https://doi.org/10.21248/ka-notizen.85.13

Computercode in seinen Dimensionen ethnografisch begegnen

Libuše Hannah Vepřek, Sarah Thanner, Lina Franken und das Code Ethnography Collective (CECO)

Zusammenfassung

Alltage sind zunehmend durch programmierte Algorithmen und Computercode strukturiert, während Alltage und Menschen diese wiederum gestalten. Doch wo genau lässt sich Computercode ethnografisch in seinen Wirkungsweisen begegnen? Er ist niemals eines, sondern immer vieles. In diesem Beitrag beleuchten wir unterschiedliche methodische Herangehensweisen an Computercode in seinen verschiedenen Dimensionen und veranschaulichen diese anhand von Fallbeispielen. Dabei unterscheiden wir die folgenden fünf Dimensionen: a) Code als Text und seine Einschreibungen b) Computercode in seiner Performanz und Ausführung, c) Programmieren als Praxis: Entwickeln, Basteln, Debuggen und Hacken, d) Infrastrukturieren mit und durch Computercode, und e) Gouvernance und Gouvernementalität: Regieren mit und durch Computercode. Die Perspektivierungen sind nicht exklusiv und überschneidungsfrei, vielmehr erlaubt das Zusammendenken verschiedener Dimensionen von Code als soziotechnische Assemblage ein tiefergehendes Verständnis. Wir diskutieren, welche Dimensionen bei welchem Forschungsinteresse hilfreich sein kann und betonen gleichermaßen die Notwendigkeit einer Reflexion jener perspektivischen Entscheidungen, die wir im Forschungsprozess treffen.

Schlagwörter: Computercode, Algorithmen, Ethnografie, Methoden, Assemblage

Libuše Hannah Vepřek, Institut für Empirische Kulturwissenschaft und Europäische Ethnologie, Ludwig-Maximilians-Universität München, Deutschland

Sarah Thanner, Institut für Information und Medien, Sprache und Kultur, Universität Regensburg, Deutschland

Prof. Dr. Lina Franken, Digital Humanities in den Kulturwissenschaften, Universität Vechta, Deutschland

Computercode und wie wir ihm in seinen verschiedenen Dimensionen ethnografisch begegnen

Alltage sind in zunehmendem Maße durch programmierte Algorithmen geprägt, also durch Computercode strukturiert. Oftmals unbewusst verlassen wir uns etwa auf digitale Verkehrsleitsysteme, die urbane Bewegungsströme lenken, sitzen in der durch programmierte Systeme gestützten Bahn oder konsumieren Nahrungsmittel von per Satellit überwachten Feldern. Auch für offensichtliche Bereiche wie etwa die Kommunikation über digitale Kanäle (sei es das Telefon oder Social Media) machen sich viele Menschen selten bewusst, dass programmierte Prozesslinien und damit die Wirkweisen von ausgeführtem Computercode Einfluss darauf nehmen, was möglich ist, was nicht und wie sich eine Interaktion, eine Praxis oder Situation entfaltet.

Die zunehmende Verwobenheit von Alltagswelten und Codestrukturen wirft die Frage nach der Rolle von Computercode als Gegenstand ethnografisch-kulturwissenschaftlicher Forschung auf. Weshalb sich Kulturwissenschaftler:innen mit Computercode beschäftigen sollten, wird anhand von drei Beispielen deutlich: Wenn sich etwa ein:e Ethnograf:in für alltägliche Nutzungspraktiken von Video-Streaming-Diensten, wie zum Beispiel YouTube oder Netflix, interessiert, dann ist es wichtig, die Rolle der Algorithmen mitzudenken: Diese nehmen auf Basis des Verhaltens von User:innen Einfluss darauf, welche Videos einzelnen User:innen als Vorschläge angezeigt werden (siehe Hagemeister in diesem Band). Oder ein Forschungsprojekt könnte die Arbeitsbedingungen von Lieferantendiensten wie Lieferando, Getir oder auch Amazon analysieren wollen. Dabei sind die Zusammenhänge kaum verständlich, ohne die die Arbeitsalltage bestimmenden Algorithmen zu berücksichtigen, welche etwa die nächste Fahrtroute berechnen oder Schichtpläne erstellen. Oder wir stellen uns vor, dass ein:e Ethnograf:in sich für Prozesse der Grenzziehung und Grenzregime interessiert. Dann ist es nicht unwahrscheinlich, dass der Einsatz von Software zur Gesichtserkennung an Flughäfen oder digitale Grenzüberwachungssysteme in den Blick geraten. Diese Beispiele zeigen, wozu es wichtig und interessant sein kann, sich als Ethnograf:in mit Computercode und seinen Wirkweisen auseinanderzusetzen und machen auch die gesellschaftliche und politische Relevanz eines solchen Unterfangens deutlich.

