Mit welchem Aufwand bekommen wir Skripte FAIR(er)?

Denis Arnold und Christian Lang

Leibniz-Institut für Deutsche Sprache

In diesem Beitrag widmen wir uns der Frage, welche Schritte unternommen werden müssen, um Skripte, die bei der Aufbereitung und/oder Auswertung von Forschungsdaten Anwendung finden, so FAIR wie möglich zu gestalten. Dabei nehmen wir sowohl Reproduzierbarkeit, also den Weg von den (Roh)daten zu den Ergebnissen einer Studie, als auch Wiederverwertbarkeit, also die Möglichkeit, die Methoden einer Studie mittels des Skripts auf andere Daten anzuwenden, in den Fokus und beleuchten dabei die folgenden Aspekte: Arbeitsumgebung, Datenvalidierung, Modularisierung, Dokumentation und Lizenz.

1 Einleitung

Die Offenheit von Forschungsdaten ist spätestens seit der Postulierung der TOP-Guidelines im wegweisenden Papier von Nosek et al. [1] ein oft wiederholtes und in weiten Teilen noch unerfülltes Desideratum, das beispielsweise auch die Deutsche Forschungsgemeinschaft im Kodex "Leitlinien zur Sicherung guter wissenschaftlicher Praxis" [2] und im Impulspapier "Digitaler Wandel in den Wissenschaften" [3] adressiert.

Ähnlich gelagert, jedoch weniger als die TOP-Guidelines auf völlige Offenheit fokussiert, sind die von Wilkinson et al. [4] 2016 formulierten FAIR-Prinzipien. Demnach sollen Forschungsdaten auffindbar (**F**indable), zugänglich (**A**ccessible), interoperabel (**I**nteroperable) und nachnutzbar (**R**eusable) sein. Ziel der FAIR-Prinzipien als Leitlinien guten Datenmanagements sind die Vereinfachung des Auffindens, der Evaluation und der Weiterverwendung von entsprechend publizierten Daten [4, S.1].

Gutes Datenmanagement als Herausforderung für Wissenschaftlerinnen und Wissenschaftler datenintensiver Disziplinen[1] berührt verschiedene Bereiche des wissenschaftlichen Prozesses. Im Zentrum einer Vielzahl an Diskussionen steht dabei oftmals die Schaffung von Repositoriumsinfrastrukturen, die es – entsprechende Metadaten vorausgesetzt – ermöglichen, möglichst flexibel Forschungsdaten der (wissenschaftlichen) Öffentlichkeit zur Verfügung zu stellen. In unserem Beitrag wollen wir dagegen den Fokus auf einen Aspekt datengetriebener Forschungsarbeit legen, der in der Diskussion um die Offenheit von

[1] Durch die digitale und empirische Wende lassen sich zunehmend auch Teile der klassischen Geisteswissenschaften als datenintensive Disziplinen einordnen.

Publiziert in: Vincent Heuveline, Nina Bisheh (Hg.): E-Science-Tage 2021. Share Your Research Data. Heidelberg: heiBOOKS, 2022.
DOI: https://doi.org/10.11588/heibooks.979.c13746

Forschungsdaten oftmals nur am Rande oder gar nicht erwähnt wird. Genauer gesagt, betrachten wir Skripte, die Forscherinnen und Forscher zur Datenaufbereitung und Datenauswertung schreiben. Unserer Erfahrung nach werden diese vielfach als Mittel zum Zweck auf dem Weg zum Ergebnis und nicht als integraler Bestandteil der Forschungsdaten und einer offenen Publikationsstrategie betrachtet. Dabei schließen Wilkinson et al. in ihre FAIR-Prinzipien explizit auch sogenannte "non-data assets" ein und nennen in diesem Kontext Algorithmen, Tools und Workflows, die zu den Forschungsdaten geführt haben und auf die ebenso die FAIR-Prinzipien anzuwenden seien [4, S. 4f.]. Auch im Impulspapier "Digitaler Wandel in den Wissenschaften" [3, S. 10] wird "[...] der Zugang zu Daten und Software für die Wissenschaften [...] nach dem FAIR-Prinzip [...]" als Herausforderung benannt und auch im Kodex "Leitlinien zur Sicherung guter wissenschaftlicher Praxis" [2] wird auf die Bedeutung von "(Forschungs)software" verwiesen. Skripte als eigene Gattung werden hier oder auch an anderen Stellen nicht explizit ausgezeichnet, aber es ist davon auszugehen, dass auch sie unter diese Gattung fallen, auch wenn sie häufig nicht als vollwertige Software angesehen werden.

