalten.

Das Konzept für die öffentliche Zugänglichkeit von Daten kennt man unter Begriffen wie *Open Data*, FAIR Data oder der Nationalen Forschungsdateninfrastruktur (NFDI). Sie überschneiden sich in weiten Bereichen und ermöglichen den Zugang zu „fertigen Versionen", oft auch zum Erstehungsprozess.

„Fertige Versionen" ist in Anführungszeichen geschrieben, weil eine Datensammlung natürlich nie wirklich „fertig" ist. Es gibt immer etwas zu ergänzen oder zu korrigieren. Kataloge in Buchpublikationen sind genauso „vorläufig" wie versionierte Schnappschüsse einer digitalen Datenbank, und beide Varianten sind entstanden aus zahllosen Zwischenschritten, die bislang zumeist undokumentiert geblieben sind. Mit Versionskontrolle können diese Zwischenschritte heutzutage jedoch genauso dokumentiert werden wie die größeren Versionsschritte.

Die Arbeit mit Versionskontrolle und den dafür notwendigen Repositorien soll im Beispiel deutlicher werden, denn sie ist mit Sicherheit der Schritt in der Arbeit mit Daten, der in der archäologischen Lehre am wenigsten beachtet wird. Dabei gehört er zum Wichtigsten, was wir für das Forscher*innenleben mitnehmen können.

Versionskontrolle ist in verschiedener Form denkbar, aber der empfehlenswerte Weg im aktiven Entwicklungsprozess geht über die ubiquitäre *Open-Source*-Lösung *Git*. In einem *Git*-Repositorium können alle Daten und der Ablauf ihrer Bearbeitung

wiederauffindbar beschrieben werden. Bei der Arbeit mit *Git* kann man alleine auf dem eigenen Computer bleiben, man kann die Daten jedoch auch in einem zentralen Repositorium auf einem Webserver speichern. Auch das kann privat geschehen oder von Beginn an oder ab einer bestimmten Stufe öffentlich gestellt werden. Auf diesen Servern kann man neben den Daten selbst die gewählten Standards und Definitionen publizieren. So sind, wie eingangs gefordert, sowohl die (1) öffentliche Zugänglichkeit als auch die (2) Verständlichkeit gewährleistet.

Es gibt verschiedene Standards für die Datenspeicherung, und gerade bei einem Upload in öffentliche Repositorien ist es wichtig, nicht nur die inhaltlichen Fragen an das Material zu klären, sondern auch die jeweils gültigen Standards zu berücksichtigen. Dennoch besteht kein Zwang, von Beginn an nach externen Kriterien zu arbeiten. In der Versionskontrolle kann man sehr frei Schritt für Schritt von chaotischem Datenmaterial hin zu standardisiertem, mit Metadaten versehenem Material übergehen. Will man die Daten dann in öffentlichen Datenbanken, als Material zu einem Journal-Artikel oder in Projekten publizieren, so können die Daten gezielt nach den Kriterien des Standards überarbeitet werden. Besonders praktisch ist die Option, solche temporären Standardisierungen in einem gesonderten Entwicklungszweig *(branch)* durchzuführen und im Hauptzweig *(main branch)* unbeeinflusst von externen Anforderungen weiterzuarbeiten.

Charakter archäologischer Datenpublikationen

Die archäologische Forschung befindet sich im Umbruch: Daten wurden bislang ausgewählt, bearbeitet und gedruckt publiziert; sie mussten von nachfolgenden Arbeiten in dieser Form angenommen werden. Dieses Vertrauen, das dabei in die Autor*innen gesetzt werden muss, ist die Basis archäologischer Forschung. Auf der anderen Seite sind die kritische Überprüfung durch Dritte und die Bereitschaft, eigene Arbeiten offenzulegen und der Kritik anheimzustellen, wohl das wichtigste Charakteristikum wissenschaftlichen Arbeitens. Insofern ist die aktuelle Entwicklung hin zur Offenlegung von Daten vom Zeitpunkt ihrer Entstehung bis zur „Präsentierbarkeit" ein großer Fortschritt. Forschung wird so in der Archäologie allgemein und zeitnah zur Entstehung der Daten zugänglich. Es bleiben aber Probleme, die vor allem unserem Fach eigen sind.

1. Die Erfassung archäologischer Daten ist mit Zerstörung verbunden, sofern es sich um Ausgrabungen oder nicht zerstörungsfreie archäometrische Untersuchungen handelt. Nachfolgende Generationen sind somit auf die Informationen angewiesen, die die damaligen Akteur*innen mit ihren forschungsgeschichtlichen Befangenheiten gesammelt haben.
2. Fundstücke sind nicht grundsätzlich allgemein zugänglich: Sie können verloren gehen, der Zugang kann eingeschränkt werden, als „unwichtig" erachtete

Fundstücke oder Beobachtungen werden wegen geringer Speicherkapazitäten bewusst vernichtet und die Kosten für eine erneute Fundaufnahme am Originalmaterial nach neuem Forschungsstand sind selten tragbar. Eine solide Aufbereitung und Publikation der Daten ist eine einmalige Gelegenheit.

Hinzu kommen zwei Besonderheiten in der Zielstellung von Datenaufnahmen:

a) Dokumentierende Datenaufnahme: Diese sollte das Material systematisch, möglichst objektiv und erschöpfend erfassen und damit so weit wie möglich den erneuten Zugriff auf das Original überflüssig machen. Das Problem ist, dass Objektivität und die erschöpfende Erfassung von Eigenschaften nie möglich sein werden. Originale vermitteln zudem oft Anderes als Daten – egal ob in Zahlen, Beschreibungen oder Abbildungen.
b) Die interpretierende oder zweckgebundene Datenaufnahme ist von vornherein nicht „objektiv", sondern explizit mit einer Forschungsfrage und damit Forschungstradition oder Methode verknüpft. Sie kann direkt am Befund geschehen oder auf Basis einer schon bestehenden Datenbank – die nach Möglichkeit auf „dokumentierender Datenaufnahme" im besten Sinne beruhen sollte.

Es ist wohl fair zu behaupten, dass ältere Publikationen eher auf Gruppe (b), der interpretierenden Datenaufnahme beruhen. Heutige Technologie ist in der Lage, weniger traditionsabhängige Daten in großen Mengen zu erfassen und wiederauffindbar zu speichern. Eine mit moderner Technologie erstellte Datenbank, die von Beginn an in ihrer Entstehung öffentlich gestellt wird, kommt damit dem Ideal (a), der dokumentierenden Datenaufnahme, näher. Trotzdem sind Datenbanken der Gruppe (b) gerade bei thematischen Projekten von Bedeutung. Letztlich gehen beide Formen ineinander über, und die Fähigkeit der spontanen Datenmanipulation in moderner Statistiksoftware ermöglicht jederzeit die Erstellung eines interessebasierten Datensatzes aus jeglicher Ausgangsdatenlage.

Ablauf einer reproduzierbaren Datenaufnahme

Plant man ein Forschungsprojekt, eine Seminar- oder Abschlussarbeit, für die eine Datenaufnahme erforderlich ist, gibt es grob drei Stufen, in denen die Datenaufnahme abläuft: (A) Theoretische Vorüberlegungen, (B) Praktische Vorbereitungen, (C) Implementierung. Die Stufen (A) und (B) umfassen fünf Bereiche, die nicht strikt voneinander getrennt sind und praktisch in keiner Reihenfolge zueinander stehen:

Bereich 1: Entscheidung über die den Daten angemessene Datenaufnahme.

Bei den (A) theoretischen Vorüberlegungen geht es zunächst um die Datenquellen: Hat man es mit einem steten Strom von neuen Informationen während einer Grabung

oder einem Survey zu tun? Kann man Originale in einem Magazin aufnehmen, vielleicht ergänzt mit älteren Dokumentationen? Oder arbeitet man allein mit Dokumenten (Schriften, Listen, Fotografien, Zeichnungen), die publiziert oder in einem Archiv gelagert sind? Hiernach entscheidet sich, wie die Aufnahme entwickelt werden und wie weit sie gediehen sein muss, bevor die Fundaufnahme beginnt.

Bereich 2: Datenmodell erstellen. Man sollte zumindest vorläufig das oben angesprochene Datenmodell erstellen (ERM/ERD).

Bereich 3: Normalisierung und Definition der Variablen. Praktisch gleichzeitig mit dem ERM wird man überlegen, welche Daten konkret in welche Tabelle gehören, also welche Spalten eine Tabelle enthält. Jede Tabelle enthält genau zu definierende Variablen.

Bereich 4: Sicherung des Zugangs zu den Daten. Zu den (B) praktischen Vorbereitungen gehört das Sichern der Datengrundlage. Hier wird geklärt, wie einem die zu erfassenden Informationen auf einer Grabung zugeführt werden, welche Stücke oder Dokumente einem wann in einem Magazin oder Archiv zur Verfügung stehen und mit welchen Mitteln sie erfasst werden können. Der entspannteste Weg zu einer Datenaufnahme ist sicher der über publizierte Listen und Kataloge, die allerdings die Vielfalt und „Objektivität" der Daten schon von vornherein einschränken. Dokumente, Scans, Fotos, 3D-Daten und Analysen – all das sind Rohdaten, die ebenfalls publiziert werden und eine erste Version darstellen können. – Dieser Bereich steht oft auch an erster Stelle des Ablaufs.

Bereich 5: Praktische Vorbereitung der Werkzeuge für die Fundaufnahme. Werkzeuge können sein: Formulare auf Papier für den manuellen Eintrag, Eingabemasken in Datenbanken, Datenblätter einer Tabellenkalkulation, Messgeräte u. a.

Die Bereiche sollten im Sinne der Reproduzierbarkeit gemeinsam mit den eigentlichen Daten als Metainformationen publiziert werden. Das Datenmodell, die Tabellen und ihre Spalten, die Definition der Kategorien von Variablen sowie Spannbreiten von Messdaten, aber auch Informationen zu Datum, Ort der Fundaufnahme und beteiligten Personen ergeben zusammen diese Metainformationen. Sie dienen als Basis für die erste Version der zu erfassenden Daten und können als Definitionstabellen die Datenbank ergänzen.