In der gegenwärtigen ethnografischen Forschungslandschaft wird Computercode bisher jedoch selten explizit betrachtet. Dies ist auch nicht immer einfach oder möglich – haben wir es doch oft mit bewusst nach außen abgeschlossenen Systemen zu tun, die etwa aus der Privatwirtschaft kommen. Obgleich uns dies vor zahlreiche (Zugangs-)Schwierigkeiten stellt, verweist es auch auf die Gemachtheit von Computercode durch Menschen. Alltage prägen und strukturieren auch Computercode. Egal wie automatisiert Systeme wirken, sie sind immer durch menschliche Akteur:innen gestaltet und werden dies in vielen Fällen fortlaufend. Sichtbar wird dies besonders dann, wenn automatisierte Systeme fehlschlagen oder nicht funktionieren und menschliche Interventionen notwendig sind.

In Anlehnung an Pierre Bourdieus Habitus-Konzept (1982) lässt sich also folgern: Code wird durch Alltag strukturiert und wirkt gleichermaßen strukturierend auf Alltagswelten. Außerdem inter- beziehungsweise intra-agiert (Barad 1996) Code nicht nur mit Menschen, sondern auch mit anderen programmierten Systemen, Geräten, Daten und Infrastrukturen.

Weitet man den Blick über disziplinäre Grenzen auf die Sozial- und Geisteswissenschaften hinaus, so lassen sich etwa ab der Jahrtausendwende durchaus verschiedene Strömungen, Bereiche oder gar Subdisziplinen ausmachen, die sich der Analyse von Computercode

widmen. Dazu zählen zum Beispiel die Critical Code Studies, die Critical Data Studies, Software Studies, sowie die Digital STS. Ansätze, Computercode in ethnografische Forschungen einzubinden, orientieren sich oft an diesen, folgen dabei aber meist keinem etablierten Vorgehen. Gerade die methodischen Fragen sind noch wenig ausgearbeitet und müssen adaptiert werden. Letzteres gilt besonders dann, wenn die Vorgehensweisen nicht direkt in empirisch kulturwissenschaftliche Fragestellungen, beispielsweise nach Bedeutungen und Praxen der Sinnstiftung, zu übersetzen sind.

Weil Alltagswelten immer auf bestimmte Weisen das Digitale strukturieren und gleichzeitig digital strukturiert und mediiert sind – auch wenn nicht unmittelbar, sondern etwa als Folge globaler Finanzmärkte oder Lieferketten – möchte dieser Beitrag unterschiedliche methodische Herangehensweisen an Computercode und damit auch immer Begegnungen mit Computercode in seinen verschiedenen Dimensionen aufzeigen. Diese können je nach Forschungsfeld und vor allem Forschungsinteresse unterschiedlich und gegebenenfalls in Kombination miteinander geeignet sein. Sie verweisen auf das umfassende Forschungsfeld von Computercode, das in einem Querschnitt erst in seiner Komplexität sichtbar wird. Wie die verschiedenen Schalen einer Zwiebel greifen auch die Dimensionen ineinander und sind miteinander verbunden (vgl. Abbildung 1). Wo die Grenzen dieser Zwiebel auszumachen sind – ob etwa die oberirdischen Triebe und Blüten oder die Wurzeln, die das Gewächs in ein Ökosystem einbinden, für unsere Forschungen eine Rolle spielen – ist eine bewusste Entscheidung. Natürlich ist dieses Bild ambivalent. Doch lässt es sich spielerisch als visuelle Metapher einsetzen, um zu verdeutlichen, dass wir es als Forschende sind, welche die Grenzen der Zwiebel festlegen und auch, ob wir auf eine bestimmte Schicht oder eben Dimension fokussieren möchten – die Schnitttechnik ist also entscheidend. So kann die Metapher der