Auch Nosek et al. [1] fordern neben der Offenheit der Daten(grundlage) und der Ergebnisse unter dem Stichwort "Analytic methods (code) transparency" eine entsprechende Offenheit von Methoden und Code, die im idealen Fall (also der höchsten Transparenzstufe) darin besteht, dass der Code in einem entsprechenden Repositorium abgelegt und die Methoden vor der Publikation unabhängig reproduziert werden [1, S. 1424].

Was kann also getan werden, um auch Skripte FAIR(er) zu gestalten? Im Zentrum unseres Interesses steht dabei weniger die Frage, wie Skripte als Forschungsdaten veröffentlicht werden können[2] und welche Anforderungen an entsprechende Repositorien zu stellen sind. Vielmehr konzentrieren wir uns auf die Beschaffenheit der Skripte an sich und fragen uns, was bei deren Erstellung und bezüglich ihrem Aufbau getan werden kann, damit diese für den Fall, dass sie in einem Repositorium gemeinsam mit den Forschungsdaten abgelegt werden, von einem möglichst breiten wissenschaftlichen Publikum nachvollzogen und wiederverwendet werden können. Dabei dienen uns die von Wilkinson et al. formulierten FAIR-Prinzipien als grundlegende Orientierung mit einem besonderen Fokus auf die Interoprabilität (**I**nteroperable) und Wiederverwertbarkeit (**R**eusable) von Skripten.

2 Aspekte

Im Folgenden gehen wir auf verschiedene Aspekte ein, die bei der Erstellung von Aufbereitungs- und Analyseskripten Anhaltspunkte für eine FAIRe(re) Gestaltung im Allgemeinen und eine Erhöhung der Wiederverwendbarkeit im Besonderen bieten. Hierbei versuchen wir eine von konkreten Sprachen unabhängige Perspektive einzunehmen.

[2] Hier bieten Zenodo (https://zenodo.org), Open Science Framework (OSF) (https://osf.io), aber auch Fachrepositorien die Möglichkeit, Skripte auffindbar zugänglich zu machen (zur Kritik an einem vorhandenen/entstehenden Repositorien-Pluralimus siehe [4, S. 2]).

Infolgedessen gehen wir nicht auf spezifische Weisen der Code-Optimierung ein, sondern fokussieren uns auf übergeordnete, allgemeine und (weitestgehend) sprachenunabhängige Aspekte von Skripten.

2.1 Arbeitsumgebung

Skripte werden für bestimmte Sprachen oder Programme geschrieben und sind nicht ohne Weiteres in anderen Versionen lauffähig. Skriptsprachen sind von Interpretern abhängig, die es häufig ermöglichen, den gleichen Code auf verschiedenen Betriebssystemen auszuführen. Trotzdem unterscheidet sich der standardmäßige Gebrauch z. B. von Encodings (etwa Windows-1252 vs. UTF-8) oder auch der Umgang mit Systembibliotheken zwischen verschieden Interpretern und Betriebssystem zum Teil stärker, als dass man sie vernachlässigen könnte. Nahezu alle Skriptsprachen werden durch umfangreiche Bibliotheken besonders attraktiv, die ihrerseits fortlaufender Entwicklung unterliegen. Ein wichtiger Schritt stellt somit eine Zusammenstellung der beteiligten Programme und Bibliotheken und ihren Versionen dar. Viele Skripte benutzen in ihrem Workflow weitere Skripte, Programme (z. B. Datenbanken) oder auch Onlinedienste. Neben der Benennung der Werkzeuge sind außerdem Konfigurationen und Interaktionen mit den Schnittstellen wichtige Bestandteile der Arbeitsumgebung.

2.2 Datenvalidierung

2.2.1 Prüfung auf Datengleichheit

Für eine Reproduktion muss die Datengrundlage verifizierbar sein. Dies ist umso wichtiger, wenn die Daten nicht frei im Sinne von Open, sondern geschützt im Sinne von FAIR sind. Hierzu eigenen sich beispielsweise Listen mit Prüfsummen. Pfade zu Dateien und anderen Programmen sollten in jedem Fall relativ angegeben sein und es sollte in den Schnittstellen die Möglichkeit gegeben sein, Pfade anzupassen.