Nach diesen Vorbereitungen beginnt die routinemäßige, eigentliche Datenerfassung (Stufe C). Zeigt sich dann, dass Theorie (Vorüberlegungen) und Praxis (Datenaufnahme) sich nicht gut vereinbaren lassen, müssen die Vorbereitungen überarbeitet und erneut dokumentiert werden. Die bereits aufgenommenen Daten müssen nach den neuen Kriterien überarbeitet werden.

Praxisbeispiel

Digitale Prozesse, besonders die Arbeit mit Daten, erfordern mehr noch als die Beherrschung von Software theoretisch gut fundierte Vorüberlegungen. Die gegenseitigen Abhängigkeiten von Theorie und faktischen Einschränkungen machen die Arbeit mit Daten besonders spannend, und der folgende Abschnitt soll in die praktische Umsetzung mit aktueller Software einführen. Ergänzend sind die Daten und ein Tutorial im Internet zu finden (Shinoto 2026).[1] Wer die Beispiele selbst nacharbeiten möchte, sollte die Software-GUI *Github Desktop*[2] für den Umgang mit dem Befehlszeilenprogramm *Git*, das in *Github Desktop* enthalten ist, installieren.[3] Zudem werden ein Tabellenkalkulationsprogramm wie *LibreOffice Calc*[4] und ein Texteditor wie *Visual Studio Code*[5] benötigt. Für die Öffentlichkeit und als externes Backup wird ein Account auf einem zentralen Server benötigt, in unserem Fall ist das *Github*[6].

Datensicherung und Einrichtung des Repositoriums

Das stark vereinfachte Beispiel basiert auf einer handschriftlichen Notiz mit Informationen zu neun Keramiken. Zunächst werden die Daten gesichert (Bereich 4), indem der Zettel als Scan oder Foto gespeichert und die enthaltenen Informationen direkt, möglichst unverändert, in eine computerlesbare Tabelle, in unserem Fall in *LibreOffice Calc*, umgetragen werden **(Abb. 1)**.

1 https://doi.org/10.5281/zenodo.20263269

2 Download *Github Desktop*: https://github.com/apps/desktop [zugegriffen am 24.04.2026].

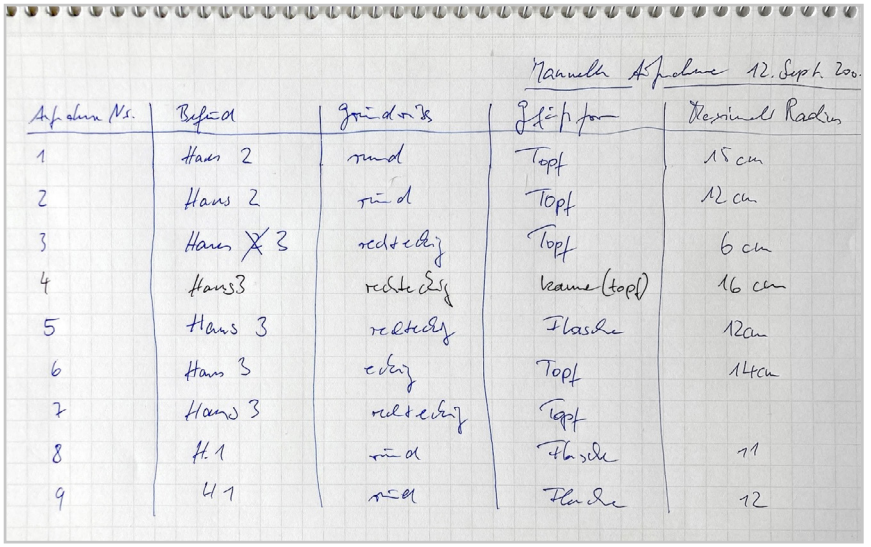
3 Download *Git*: https://git-scm.com/install [zugegriffen am 24.04.2026]. Dieses Programm muss nicht extra heruntergeladen und installiert werden, es wird automatisch mit installiert, wenn man *Github Desktop* installiert.

4 Download *LibreOffice*: https://www.libreoffice.org/download [zugegriffen am 24.04.2026].

5 Download *Visual Studio Code*: https://code.visualstudio.com/download [zugegriffen am 24.04.2026].

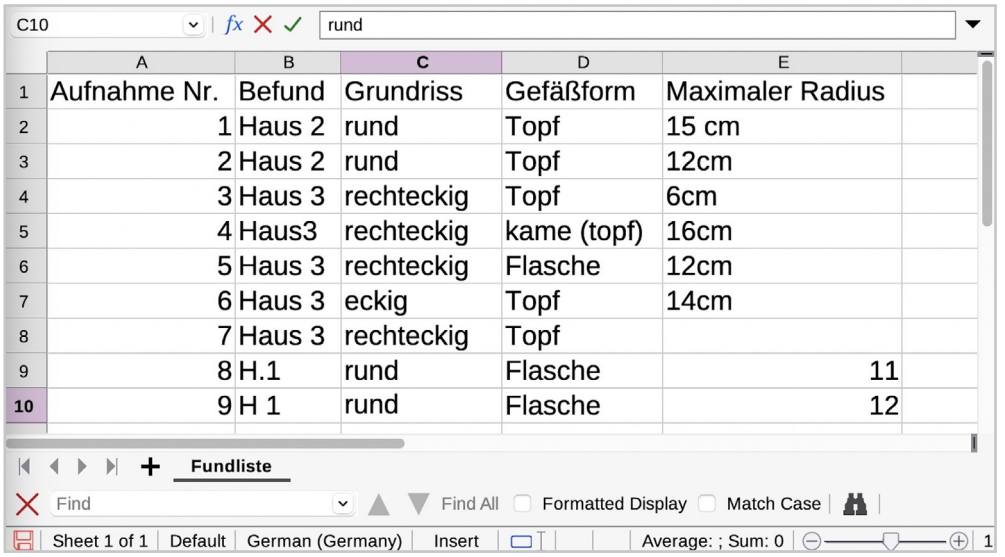
6 Einrichtung eines Kontos auf *Github*: https://github.com/signup?source=form-home-signup&user_email [zugegriffen am 24.04.2026].

a



Aufnahme Nr.	Befund	Grundriss	Gefäßform	Maximaler Radius
1	Haus 2	rund	Topf	15 cm
2	Haus 2	rund	Topf	12 cm
3	Haus X 3	rechteckig	Topf	6 cm
4	Haus3	rechteckig	kame (topf)	16 cm
5	Haus 3	rechteckig	Flasche	12cm
6	Haus 3	eckig	Topf	14cm
7	Haus 3	rechteckig	Topf	
8	H.1	rund	Flasche	11
9	H 1	rund	Flasche	12

b

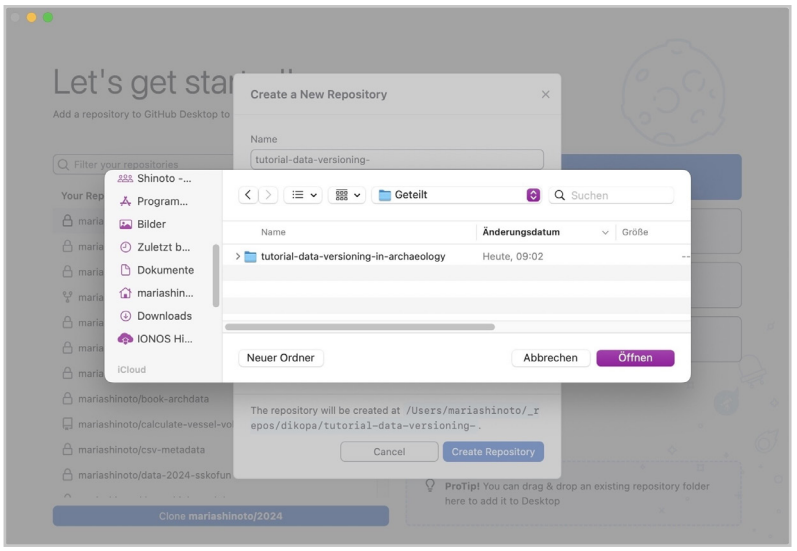


	A	B	C	D	E
1	Aufnahme Nr.	Befund	Grundriss	Gefäßform	Maximaler Radius
2	1	Haus 2	rund	Topf	15 cm
3	2	Haus 2	rund	Topf	12cm
4	3	Haus 3	rechteckig	Topf	6cm
5	4	Haus3	rechteckig	kame (topf)	16cm
6	5	Haus 3	rechteckig	Flasche	12cm
7	6	Haus 3	eckig	Topf	14cm
8	7	Haus 3	rechteckig	Topf	
9	8	H.1	rund	Flasche	11
10	9	H 1	rund	Flasche	12

Abb. 1: Handschriftlich erfasste Rohdaten und in direkter Digitalisierung als computerlesbare Informationen. **a)** Ausgangsdaten, **b)** Umtrag in ein computerlesbares Format.