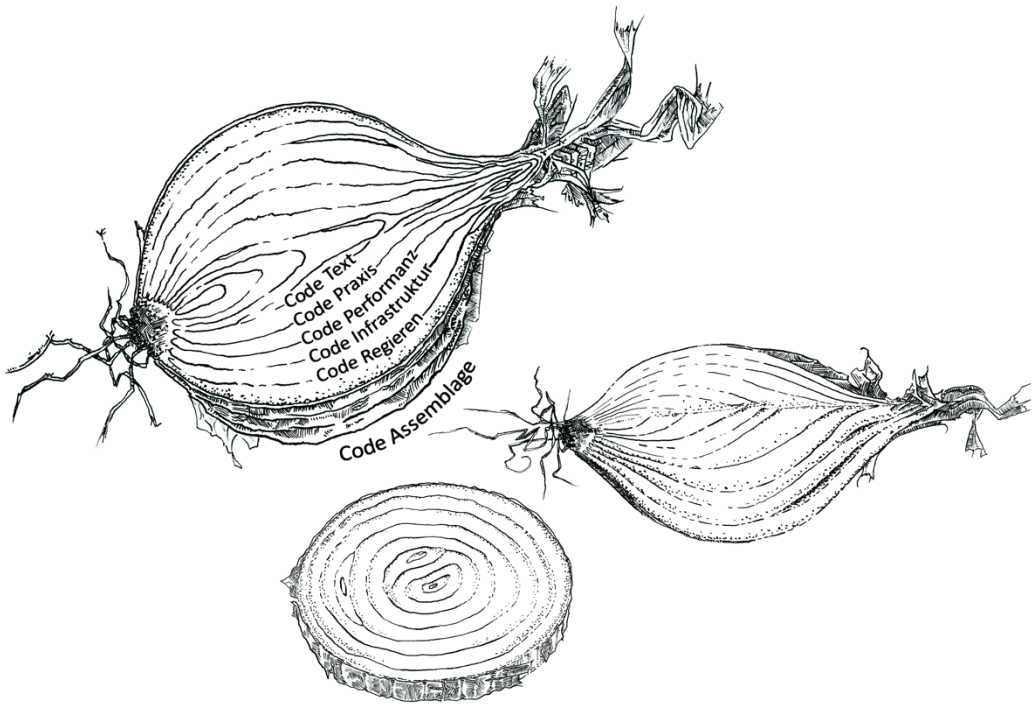

Abbildung 1: Code in seinen Dimensionen.
Grafik: Emil Rieger.

Zwiebel, ihre Schichtungen und Unabgeschlossenheit ebenso wie die im Folgenden aufgefächerten Dimensionen zum Weiterdenken anregen – auch wenn sie nie vollumfassend passend sind.

Mit der Darstellung von Begegnungen mit Computercode in seinen unterschiedlichen Dimensionen möchten wir forschungspraktische Anregungen und Hilfestellungen dazu geben, wie Computercode konstruktiv in Forschungsprojekte einbezogen werden kann. Wir möchten deutlich machen, warum und wozu es sinnvoll und in etlichen Forschungsfeldern auch notwendig ist, sich auch und gerade in ethnografischer Forschung mit Computercode zu beschäftigen.

Die verschiedenen Verfahren weisen unterschiedliche methodische Komplexitäten auf und einigen mag das eine, anderen das andere Vorgehen leichter fallen. Wir möchten Leser:innen motivieren, sich mit den einzelnen Vorschlägen auseinanderzusetzen und darüber nachzudenken, inwiefern Computercode in der eigenen Forschung relevant ist oder werden sollte. Da Code uns in unseren eigenen Alltagen ebenso wie in unserer Forschung in den unterschiedlichsten Kontexten begegnet, möchten wir Ethnograf:innen dazu einladen, die mitunter vorhandene Scheu kritisch zu reflektieren. Für eine ethnografische Beschäftigung mit Computercode ist es nicht immer notwendig, selbst eine Programmiersprache zu beherrschen. Nach außen abgeschlossene Systeme, die durch Computercode (mit)bedingt sind, erfordern aber zumindest eine gewisse digitale Literalität, also ein Verständnis für Prozesse digitaler Informationsverarbeitung und deren zugrunde liegende Logiken. Je nach Forschungskontext kann es grundlegend sein, mit den im Feld verwendeten Programmiersprachen vertraut zu sein. Gerade als Ethnograf:innen sind wir bestens darauf vorbereitet, uns ein solches Verständnis im Forschungsprozess reflexiv und multiperspektivisch zu erarbeiten und Code in seinen sozio-materiellen Zusammenhängen und Bedeutungen zu betrachten. Die Einbindung von Computercode in ethnografische Forschungen bedeutet also mehr als die Berücksichtigung von technischen Komplexitäten und dient vielmehr der Untersuchung von Code als welt-erzeugender Bestandteil unseres Alltags. Unser Text erhebt dabei nicht den Anspruch, umfassend zu sein oder eine konkrete Anleitung zu geben, sondern soll vielmehr Hinweise auf bestehende Ansätze im Überblick aufzeigen und zur konkreten Ausgestaltung der eigenen Methodenkombination anregen.

Infobox 1: Code Ethnography Collective (CECO). Dieser Text geht aus der Zusammenarbeit verschiedener Ethnograf:innen im CECO hervor, welches aus der Feststellung entstanden ist, dass Computercode selten in ethnografische Arbeiten einbezogen wird und es demnach auch an Beispielen mangelt, wie ein solcher Einbezug methodisch umgesetzt werden kann. Seit 2020 kommt die aus circa zehn festen Mitgliedern bestehende Gruppe in regelmäßigen Abständen zusammen, um Zugänge und Relevanzen von Computercode in Ethnografien zu diskutieren. Aus der Gruppe ist bereits eine eher theoretisch-konzeptionelle Darstellung der Frage, wie Code ethnografisch beforscht werden kann, hervorgegangen (Carlson et al. 2021). Die Studien der Autorinnen des vorliegenden Aufsatzes, die Teil dieser Gruppe sind und sich auf unterschiedliche Weise mit Computercode beschäftigen, dienen im Folgenden als exemplarische Engführungen in Form von Hands-on-Beispielen für methodische Umsetzungen. Die Gruppe ist als kollaboratives, digitales Format auf Dauer ausgelegt und für neue Mitglieder offen. Weitere Informationen sind unter https://codeethnographycollective-ceco.github.io/ zu finden.