2.2.2 Testsuiten

In der professionellen Softwareentwicklung sind Implementierung von Tests und Testdaten, also kleinen Datensätzen, die eine Überprüfung der Funktionalität ermöglichen, üblich. Testsuiten ermöglichen eine automatisierte Evaluierung des Codes und unterstützen nicht nur stark in der Entwicklung, sondern dienen auch zur abschließenden Überprüfung. Diesen Testsuiten kommt insbesondere dann eine wichtige Bedeutung zu, wenn die originalen Datensätze zu groß sind oder rechtlichen Beschränkungen unterliegen. Außerdem könnten sie Respositorien automatisch überprüfbare Kriterien für die Übernahme an die Hand geben. Hierzu bräuchte es dann entsprechende Vorgaben durch das Repositorium.

2.2.3 Test auf verarbeitbare Datenstruktur

Wenn Skripte im Sinne der Nachnutzung auf neue Datensätze angewendet werden sollen, müssen Skripte überprüfen, dass Daten auch die vorausgesetzten Strukturen und Eigenschaften besitzen. Mindestens sollte eine Überprüfung in den Funktionen erfolgen und mit sprechenden Fehlermeldungen versehen werden.

2.3 Modularisierung

Für eine Wiederverwendbarkeit ist es besonders günstig, den Workflow möglichst weit zu modularisieren und die Schnittstellen umfassend zu beschreiben. Übliche Schritte eines Workflows sind das Laden von Daten, Vorverarbeitung, Anreicherung, Analysen und Visualisierung. Ein erster Schritt wäre hier, die einzelnen Bestandteile des Workflows zu identifizieren und modular anzulegen. Wenn nun bei der Nachnutzung andere Formate eingelesen werden sollen, kann ein neues Modul die gewünschten Funktionen hinzufügen, während die weiteren Module weiterverwendet werden können. Dies gilt in gleicher Weise, wenn in den übrigen Schritten andere Methoden angewendet werden sollen.

2.4 Dokumentation

Die Dokumentation sollte die Arbeitsumgebung und Nutzung des Skripts vollständig beschreiben. Hierbei sollte möglichst die konkrete Benutzung der einzelnen Skripte auch beispielhaft aufgezeigt werden. Hierfür eignet sich am besten das Testset. Besondere Wichtigkeit kommt den Interaktionen mit anderen Werkzeugen zu. Hier gehört dann die Konfiguration der Komponenten in die Dokumentation. Auch im Skript selbst sollte Dokumentation durch sprechende Fehlermeldungen und Nachrichten, sprechende Benennungen von Funktionen, Variablen und schließlich Kommentare im Quellcode gegeben sein.

2.5 Lizenz

In den FAIR Prinzpien werden Lizenzen in R1.1 adressiert: “(meta)data are released with a clear and accessible data usage license.” Die Frage, welche konkrete Lizenz man für sein Skript, aber auch Daten verwenden sollte, ist nicht leicht, insbesondere wenn die eigenen Arbeiten auf dem Werk Dritter aufbauen. Einen guten Einstig in die Thematik bietet [5]. Der dort beschriebene Public License Selector ist in verschiedenen Diensten integriert worden und lässt sich auch hier finden: `http://ufal.github.io/public-license-selector/`. Licentia [6] ist ein weiteres Werkzeug, das Lizenzen vergleicht und die Übersetzung von Lizenzen in RDF unterstützt und findet sich hier: `http://licentia.inria.fr/`

3 Diskussion und Ausblick

Viele der in Abschnitt 2 genannten Aspekte sind letztlich generelle Hinweise zur Qualitätssicherung von Skripten und Code im Allgemeinen und werden als solche zum Teil auch durch die Leitlinien des *clean codings*, best practices zur Erstellung guten Codes, in der Softwareentwicklung abgedeckt [7]. Einige Gesichtspunkte sind ohne weiterführende Programmierkenntnisse mit weniger (z. B. sprechende Variablen-/Funktionsbenennungen, Kommentare im Skript) oder mehr (z. B. Erstellung einer umfassenden und durch unterschiedliche Nutzergruppen anwendbaren Dokumentation) Aufwand umsetzbar.[3] Andere erfordern fortgeschritteneres (und für die eigentliche Aufgabe nicht unmittelbar und zwangsweise notwendiges) Wissen (z. B. die Implementierung von Datenstrukturtests). Hinsichtlich einer Kontrolle der Arbeitsumgebung bietet sich für die Bereitstellung von lokalen Abhängigkeiten, also dem Interpreter, system- und sprachspezifischen Bibliotheken und weiteren lokalen Softwarekomponenten, grundsätzlich eine Virtualisierung in virtuellen Maschinen und Containerlösungen an. Hier entsteht allerdings Mehraufwand, der wiederum zusätzliche Kenntnisse sowie mehr Zeit bei der Implementierung und Testung benötigt und schlussendlich mehr Speicherplatz belegt, da nicht nur das Skript und die Dokumentation gespeichert werden müssen, sondern der ganze Container beziehungsweise die ganze virtuelle Maschine. Für verschiedene Aufgaben bietet sich diese Herangehensweise an und es gibt einige Projekte, die hierzu Infrastruktur anbieten. Die Langfristigkeit solcher Lösungen ist hierbei auch vom Bestand der Formate oder gegebenenfalls Emulatoren abhängig.