Der Scan und die Umsetzung in eine computerlesbare Tabelle sollten in der Versionierung festgehalten werden. Hierzu steigen wir mit *Git* und *Github Desktop* ein. Öffnet man *Github Desktop* und ruft den Befehl „New Repository…“ auf, so wird man angewiesen, einen Ordner auszuwählen oder einzurichten, in dem fortan alle Dateien eines Projekts und die Dateien für die Repo-Verwaltung gespeichert werden **(Abb. 2)**. Die Repo-Verwaltung besteht aus Dateien in einem unsichtbaren Ordner „.git“ auf der obersten Ebene des Projektordners. Dieser Ordner pflegt sich selbst, und da er prinzipiell unsichtbar ist, stört er nicht. Wir können ihn ab jetzt vergessen.

a



b

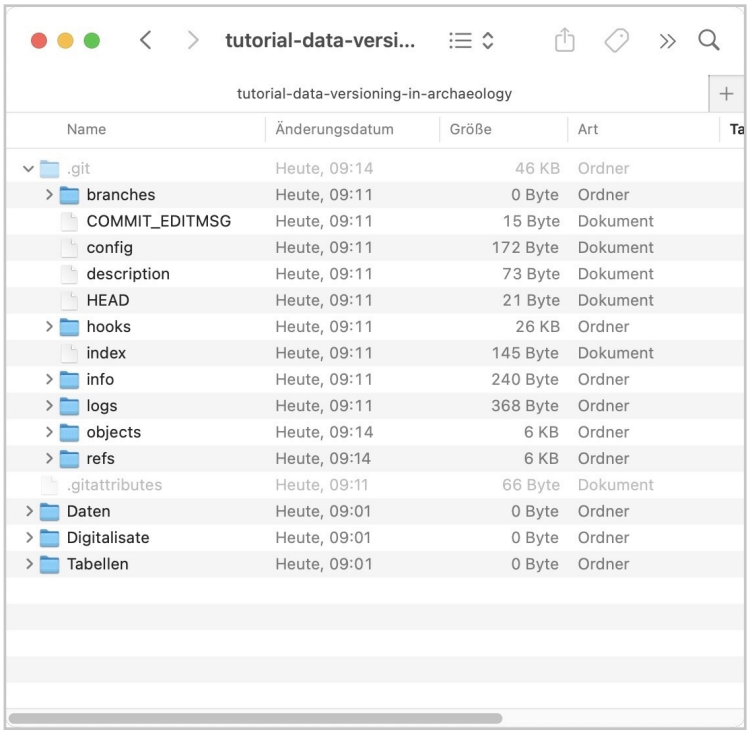


Abb. 2: Das frische *Git*-Repo ohne Daten auf dem eigenen Computer.
a) Einrichtung mit *Github Desktop*, **b)** Dateien im unsichtbaren Ordner „.git“, die das Repo administrieren.

Mit Scan und Tabellendatei gibt es nun schon zwei Varianten der Daten, die in das Repo aufgenommen, also zunächst einmal in den Projektordner gesichert werden müssen. Damit es im Repo übersichtlich bleibt, sollte eine Ordnerstruktur eingerichtet werden: Digitalisate, Tabellen, Daten **(Abb. 3)**.

Zunächst wird nur der Scan im Ordner „Digitalisate“ gesichert. Hier hinein kommen die dem Original am nächsten stehenden Daten, wie Scans oder Fotos. Es sind die Referenzdaten, auf die man immer zurückkommen kann, wenn sich in späteren Versionen Probleme ergeben.

Das Foto der handschriftlichen Notizen ist unser Digitalisat, wir haben es „original-data“ getauft und im TIFF-Format abgespeichert.

Ein paar Anmerkungen: (1) Es empfiehlt sich, auf Englisch zu arbeiten, wenn man auf internationale Publikationen oder Zusammenarbeit hinarbeitet; der Mix aus Deutsch und Englisch in diesem Beispiel ist ein Missgeschick. (2) Dateinamen sollten gut strukturiert sein. Hier wäre es im Hinblick auf die anderen Dateien und die Unterscheidung zu später hinzukommenden Beschreibungsdateien oder Code günstiger, die Datei „data-original“ zu taufen. Das kann in einem späteren Schritt immer geändert und dokumentiert werden. (3) Nehmen die Digitalisate zu viel Platz in einem Online-Repo ein, empfiehlt sich ein Vermerk in einem Textdokument, in dem festgehalten wird, wo die Digitalisate gespeichert – und möglichst auch erreichbar – sind. (4) Die ursprünglichen Digitalisate werden als Referenz für alle weiteren Datenbearbeitungen nur in Ausnahmen verändert und diese Veränderung grundsätzlich dokumentiert.

Momentan besteht unser Repo nur aus dem übergeordneten Projektordner mit drei Ordnern und dem Foto „original-data.tiff“ im Ordner „Digitalisate“. Für *Git* ist alles im Projektordner die *working copy*, der Ort, an dem wir Veränderungen an Dateien vornehmen, die wir von Zeit zu Zeit in einem *commit* registrieren. Leere Ordner erkennt

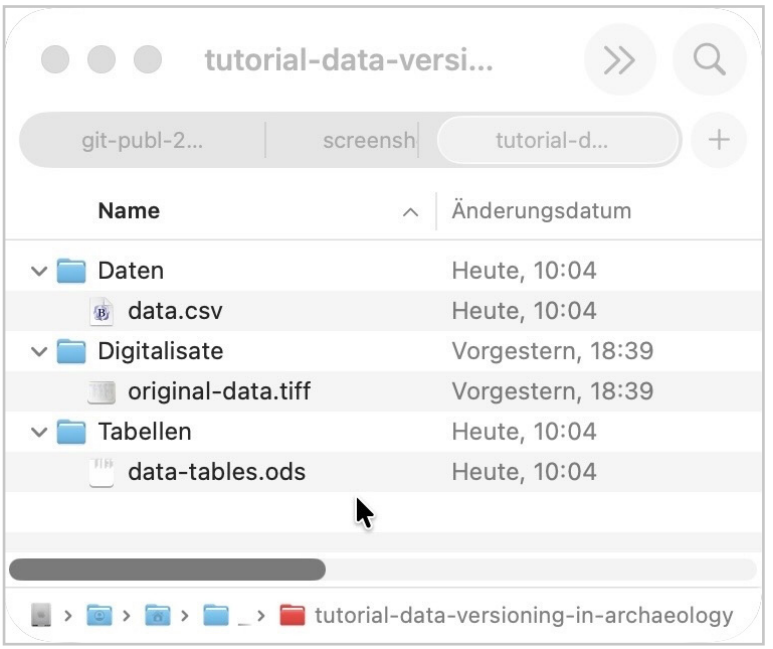
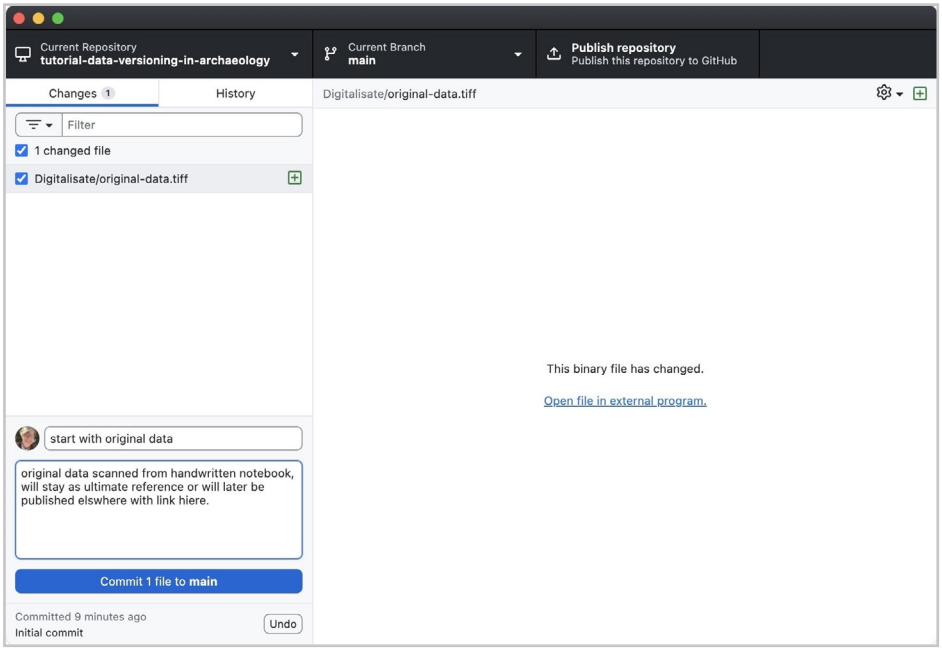


Abb. 3: Ordnerstruktur im *Git*-Repo zu einem späteren Zeitpunkt mit weiteren Daten-Varianten; *Git*-Administrationsdateien verborgen.

Git nicht, nur Dateien und deren Pfade in den Ordnern. Unsere *working copy* hat für *Git* seit der Initialisierung des Repos also nur eine Veränderung gesehen: die Einrichtung des Ordners „Digitalisate“ und die neue Datei „original-data.tiff“, also den Pfad „Digitalisate/original-data.tiff“. **Abb. 4** zeigt, wie *Github Desktop* die Situation darstellt.

a



b

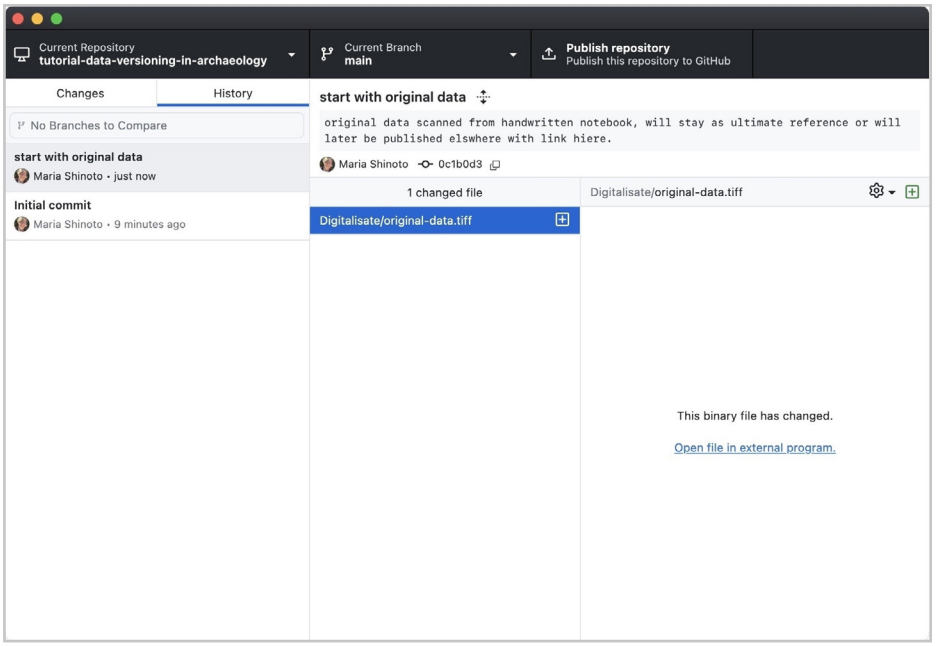


Abb. 4: Die erste Datei wird im Repo versioniert.
a) *Working copy* mit neuer Datei, **b)** Neue Datei nach dem *commit*.