Wenn wir über Computercode nachdenken, haben wir oft bestimmte Repräsentationen vor Augen: Etwa lange Reihungen von Textzeilen, die für jene, die nicht in der jeweiligen Programmiersprache ausgebildet sind, nur schwer in ihrer Bedeutung zu entschlüsseln sind. Oder Anhäufungen von Nullen und Einsen, die keine Ähnlichkeit mehr mit menschlicher Sprache aufweisen. Doch ist Computercode niemals nur eines, sondern immer vieles, oder, mit Annemarie Mol (2002) gesprochen, multipel. Nähern wir uns einer Repräsentation, entschwindet uns das Fassbare in zahlreiche Elemente und Akteur:innen (Programmierer:innen, Nutzer:innen, Code als Text, binäre Operationen, logische Gatter, Server, Protokolle, etc.) und Ebenen (die des Alltags, der Zukunft oder der Vergangenheit etwa). Gleichzeitig haben wir es aber auch nicht mit einer Summe von Elementen zu tun, die ein Ganzes bilden, sondern mit einem situativ hervorgebrachten Phänomen. Ein Verständnis in Anlehnung an Assemblage als theoretisches Konzept (vgl. Deleuze & Guattari 1987; Collier & Ong 2005) ist erkenntnisbringend, wie wir an anderer Stelle formuliert haben: „[C]ode is an assemblage of multiple layers in polymorph existence; of parallel aspects that are mutually dependent and interlinked." (Carlson et al. 2021: 13)

Verstehen wir Computercode kulturwissenschaftlich, so spannt sich ein weites Feld möglicher und relevanter Forschungsfragen und Perspektivierungen auf: Interessieren wir uns für Normvorstellungen, gesellschaftliche oder informatische Ordnungen, Vorstellungen von zukünftigen Nutzer:innen oder Wirkungsweisen von digitalen Technologien, die in Computercode eingeschrieben sind? Sind es die dahinterliegenden Infrastrukturen und Materialitäten, wie zum Beispiel die Hardware, die Code ausführbar machen und auf unseren Alltag wirken? Oder geht es uns vorrangig um die Arbeit am Code als Alltagspraxis von Programmierer:innen? Dies sind nur einige Beispiele möglicher Forschungsperspektiven.

Infobox 2: Computercode und Algorithmen. Für eine erste Annäherung an ethnografische Begegnungen mit Computercode und Algorithmen kann es hilfreich sein, sich zunächst mit einer eher allgemeinen und funktionalen Definition vertraut zu machen. Demnach beschreiben Algorithmen Vorgehen und Anleitungen für die Lösung eines konkreten Problems. Es handelt sich dabei um strukturierte und systematische Anweisungen oder Regeln in Form von Schritten, die aufeinander aufbauen. Algorithmen können beispielsweise als verbale Ideen (ähnlich wie etwa Kochrezepte) oder Diagramme ausgedrückt oder mit Programmiersprachen implementiert werden. Als solche können sie dann in Computerprogrammen ausgeführt werden und sind Teil von Computercode.
Computercode dagegen umfasst grundsätzlich alle Schritte und Anweisungen, die von einer Maschine ausgeführt werden können. Er ist in einer spezifischen Programmiersprache (vgl. unten) verfasst, die oft über mehrere Zwischenschritte (mit sogenannten Compilern, oder direkt über Interpreter) in Binärcode (synonym zu Maschinencode) übersetzt wird. In diesem Binärcode werden die Informationen aus den Programmen lediglich als 0 und 1 dargestellt, die technisch, etwa durch elektrische Spannungen, umgesetzt werden können. Computercode kann daher mehrere Algorithmen beinhalten, die oft ineinander verschachtelt sind.
Wie wir in diesem Beitrag anhand der unterschiedlichen Dimensionen aufzeigen werden, ist das empirisch-kulturwissenschaftliche Verständnis von Computercode und Algorithmen jedoch weiter gefasst und nicht auf diese Definition beschränkt.

Deshalb besteht die Schwierigkeit, einen sinnvollen Ausgangspunkt zu finden, die abhängig von der jeweiligen Forschungsfrage unterschiedlich gelöst werden kann. Wir möchten in diesem Beitrag daher auch nicht ‚die eine' Methode vorstellen, mit der Computercode in ethnografische Forschung eingebunden werden kann – schließlich zeichnet sich ethnografisches Forschen ohnehin immer durch ein vielfältiges Bündel an Methoden aus, das jeweils unterschiedlich zum Einsatz kommt. Stattdessen zeigen wir im Folgenden mit Blick auf verschiedene Dimensionen von Computercode mögliche methodische Herangehensweisen auf. Diese Dimensionen umfassen Code als Text und seine Einschreibungen, Computercode in seiner Performanz und Ausführung, Programmieren als Praxis: Entwickeln, Basteln, Debuggen und Hacken, Infrastrukturieren mit und durch Computercode sowie Governance und Gouvernementalität: Regieren mit und durch Computercode.