Von Seiten der Forschungsinfrastrukturen stellt sich für die Übernahme von Skripten in Repositorien die Frage nach geeigneten Metadatenschemata und danach, wie man den Nutzen aufwandsarm validieren kann. Für Ersteres könnten bestehende Standards ein guter Ausgangspunkt sein, für Zweiteres könnten Testsuiten einen ersten Schritt darstellen.

Eine FAIR(ere) Aufbereitung von Skripten setzt – wie die vorangegangene Diskussion zeigt – über die Untersuchung der eigentlichen Forschungsfrage hinausgehenden Aufwand und Kompetenzen voraus (und dies von Fachwissenschaftlerinnen und Fachwissenschaftlern, die in der Regel keine ausgebildeten Programmierer, sondern häufig Autodidakten sind, die bedarfsgeleitet Programmiersprachen einsetzen). Dennoch sind Skripte – verstanden als Implementierung analytischer Workflows – als "non-data assets" nach Wilkinson et al. eine zentrale Komponente des wissenschaftlichen Prozesses und deshalb lohnt es sich, den Aufwand zu investieren, diese so FAIR wie möglich zu gestalten: "Analytical workflows, for example, are a critical component of the scholarly ecosystem, and their formal publication is necessary to achieve both transparency and scientific reproducibility." [4, S. 5] Über die Nachnutzbarkeit hinaus nutzt eine solche Vorgehensweise letztlich auch dazu, den eigenen Forschungsprozess besser zu handhaben.

[3] Für viele verschiedene Programmiersprachen existieren Pakete, die Schnittstellen zu *pandoc* [8] bieten und somit eine Dokumentationserstellung direkt aus dem Code in unterschiedliche Ausgabeformate ermöglichen.

4 Danksagung

Wir bedanken uns bei zwei anonymen Gutachtern und Bernhard Fisseni und Thorsten Trippel für hilfreiche Kommentare und Diskussionen.

Literaturverzeichnis

[1] Nosek, B. A., G. Alter, G. C. Banks, D. Borsboom, S. D. Bowman, S. J. Breckler, S. Buck et al. Promoting an open research culture. *Science*, 348(6242):1422–1425, 2015.

[2] Deutsche Forschungsgemeinschaft "Leitlinien zur Sicherung guter wissenschaftlicher Praxis", 2019. `https://doi.org/10.5281/zenodo.3923602`

[3] Deutsche Forschungsgemeinschaft "Digitaler Wandel in den Wissenschaften", 2020. `https://doi.org/10.5281/zenodo.4191345`

[4] Wilkinson, M. D., M. Dumontier, I. J. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg et al. The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3(1):1–9, 2016.

[5] Kamocki, Pawel, P. Straňák, M. Sedlák. "The Public License Selector: Making open licensing easier." In *Proceedings of the Tenth International Conference of Language Resources and Evaluation (LREC 2016)*, edited by N. Calzolari, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard et al., 2533-2538. Paris: European Language Resources Association (ELRA), 2016.

[6] Cardellino, C., S. Villata, F. Gandon, G. Governatori, H. Lam, A. Rotolo. "Licentia: a Tool for Supporting Users in Data Licensing on the Web of Data." In *Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference, ISWC 2014, Riva del Garda, Italy, October 21, 2014)*, edited by M. Horridge, M. Rospocher, J. van Ossenbruggen, 277-280, CEUR-WS.org, 2014. http://ceur-ws.org/Vol-1272

[7] Martin, R. C. *Clean Code. A Handbook of Agile Software Craftsmanship*. Upper Saddle Rive, NJ et al.: Prentice Hall, 2009.

[8] McFarlane, J. "pandoc: A universal document converter [Software]" (2016)