Abb. 4 zeigt in (a) das zweigeteilte Fenster mit neuen, gelöschten oder geänderten Dateien in der linken Spalte. Wir sehen oben in der Spalte, dass der Tab *„Changes“* angezeigt wird. Um die *History* werden wir uns weiter unten kümmern. Die hinzugekommene Datei ist mit einem Häkchen ausgewählt für einen neuen *commit*. Unten in der Spalte sehen wir ein einzeiliges Eingabefeld, in dem eine kurze *message* eingetragen wird, die den *commit* mit wenigen Worten beschreibt. Darunter ist in das größere Textfeld eine längere Beschreibung eingetragen, die helfen soll, die Veränderungen zu verstehen und diesen *commit* später noch besser finden zu können. Wird dann der blaue Knopf mit „*commit* 1 file to main“ geklickt, sind die Veränderungen im Repo registriert, und der Arbeitsbereich, die *working copy*, wird leer. Die Aufschrift des Knopfes richtet sich danach, wie viele Dateien registriert werden sollen, und in welchem Zweig des Repos *(branch)* sie sich befinden. – Unser Beispiel arbeitet nur im automatisch erstellten Zweig *main*. Die rechte Seite des Fensters ist leer, denn ein binäres Format wie bei Bildern oder proprietären Formaten kann *Git* nicht ohne Weiteres lesen. Es erkennt aber, ob es Änderungen gegeben hat.

Abb. 4b zeigt die Situation im Anschluss an den *commit*: Der Arbeitsbereich ist leer, und die *History* wird angezeigt. Wir sehen in der linken Spalte nach einem automatisch erstellten „Initial Commit“ als zweiten *commit* den eben erstellten mit unserer kurzen *Message*. Der rechte Teil ist nun in zwei Spalten geteilt: Links sind die Dateien, die in diesem *commit* registriert wurden, aufgelistet – in unserem Fall nur eine. Rechts davon wäre ein *diff*, die Darstellung der genauen Änderungen, zu sehen, die bei dem binären TIFF nicht möglich ist. Über beide Bereiche sind jedoch weitere Informationen angegeben: (1) die kurze Message, (2) die längere Erläuterung, (3) die Verantwortliche für das *commit*, (4) die Anfangsstellen einer längeren, eindeutigen Identifikationsnummer dieses *commits* im Repo.

Dieser ständige Wechsel von Veränderungen in den Dateien im *work space* – oder *working copy* oder Arbeitsbereich – und der Registrierung der Änderungen in einem *commit* ist der Kern der Versionskontrolle.[7]

Wir haben die Digitalisate nun gespeichert und in einem *commit* registriert. Dieser Schritt ist bedeutsam und kann mit einer von den *commits* unabhängig laufenden Versionierung mit einer Versions-ID kenntlich gemacht werden; später können hieraus *releases*, also Veröffentlichungen erstellt werden. Für die Versionsbezeichnungen gibt es keine Restriktionen, aber es empfiehlt sich ein Schema, das aus der Softwareentwicklung übernommen werden kann: *semantic versioning*[8]. Hierbei gibt es eine

7 Eine Zusammenstellung von Literatur und Tutorials findet sich auf dem offiziellen Server von *Git*: https://git-scm.com/learn [zugegriffen am 24.04.2026]. *Github* und andere Anbieter bieten ebenfalls sehr gute Einführungen, die aber zum Teil auf die jeweiligen Anbieter zugeschnitten sind und das Verständnis der Prinzipien verwässern können.

8 https://semver.org/lang/de [zugegriffen am 24.04.2026].

Hierarchie von drei Stufen: *Major*, *Minor* und *Patch*, die durch eine aufsteigende Numerierung in drei Sektoren beginnend mit „0.0.0“ gekennzeichnet werden. Es steht jedem frei, die Bedeutung der Schritte in der eigenen Forschung nach eigenen Kriterien zu bewerten.

In unserem Beispiel wollen wir die erste *Major*-Version mit der Nummer 1.0.0 erst später vergeben, wenn nämlich die Daten korrekt modelliert, standardisiert und klassifiziert sowie mit hilfreichen Metadaten ausreichend versehen sind und als Textdateien frei verfügbar im Repo stehen. Das Speichern von Digitalisaten ist aber genügend bedeutsam, um eine *Minor* Version zu vergeben; in unserem Falle v0.1.0. Hierzu müssen wir den gerade erstellten *commit* mit einem Etikett *(tag)* versehen, das die Versionsnummer enthält **(Abb. 5)**.

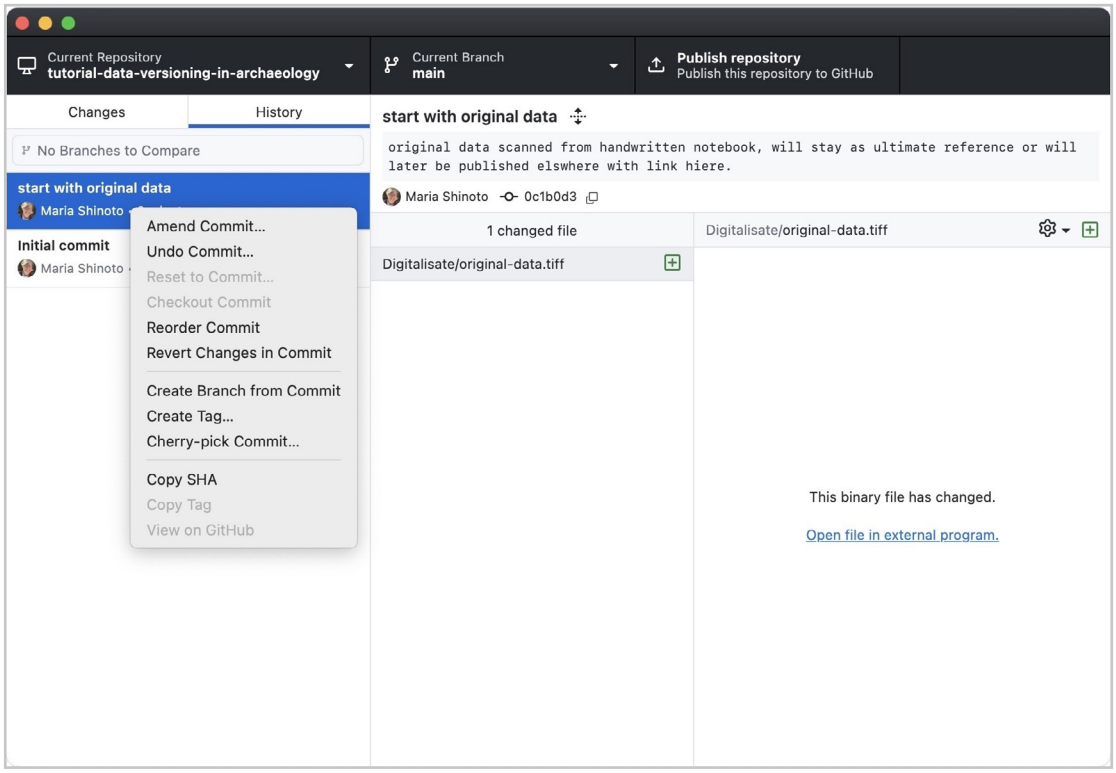


Abb. 5a: Vergabe einer Versionsnummer mit Etiketten (*tags*) und Ansicht in der *History*. Etikett (*tag*) mit Rechtsklick auf ein *commit*.

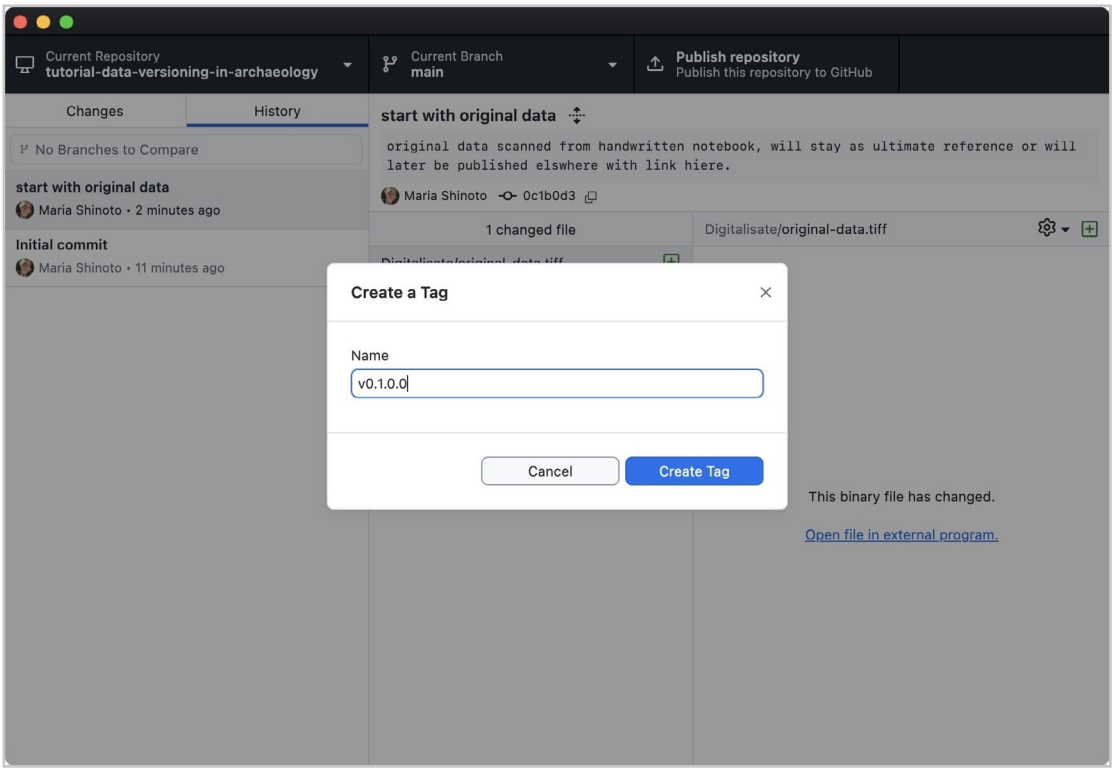


Abb. 5b: Dialog zur Benennung.