In den einzelnen Abschnitten stellen wir zunächst das Verständnis von Computercode aus der jeweiligen Perspektive kurz vor und erläutern dann bestehende Ansätze zu dessen Analyse. Wir teilen das Phänomen Computercode im Folgenden bewusst in einige ausgewählte Dimensionen und damit einhergehende Perspektivierungen auf, um diese anschließend wieder zusammenzubringen (vgl. die unterschiedlichen Zwiebelformationen in Abbildung 1). Ergänzend dazu bieten wir jeweils ethnografische Fallstudien an, die mögliche methodische Vorgehensweisen konkret aufzeigen. Die einzelnen Beispiele wurden von uns bewusst in die Nähe bestimmter Dimensionen gerückt, da sie einen gewissen Bezug zu den jeweils beschriebenen Perspektiven auf Computercode aufweisen. Und doch gehen sie meist darüber hinaus, lassen sich nicht klar zuordnen und haben stets auch Bezüge zu anderen Perspektiven und Dimensionen von Computercode. Bei der nachfolgenden Strukturierung handelt es sich somit um eine von uns für diesen Beitrag vorgenommene Setzung, die durchaus in Frage gestellt werden kann (und soll), die auch anders hätte vorgenommen werden können und die nicht als abschließend zu verstehen ist. Die Frage, in welcher Kombination und Schwerpunktsetzung die unterschiedlichen Dimensionen von und Perspektiven auf Computercode in welchem methodischen Zuschnitt schließlich in die eigene Forschung einfließen, ist daher immer eine Forschungsentscheidung, die mit Blick auf das eigene Erkenntnisinteresse und die verfolgte Fragestellung entschieden werden muss.

Code als Text und seine Einschreibungen

Computercode wird in dieser Dimension als Text verstanden, den wir lesen können und der Einschreibungen aufweist. Hier ist Computercode in einer höheren Programmiersprache repräsentiert, wie etwa Java, Python oder C++. Unter ‚höheren' Programmiersprachen werden solche verstanden, die ein höheres Abstraktionsniveau aufweisen als beispielsweise maschinenlesbarer Code, der aus 1 und 0 besteht. Oft ist Computercode in höheren Programmiersprachen in einer der menschlichen Sprache ähnlichen formalen Sprache geschrieben, was sie grundsätzlich für Menschen zugänglich macht. Computercode in höheren Programmiersprachen ist nicht direkt ausführbar, sondern muss von sogenannten Compilern oder Interpretern zunächst in Maschinensprache übersetzt werden.

Da Computercode (zumindest meistens) von Menschen geschrieben wird, kann er auch von anderen Menschen gelesen werden. Voraussetzung ist ein grundlegendes Verständnis der Programmiersprache selbst. Oft besonders bedeutsam für das menschliche Verständnis sind außerdem die im Code enthaltenen Kommentare und Erläuterungen, mit denen Programmierende ihren Code strukturieren und nachvollziehbar halten. Diese sind für die ma-

schinelle Les- und Ausführbarkeit nicht notwendig, jedoch gerade aus ethnografischer Perspektive interessant, da sie neben ihres erläuternden Charakters auch Einblicke in Arbeit und Verständnis von Programmierenden geben.

Seit den frühen 2000er Jahren hat sich das wissenschaftliche Feld der Critical Code Studies (CCS) herausgebildet. Hier wird mit einem engen Verständnis von Computercode gearbeitet, das sich zunächst auf den Code selbst als einen lesbaren und hermeneutisch interpretierbaren Text bezieht. Dieser wird – oft in einer Programmierumgebung – in hermeneutischen Zirkeln mehrfach gelesen und eine Interpretation aus dem Text heraus entwickelt. Nicht überraschend wurden die CCS maßgeblich von dem Literaturwissenschaftler Mark Marino (2020; 2018) geprägt: „As with all texts, the discovery and creation of meaning grow out of the act of reading and interpretation" (Marino 2020: 17). Das Verständnis geht also vom Text selbst aus. Sehr wohl wird aber nach den Spuren Ausschau gehalten, welche die Programmierenden im Code hinterlassen haben, wozu etwa getroffene Design-Entscheidungen im Aufbau der Codestruktur zählen. Code soll somit in seiner Situiertheit in Plattformen, der zusammenwirkenden Software und eben den Spuren seiner Entwicklung und Verbreitung analysiert werden (Marino 2018: 474). Marino entwickelt eine semiotische Lesart von Code, der eine „unique semiotic form of discourse" (ebd.: 18) darstellt. Zu beachten ist dabei, dass der verwendete Diskursbegriff an linguistische Diskursanalysen (und weniger etwa an die in der Empirischen Kulturwissenschaft verbreitete wissenssoziologische Diskursanalyse, vgl. Müske 2020) angelehnt ist. Dennoch wird auch hier der Kontext einbezogen, in welchem der Code entstanden ist und verwendet wird (Marino 2020: 23).

Marino zufolge wird Code somit als „kultureller Text" (Marino 2018: 472; Übersetzung der Autorinnen) verstanden, der Zugang zur Analyse der Interaktionen unterschiedlichster menschlicher und nicht-menschlicher Akteure (Menschen, Computer, unterschiedliche Systeme, Software, Hardware) eröffnet. Das Kritische der CCS schließlich, versteht Marino einerseits als die Kritik verbreiteter Narrative über Code (zum Beispiel ein rein funktionales Verständnis von Code) und andererseits als die Anknüpfung an kritische Ansätze des Konstruktivismus (Marino 2018: 474). Dieser Ansatz ist methodisch bereits sehr konkret ausgearbeitet. Inspirationen aus der literaturwissenschaftlichen Zuwendung zu Code, die sich womöglich für ohnehin stets kritische Perspektiven einbeziehende kulturwissenschaftliche Forschungsarbeiten ergänzend hinzuziehen oder anpassen lassen, finden sich auf der gleichnamigen Website der CCS.

Ein anschauliches Beispiel bildet die Analyse des einzeiligen BASIC-Programms „10 PRINT", das Nick Montfort et al. als kulturelles Artefakt aus unterschiedlichen Perspektiven untersuchen, um zu zeigen „how to read code deeply and show[] what benefits can come from such readings" ohne dabei den Kontext des Codes aus den Augen zu lassen (2013: 5).

Daneben zeigt das kollaborative Leseprojekt von Jessica Pressman, Mark Marino und Jeremy Douglass (2015) zu William Poundstones „Project for Tachistoscope (Botomless Pit)", wie verschiedene interpretative Methoden kombiniert werden können, um das digitale Literaturwerk zu untersuchen. Während gerade die Kombination verschiedener Methoden und Perspektiven zur Analyse digitaler Objekte erkenntnisbringend ist, stellte sich in diesem Projekt die Codeanalyse als wichtiger Zugangspunkt für das Gesamtprojekt heraus, der das literarische Werk zugänglicher werden ließ (Douglass et al. 2020: 5–6).

Ethnografisches Beispiel: Mensch-Technologie-Relationen in Human Computation

Im Rahmen ihrer Promotionsforschung untersuchte Libuše Hannah Vepřek, wie sich Mensch-Technologie-Relationen in Human-Computation-basierten Citizen-Science-Projekten entfalten und fortwährend verändern (2023; siehe Vepřek in diesem Band). In Human-Computation-Systemen lösen Menschen im Zusammenwirken mit Computern Probleme, die über aktuelle Künstliche-Intelligenz-Technologien hinausgehen. Im Citizen-Science-Kontext werden Teilnehmende eingeladen, beispielsweise an der Analyse von Forschungsdaten im Zusammenspiel mit Computeralgorithmen beizutragen. Vepřek untersuchte dabei zudem, welche Vorstellungen des ‚richtigen' Zusammenwirkens von Mensch und Maschine bestehen und wie Menschen in rechnerische Systeme eingebunden werden (sollen).

Einen wichtigen Zugangspunkt bildete der in den Programmiersprachen PhP und Python verfasste Computercode des Fallbeispiels *Stall Catchers* selbst, den sie mittels ihrer „fokussierten Code Analyse" (Carlson et al. 2021) als Text analysierte. Basierend auf einem grundlegenden Verständnis des Human-Computation-Systems und der Abläufe auf Benutzer:innenseite durch vorausgehende Teilnehmende Beobachtungen und Interviews mit Entwickler:innen und Teilnehmer:innen, konnte Vepřek für die Analyse interessante Codeabschnitte identifizieren, die Einstiegspunkte für den Erkenntnisgewinn zu Mensch-Technologie-Relationen im Fallbeispiel bildeten. Von diesen ausgehend analysierte sie den Code ähnlich dem qualitativen Kodieren in ethnografischer Forschung, indem sie den Handlungslinien im Code folgte, etwa durch das Nachspüren von I/O(Input/Output)-Operationen oder Abfolgen von Funktionsaufrufen. So wurde der Code, der über verschiedene Dateien und Module organisiert ist, nicht starr von oben nach unten gelesen beziehungsweise kodiert, sondern vielmehr entlang der Aktions- und Handlungslinien von situiertem Code in seiner Ausführung zusammen mit Datenbanken- und Nutzer:innen-Intra-aktionen. Für die spätere Reflexion des Vorgehens und die Analyse stellte sich dabei das Festhalten der einzelnen Schritte als wichtig heraus – beispielsweise in Form von Prozessskizzen. Während dieses Vorgehen eine gewisse digitale Literalität bedarf, ermöglicht der Einbezug von qualitativen Interviews mit Programmierer:innen einen weiteren textnahen Zugang. Hier nehmen Programmierer:innen vor den Augen der Ethnograf:in eine Durchsicht des Codes vor, erklären Zusammenhänge und können Fragen beantworten. Computercode wird dann über die vermittelnden Worte von Programmierer:innen zugänglich für uns. Dabei kann es vorkommen, dass diese den Code bereits vor einiger Zeit, vielleicht aber auch gar nicht selbst geschrieben, sondern von jemand anderem (und inzwischen sogar von einem Computermodell) übernommen haben. Dies in der Analyse mitzudenken ist wichtig, um die verschiedenen Praxen, Zusammenarbeiten und Zugänge zum Code verstehen zu können.

Um die Prozess- und Handlungslinien und die Entstehung von Human-Computation-Systemen darüber hinaus im Kontext von Citizen Science situieren zu können, ging Vepřeks ethnografische Forschung über die fokussierte Analyse des Computercodes hinaus.