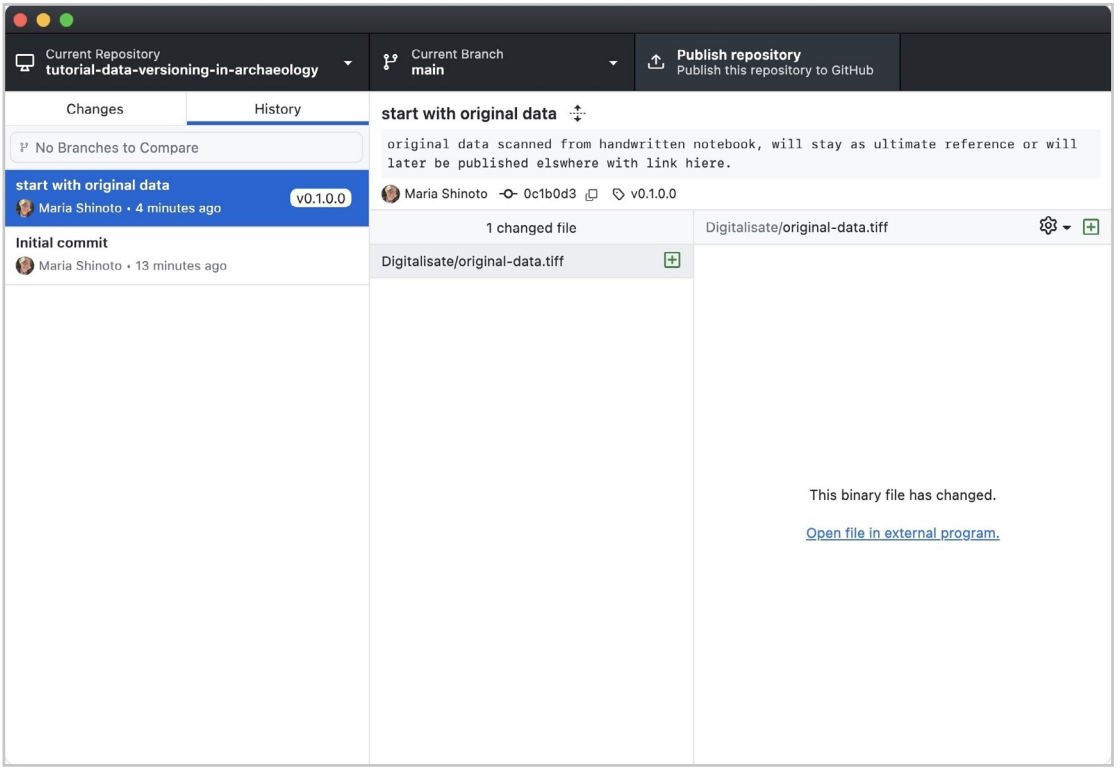


Abb. 5c: Ansicht in der *History*.

Die weitere Arbeit erfolgt in derselben Art: (a) Es wird in der *working copy* – faktisch im Ordner auf dem Computer – an den Dateien gearbeitet, (b) in sinnvollen, möglichst kleinteiligen Abschnitten werden die Änderungen überprüft und als *commit* registriert, (c) an sinnvollen *commits* wird ein *tag* vergeben, der als Versionsnummer dienen kann.

Zur Sicherung unserer Daten können wir nun die Tabelle „data-tables.ods", die wir in *LibreOffice Calc* nach der handschriftlichen Vorlage eingegeben haben, in den Ordner „Tabellen" geben und auf gleiche Weise registrieren. Dieser *commit* bekommt die *Message* „Transfer handwriting to spreadsheet". Auch diese Tabelle ist im binären Datenformat gespeichert und kann von *Git* nur als „geändert" erkannt, inhaltlich jedoch nicht dargestellt werden.

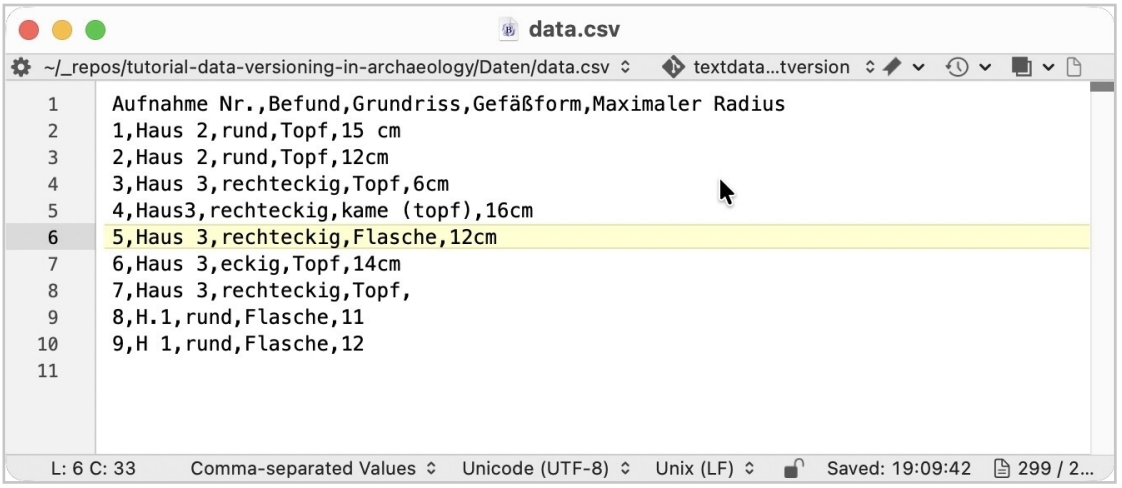
Umsetzung in Text

Text ist das Dateiformat, das auf Computern jedweder Art nachhaltig gespeichert und dargestellt werden kann, und es kann fast alles darstellen, was sonst in proprietären Datenformaten gespeichert wird. Da Versionskontrolle zuallererst für Code, also Text, gedacht ist, können Daten, die als Text gespeichert werden, sehr detailliert in Versionskontrolle entwickelt und dargestellt werden (Shinoto 2025).

Für unsere erste „echte" Datenversion im Textformat exportieren wir die Tabelle „data-tables.ods" im Ordner „Tabellen" aus dem Spreadsheet in *LibreOffice Calc* in eine Textdatei im Unicode-Format (UTF-8). Wir nennen die neue Datei im Ordner „Daten" „data.csv". Diese Datei lässt sich grundsätzlich in jedem Texteditor lesen und bearbeiten und auch in Tabellenkalkulationen, Datenbanken und Statistikprogramme importieren.

Jetzt hat unser Projektordner den Zustand in **Abb. 3** erreicht. Die Textdatei stellt die Daten wie von den handschriftlichen Notizen eingegeben dar, die Spalten sind durch Kommata getrennt, sodass die übersichtliche Form einer Tabelle nicht mehr gegeben ist **(Abb. 6a)**. *Git* zeigt ein *diff* zur vorigen Version der Datei „data.csv" **(Abb. 6b)**. Da die Datei vorher nicht vorhanden war, zeigt der *diff* nur grüne Ergänzungen für neu hinzugekommene Zeilen. Hätte die Datei vorher schon bestanden und wären Zeilen gelöscht worden, fänden sich rote Einträge links der Ergänzungen, und Änderungen in einer Zeile würden als eine Kombination aus roten Löschungen und grünen Ergänzungen dargestellt **(s. u. Abb. 7b)**.

a



b

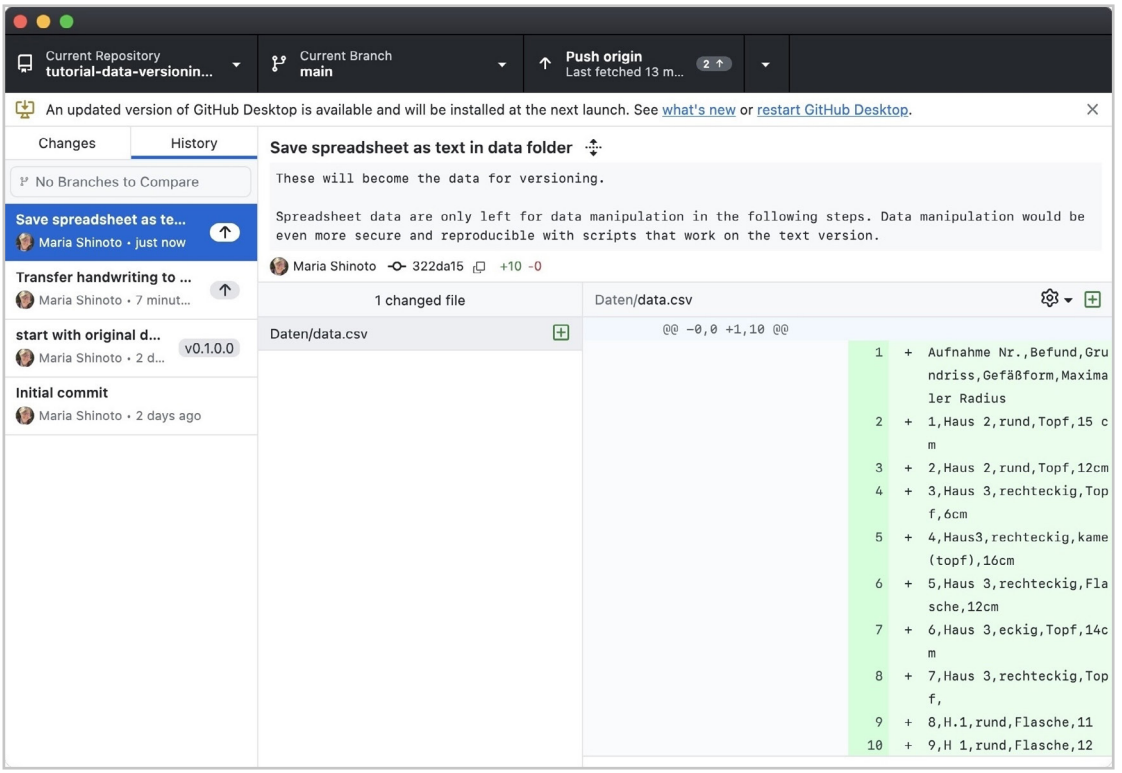


Abb. 6: Ansicht der Textdaten in einem Texteditor und Interpretation in Versionskontrolle als neu hinzugekommene Zeilen (grün). **a)** Ansicht im Texteditor, **b)** Ansicht in der Versionskontrolle.

Da wir jetzt alle Daten *verbatim* so gespeichert haben, wie wir sie aus der handschriftlichen Aufnahme verstanden haben, können wir nun eine neue *minor version* vergeben (0.2.0) und uns daran machen, die Daten nach allen Regeln der Kunst zu bearbeiten und nutzbar zu machen.

Datenmodellierung – Datenformate – Metadaten

Nun kommt es zur Umsetzung der Bereiche 2 (Datenmodellierung) und 3 (Datenbereinigung, Klassifikationen und Codierungen). Die in **Abb. 1** dargestellte Ausgangstabelle müssen wir uns hierzu genauer ansehen.

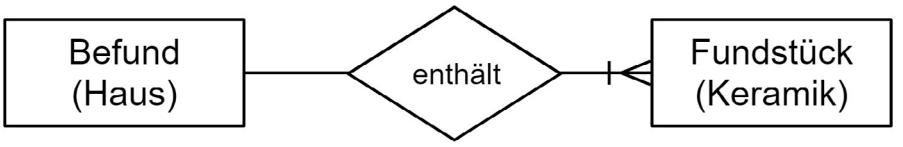
- Spalte 1 zeigt offenbar einmalige Identifikationsnummern, die durchgezählt werden und sich wahrscheinlich auf Gefäße (Keramik?) beziehen, deren Gefäßform und maximaler Radius in den vierten und fünften Spalten erfasst sind.
- Es fällt gleich auf, dass der maximale Radius zum Teil mit einer Maßeinheit erfasst wurde, zum Teil nur als Zahl. Da sich alle Größen in einem gemeinsamen Rahmen bewegen, kann man davon ausgehen, dass alle Angaben in Zentimetern ausgewiesen sind. Eine Angabe fehlt allerdings bei Nr. 7. Konnte man diesen Wert nicht messen, oder wurde es vergessen?
- Zur Gefäßform fällt auf, dass es nur zwei Klassen gibt: „Topf“ und „Flasche“. Der Wert zu Nr. 4, „kame (topf)“ erfasst den japanischen Begriff mit deutscher Übersetzung in falscher Orthographie. Wir können davon ausgehen, dass es sich ebenfalls um einen „Topf“ handelt.
- Zu allen neun Gefäßen sind die Befunde erfasst, allerdings nicht konsistent. Man darf aber davon ausgehen, dass es sich um drei Hausgrundrisse mit den Identifikationsnummern 1 bis 3 handelt. In Zweifelsfällen wie „Haus3“ oder „H.1“ und „H 1“ können die Angaben zum jeweiligen Grundriss etwas Sicherheit geben.

Um die Daten konsistent zu gestalten, bietet sich eine dokumentierte Überarbeitung an, wobei die Befunde nach dem Schema „H01“ nummeriert werden, die Grundrisse auf entweder „rechteckig/eckig“ oder „rund“ vereinheitlicht werden und die Gefäßformen auf die Kategorien „Topf“ und „Flasche“. Die einzelnen Maße des maximalen Radius der Gefäße werden nur noch als Zahlen gespeichert und die Maßeinheit getrennt erfasst, möglicherweise vorerst nur in der Dokumentation.

Diese Änderungen können in der Textdatei oder im Kalkulationsprogramm vorgenommen werden. Wir verwenden *LibreOffice Calc*, verdoppeln das ursprüngliche Datenblatt, führen einen zusammenhängenden Satz von Arbeitsschritten aus, verdoppeln das Blatt wiederum und bearbeiten es, bis alle Schritte, die sich aus der Diskussion der Tabelle ergeben haben, durchgeführt sind. An den Namen der Blätter kann man die Entwicklung ablesen und jederzeit Schritte zurückgehen. Dieses Vorgehen ist im Gegensatz zur Arbeit direkt in der Textdatei übersichtlicher und hilft bei der Fehlervermeidung. Schließlich werden die überarbeiteten Daten in die Datei „data.csv“ exportiert, die sich dadurch ändert. Ein neues *commit* wird notwendig, und es empfiehlt sich eine *minor version* zu vergeben, da die Daten jetzt nicht mehr im Urzustand vorliegen. Dafür sind sie aber zunehmend für systematische und vergleichende Arbeiten geeignet.

Die Daten als solche sind jedoch nicht sehr aussagekräftig, die Überlegungen zu den originalen Vorlagen und Entscheidungen hinsichtlich des Charakters der Daten (Definition der Variablen, Maßeinheiten, statistische Messniveaus u. a.) sollten sich in Metadaten widerspiegeln. In öffentlichen Datenrepositorien, die Workbooks von Tabellenkalkulationen bereithalten, werden meist weitere Datenblätter mit solchen Metadaten angehängt. In unserem Fall können wir die Metadaten als kommentierte Zeilen (man kann hierzu das Zeichen „#“ zur Kennzeichnung wählen) direkt in die Daten-Dateien eingeben (Metadaten in der Textdatei s. **Abb. 8b**), um diese Daten beim Import in Analysen herauszufiltern und gleichzeitig die Informationen zu nutzen.

a



b

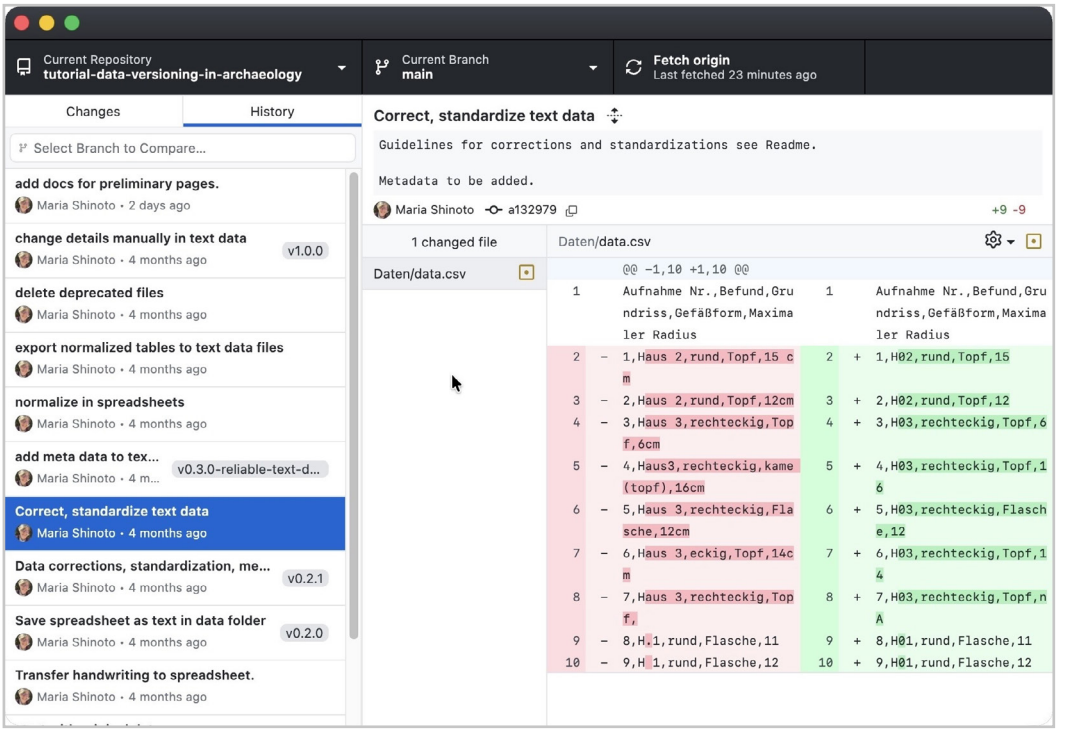


Abb. 7: Datenmodell und Darstellung der Entstehung in der *History* des Repos. **a)** Relationales Datenmodell, **b)** *History* der Rohdaten über Modellierung und Standardisierung bis Version 1.0.0. (*commit* „a132979“ (links, blau unterlegt) zeigt das *diff* der überarbeiteten Textdaten, Version v0.3.0 ergänzt Metadaten, die *major version* v.1.0.0 ist erst nach vollständiger Datenmodellierung und Umsetzung in mehrere Tabellen erreicht.)