Über zwei Jahre arbeitete sie dabei mit dem *Human Computation Institute*, das *Stall Catchers* entwickelt hat, zusammen, während sie es gleichzeitig ethnografisch beforschte. Als Teil der Mitarbeit wirkte Vepřek beispielsweise an der Optimierung des Computercodes von *Stall Catchers* mit. Für die kulturwissenschaftliche Fragestellung war dabei nicht nur die direkte Auseinandersetzung mit dem Code relevant, sondern ebenso das Teilnehmen an regulären Besprechungen, schriftlichen Konversationen oder kurzfristig einberufenen Krisensitzungen. So wurden einerseits die Abläufe, Routinen und Vorgehensweisen beim Entwickeln von

Computercode erfahrbar. Andererseits konnten so auch die Umgangsweisen mit Fehlern und Leistungsproblemen oder Infrastruktur-Ausfällen beobachtet und Lösungsansätze in Form von Debugging oder dem Ausbauen der Infrastruktur nachvollzogen werden.

Diese Perspektive ermöglichte nicht nur Einblicke in die Praxen, sondern ebenso in die Wertvorstellungen, Normen und Vorstellungen von Mensch-Technologie-Relationen, die in das *Stall-Catchers*-Projekt eingeschrieben beziehungsweise einprogrammiert werden und die Implementation anleiten.

Teilnehmende Beobachtungen und qualitative Interviews dienten über die Identifizierung interessanter Ansatzpunkte im Code hinaus auch dem besseren Verständnis desselben und des Erkenntnisgewinns. Gerade durch die Verknüpfung dieser unterschiedlichen methodischen Vorgehensweisen wurden beispielsweise neue User-Technologie-Relationen erkennbar, die aus dem spielerischen und experimentellen Handeln einiger Teilnehmer:innen in *Stall Catchers* hervorgingen. Anstatt den vorgegebenen Interaktionslinien von User:innen und Software zu folgen, entdeckten Teilnehmer:innen mittels eines Tricks neue Spielweisen, die so nicht per Design intendiert waren (Thanner & Vepřek 2023). Indem Teilnehmende die richtigen Momente ergriffen (Mousavi Baygi et al. 2021), konnten sie indirekt mit dem Computercode intra-agieren und neue Optionen eröffnen. Diese Praxen können als Gegenhandeln oder Widerständigkeiten interpretiert werden. Diese Beobachtungen wurden erst durch die Methodentriangulation aus Interviews, Teilnehmender Beobachtung und Codeanalyse möglich. In diesem Fall konnte der Trick weder aus der Software, noch aus den Logs (Protokolldateien) herausgelesen werden. Vielmehr konnte er erst im Austausch mit den Teilnehmer:innen nachvollzogen werden.

Computercode in seiner Performanz und Ausführung

Eine zweite Dimension stellt die Fokussierung auf Code in Ausführung und seine Performativität dar. Code wird dann nicht als Objekt oder Text verstanden, sondern in seinem Werden, im situativen Entfalten betrachtet. Damit erweitert sich die Perspektive der Forschenden deutlich. Nicht allein der Code selbst mit seinen vielfältigen soziokulturellen Bedeutungen bildet den Fokus des Forschungsinteresses, sondern dessen Verwobenheit mit alltäglichen Handlungen. Denn Code – so diese Perspektive – ist nur in seiner Situiertheit verständlich, in seinem Handlungsvollzug und der Verwobenheit mit menschlichen und nichtmenschlichen Aktionen. Damit werden also neben dem Code sowohl die handelnden Akteur:innen, also auch die Geräte, Software und Infrastrukturen bedeutsam, welche Computercode erst ausführbar machen und gleichzeitig Einfluss darauf haben, ob und wie er ausgeführt wird beziehungsweise werden kann. Die Handlungsmacht ist dabei verteilt, wenn auch nicht gleichermaßen menschlichen und nicht-menschlichen Akteur:innen zugeschrieben. Aus dieser Perspektive werden besonders auch Vermittlungs- und Translationsprozesse in den Blick genommen.

Wenn Computercode in seiner Performanz betrachtet wird, ist seine Historizität nicht immer eine naheliegende Perspektive, weshalb wir sie an dieser Stelle explizit hervorheben möchten. Mit Historizität beziehen wir uns dabei weniger auf große historische Epochen, als vielmehr auf die vielen, sich innerhalb kürzester Zeiträume ereignenden Veränderungen. Denn Computercode entsteht in der Regel nicht in einem klar abgrenzbaren Zeitraum und bleibt dann in seiner aktuellen Form bestehen. Stattdessen wird Code prozessual entwickelt und in seinen Funktionalitäten oder Modulen Schritt für Schritt und oft von verschiedenen

Personen programmiert. Ist Computercode als Produkt zu einem gegebenen Zeitpunkt fertig, bedarf er zudem einer weiteren kontinuierlichen Instandhaltung und Wartung. Beispielsweise müssen Fehler behoben und die verwendete Programmiersprache auf dem neuesten Stand gehalten werden. Auch die importierten Bibliotheken, auf denen Code oft, wie oben erwähnt, unter anderem beruht, entstehen über gewisse Zeiträume hinweg und können sich ebenso kontinuierlich verändern. Damit findet also auf verschiedenen Ebenen inner- und außerhalb des eigentlichen Codes Wandel statt, der sich in vielfältigen, miteinander verstrickten und teils nicht linear laufenden einzelnen kleinen Änderungen zeigt. Unter der Historizität von Code verstehen wir Code in seinem Entstehen über verschiedene Versionen und Zeiträume hinweg, die über die *commit history*, also einzelner hinzugefügter Änderungen am Code, nachverfolgt werden können.