Nun kennen wir die Daten gut genug, um ein Datenmodell zu erstellen und den Datenbestand demgemäß zu „normalisieren“ (Hernandez 2020). Der hiesige Fall ist ausgesprochen einfach: Wir haben zwei Entitäten, die in einer 1:n-Beziehung stehen **(s. Abb. 7a)**. Dabei beziehen sich die Spalten 1 (Aufnahme Nr.), 4 (Gefäßform) und 5 (Maximaler Radius) auf die Entität der Gefäße, die Spalten 2 (Befund) und 3 (Grundriss) auf die Entität der Befunde. Es gibt n Gefäße in 1 Befund, die Kennung des Befundes (H01 bis H03) muss daher als zusätzliche Spalte in die Tabelle zu den Gefäßen mit aufgenommen werden. So kann immer auf zusätzliche Informationen zu den Befunden (hier die Grundrisse) zugegriffen werden; auch dies ist über Statistik- und andere Software genauso möglich wie in einer spezialisierten relationalen Datenbank.

Nachdem das Konzept erarbeitet und die Entscheidung, welche Spalten in welche Tabellen gehören, getroffen worden sind, können die Spalten mit den jeweiligen Metadaten in neue Dateien gespeichert werden, die nun die Version 1.0.0 bilden **(Abb. 7)**.

Abb. 8 zeigt die Situation zur schrittweisen Bearbeitung in der Tabellenkalkulation (a), die exportierte Textdatei mit Metadaten zu Befunden, die jetzt „contexts.csv“ genannt wird (b), und die Möglichkeit, die Metadaten in der Textdatei weiter zu verbessern und in *commits* zu registrieren (c).

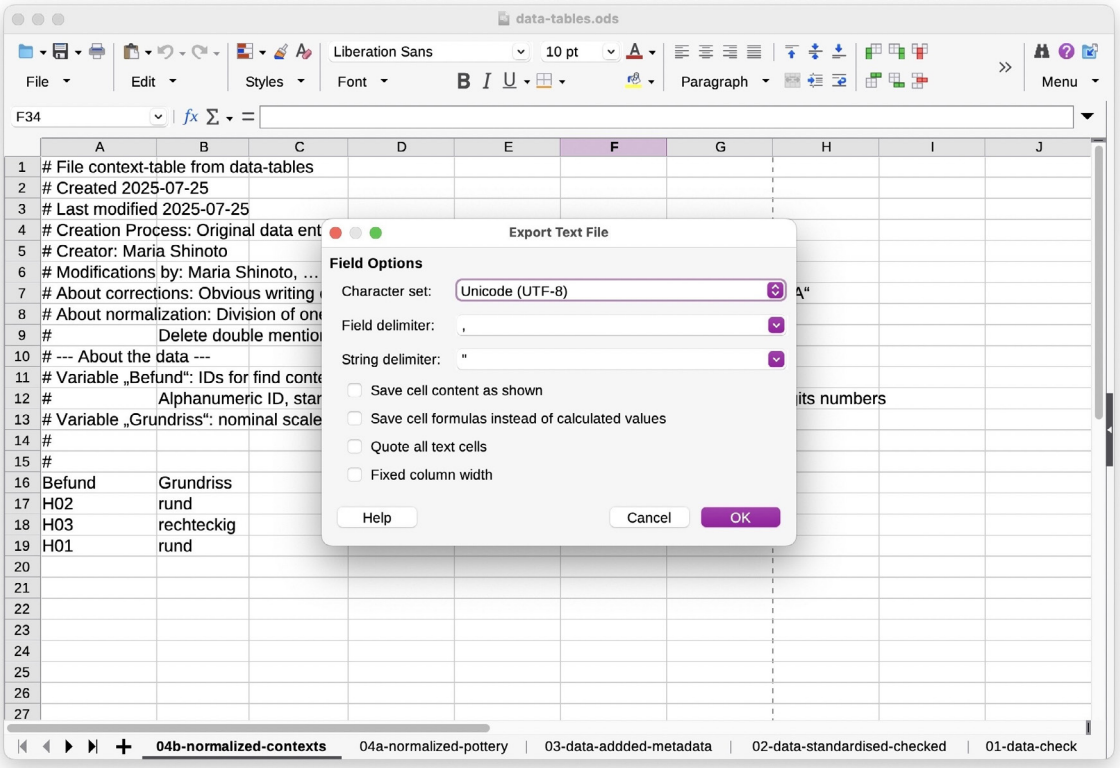


Abb. 8a: Ansicht der Textdaten in einem Texteditor und Interpretation in Versionskontrolle als neu hinzugekommene Zeilen (grün). Metadatenbearbeitung in *LibreOffice Calc.*

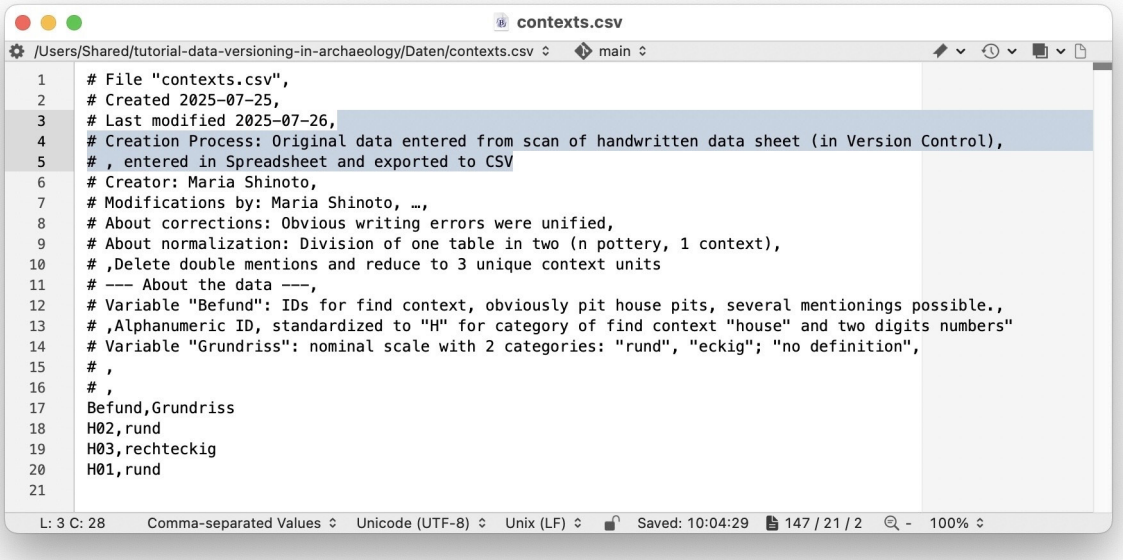


Abb. 8b: Ansicht in der exportierten Textdatei.

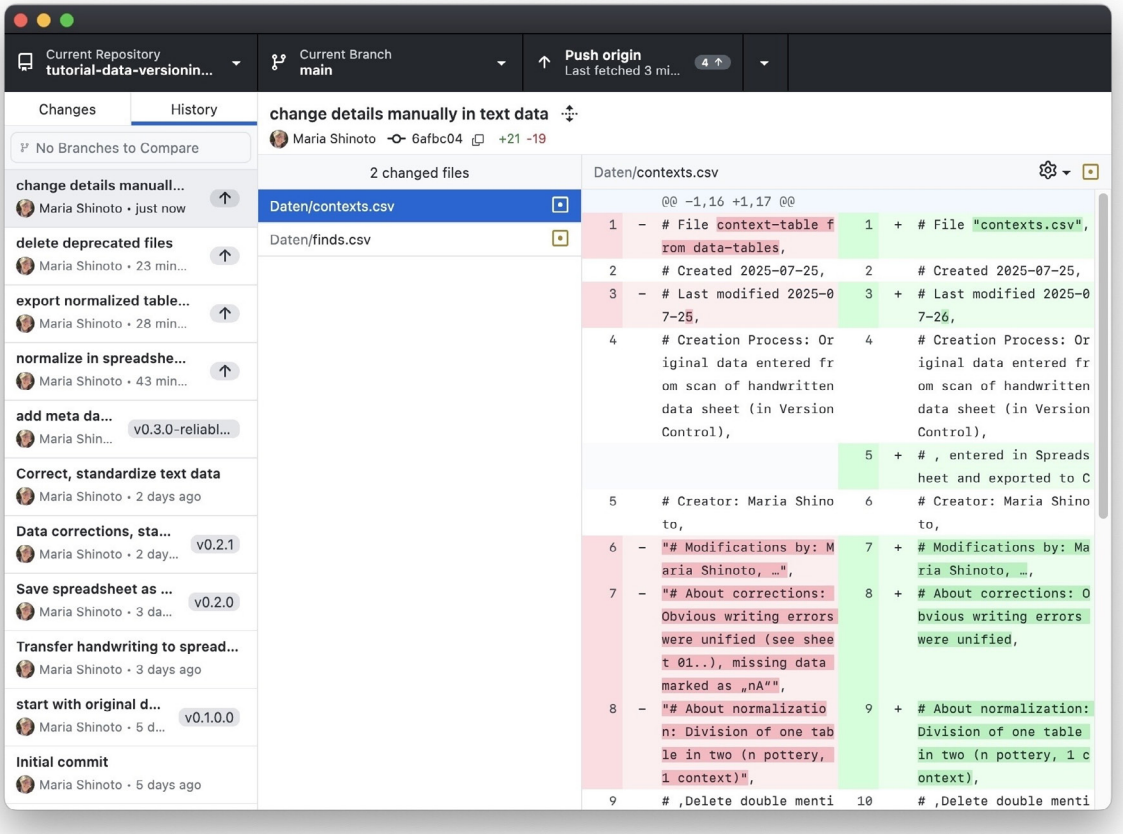
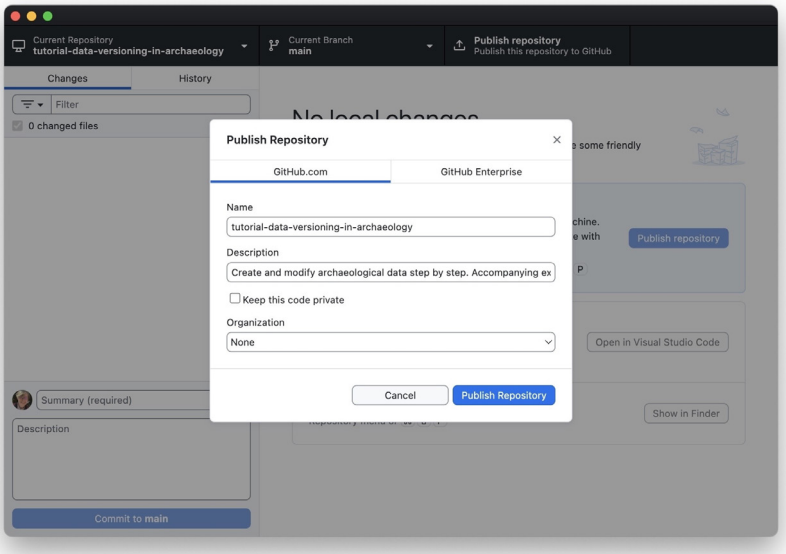


Abb. 8c: Inkrementelle Verbesserungen und Patches.

Öffentlichkeit und Zusammenarbeit über Github

Im momentanen Zustand ist die Entstehung des Datensatzes nur auf dem eigenen Computer nachvollziehbar; sie soll aber öffentlich reproduzierbar sein. Hierzu müssen die Daten online gestellt werden, und zwar auf einem *Git*-Server, den man durchaus selbst einrichten kann. Allerdings bieten die drei großen Services *Github, GitLab* und *Bitbucket* mehr Komfort und Zusatzfunktionen. Es gibt neuere Anbieter wie *Codeberg*, die künftig Unabhängigkeit von Technologiekonzernen und Speicherung in der EU ermöglichen könnten. *Github* ist jedoch die älteste Plattform mit der größten Verbreitung; die meisten Tutorials zu digitalen Verfahren und *Git* nutzen *Github*. Daher sei hier empfohlen, erst einmal mit diesem Service zu beginnen.[9]

a



b

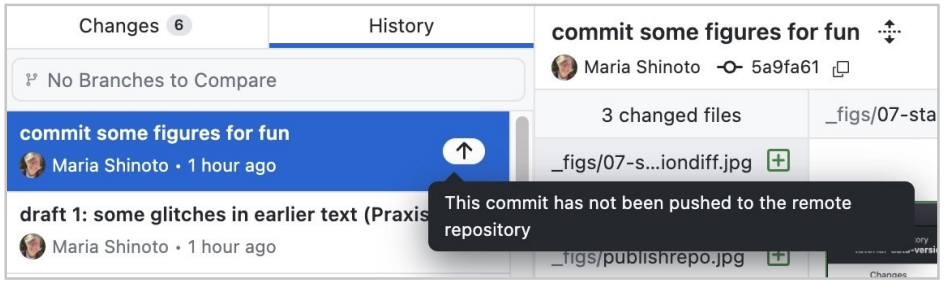


Abb. 9: Veröffentlichung des Repos in einfachen Schritten. a) Dialog nach Klick auf „Publish Repo“, b) Pfeil in der *History* in *Github Desktop*, der Änderungen auf den zentralen Server schickt (aus anderem Repo).

9 Github: https://github.com, Gitlab: https://about.gitlab.com, Bitbucket: https://bitbucket.org, Codeberg: https://codeberg.org [zugegriffen am 24.04.2026].

Die Schritte in diesem Beispiel sehen wie folgt aus: (1) Es muss ein *Github*-Account eingerichtet werden, (2) das Repo auf dem eigenen Computer muss mit dem Account verknüpft und dort das zentrale Repo eingerichtet werden, (3) Änderungen werden mit *push* vom lokalen Computer oder *pull* vom zentralen Server ausgetauscht, sodass beide Versionen auf einem Stand bleiben.

1) Die Einrichtung des Accounts erklärt sich von alleine, wenn man die Homepage von *Github* besucht. Ein Beispiel ist der Account der Autorin, auf dem auch die Beispieldaten aus diesem Tutorial und weitere Informationen und Diskussionen öffentlich zugänglich sind[10].
2) Es gibt zahlreiche Arten, zentrale und lokale Repos einzurichten und miteinander zu verbinden. Hier haben wir damit begonnen, ein lokales Repo einzurichten, und dieses können wir nun mit dem Befehl „Publish Repository" in *Github Desktop* auf den Server hochladen und auf *public* stellen, um es sichtbar zu machen.
3) Von nun an kann jedes *commit* oder in Abständen eine Abfolge von *commits* auf den Server hochgeladen werden *(push)*.

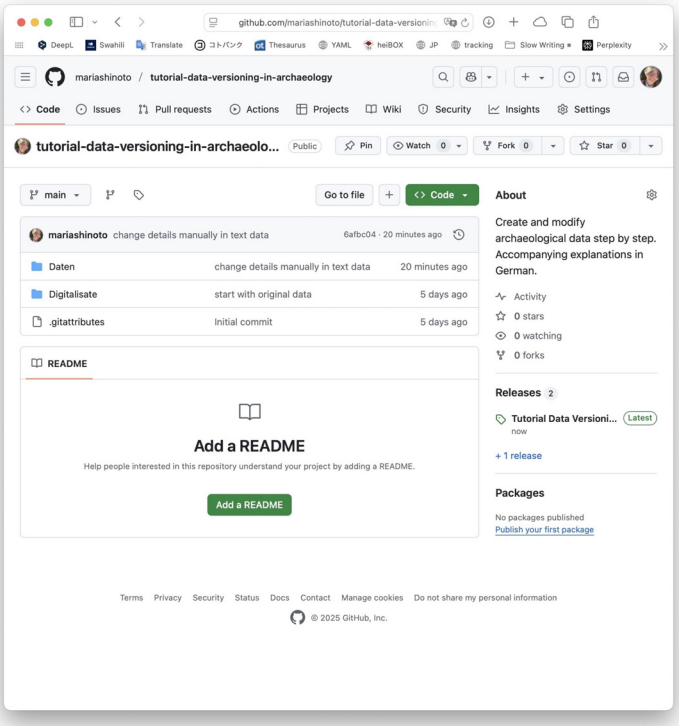
Die Möglichkeiten von *Git* alleine sind vielfältig, die Nutzung von *Github* und ähnlichen Services ist noch komplexer.

Unabhängigkeit und Reproduzierbarkeit

Dieses Tutorial hat die am leichtesten zugängliche Software vorgestellt, insbesondere *Visual Studio Code, Github Desktop* und den *GitHub*-Service. Es gibt empfehlenswerte Alternativen, die Unabhängigkeit und Datensicherheit ermöglichen. Der Einstieg mit den genannten Werkzeugen empfiehlt sich dennoch, da fast alle Erklärungen und Tutorials im Internet hierauf aufbauen. Die Basis der vorgestellten Prozesse sind offene Formate, die nicht von proprietärer Software abhängig sind, die Datenformate als Text und die Prinzipien der Vorgehensweise bleiben dieselben und sind universal. Dies ist neben der offenen Versionierung die Voraussetzung für Reproduzierbarkeit.

10 Account: https://github.com/mariashinoto [zugegriffen am 24.04.2026].

a



b

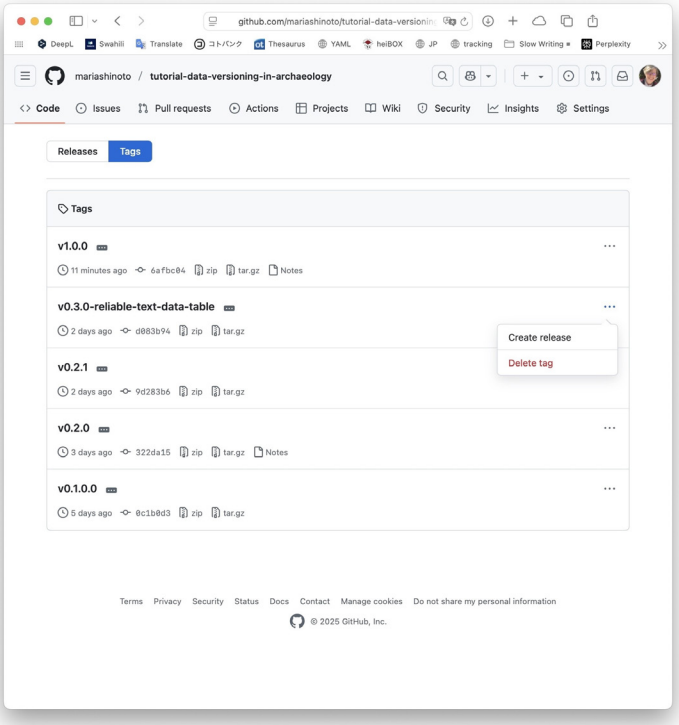


Abb. 10: Veröffentlichung des Repos in einfachen Schritten.
a) Homepage des Repos nach der Veröffentlichung, **b)** Erstellung von Releases aus Etiketten (*tags*).

ORCID®

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

Referenzen

Codd, Edgar F. (1970): A relational model of data for large shared data banks, in: *Communications of the ACM 13*, New York: Association for Computing Machinery, S. 377–387.

Hernandez, Michael J. (2020): *Database design for mere mortals, 25th anniversary edition: a hands-on guide to relational database design*, 4. Aufl. Hoboken: Pearson Education, Inc.

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. Chancen und Herausforderungen*, Heidelberg: Propylaeum, S. 95–108, [online] https://doi.org/10.11588/propylaeum.1572.c23243.

Shinoto, Maria (2026): *Material zu einem Tutorial: Reproduzierbarkeit mit Versionskontrolle (v1.0.1)*. Zenodo. https://doi.org/10.5281/zenodo.20263269.