Um Computercode zu verwalten, gemeinsam daran zu arbeiten oder ihn zu verbreiten, wird Versionsverwaltungssoftware, wie etwa Git – oft in Formen der Webanwendungen GitLab oder GitHub – verwendet. Code kann über diese Plattformen hochgeladen (*push*) und heruntergeladen (*pull*) werden, um jeweils mit der gewünschten Version zu arbeiten. GitLab und GitHub bieten dabei – neben vielen anderen Möglichkeiten – einen interessanten Zugang, um Code unter anderem in seiner Historizität zu untersuchen. Hierfür ist es allerdings notwendig, dass wir als Ethnograf:innen Zugang zu den sogenannten Repositories und einzelnen Projekten darin erlangen, die über Versionierungssoftware verwaltet werden. Dabei sind Politiken zu beachten, denen folgend Veränderungen und einzelne Beiträge nicht öffentlich nachvollziehbar sein sollten und dazu führen, dass nicht alle Änderungen sichtbar gemacht werden.

Schließlich weist Wendy Chun darauf hin, dass Programmiersprachen und Computercode niemals eine kausale performative Äußerung beziehungsweise Quelle einer Handlung sind, sondern ihre Wirksamkeit erst im Nachhinein in den sozialen und maschinellen Relationen und Netzwerken erhalten. Ein kausales Verständnis von Software als eine Quelle für Handlung ist für Chun ein Fetischismus, der die Unbeständigkeiten und Kontingenzen der Ausführung und Relationen zwischen Code, den Handlungen von User:innen und dem Interface verschleiert (Chun 2008: 300).

Anders als bei der Zuwendung zu Computercode als Text, die sich mit der Herausbildung der CCS als eigene Forschungsrichtung abgrenzen lässt, lässt sich bei der Betrachtung der Performanz von Computercode in Ausführung keine eigene Forschungsrichtung anführen, sondern mehr ein loses Bündel an ethnografischen Studien, denen häufig ein Bezug zur Praxistheorie gemeinsam ist. (Computer-)Algorithmen sind in dieser Dimension „ontogenetic in nature (always in a state of becoming)“ (Kitchin 2017: 5), das heißt: Sie entfalten und verändern sich in einem Prozess, an dem viele beteiligt sind und der nie abgeschlossen ist (vgl. auch Gillespie 2014; Dourish 2016).

Diese Perspektive nimmt daher besonders das Wirken von Computercode in seiner Einbettung in bestimmte soziokulturelle Kontexte und dessen Effekte in den Blick. Dem Code kommt damit durchaus eine eigene Handlungsmacht zu, auch wenn er nur in spezifischen Kontexten und technischen Konstellationen aktiv sein kann. Gleichzeitig ist er menschengemacht. Dabei ist Code nicht ohne vorhergehende Codestrukturen zu realisieren: Sie bauen aufeinander auf, indem etwa Module und Bibliotheken wiederverwendet werden. Handlungsmacht ist hier also verteilt, aber nicht symmetrisch. Dabei treten für Kitchin einzelne Algorithmen eher in den Hintergrund und das algorithmische System in den Fokus (Kitchin 2017: 7), also die miteinander vernetzten vielfältigen Algorithmen, Code-Module und ihre Verbindungen, sogenannter ‚Glue Code', aus dem Software in der Regel besteht. Denn die

Analyse eines einzelnen Algorithmus kann immer nur eine Momentaufnahme widerspiegeln, bei der das Ontogenetische, die kontextuelle Vielfältigkeit und Veränderbarkeit von Algorithmen übersehen wird. So kann etwa ein und derselbe Sortieralgorithmus zum Sortieren von Kund:innendaten, für die Priorisierung interner Vorgänge in einem Unternehmen, als auch in Computerspielen eingesetzt werden (Bucher 2012: 1167).

Gerade eine ethnografische Alltagsperspektive kann auf die Performanz von ausgeführten Algorithmen und auf die mitunter großen Unterschiede verweisen, die zwischen dem bestehen, was theoretisch und per Design definiert oder sogar in Code programmiert wurde und dem, was in der Praxis tatsächlich passiert, etwa im Falle von Fehlern im Code oder von Umnutzungen und Weiterentwicklungen. So lässt sich dem relationalen und stets fragilen Entfalten im Zusammenwirken mit verschiedenen menschlichen Akteur:innen und nichtmenschlichen Entitäten nachspüren (ebd.: 12). In diesem Sinne ist Code in der Ausführung immer situiert (vgl. Suchman 2007). Damit sind Algorithmen mit Introna nicht nur Teil einer Assemblage, sondern auch – und das ist hier bedeutsam – im zeitlichen Fluss ihrer Entfaltung zu verstehen: Es besteht ein „temporal flow of action" (Introna 2016: 20). Das Wirken von Algorithmen im zeitlichen Fluss ist dabei nicht nur die Ausführung von miteinander verketteten Prozessen durch Programmierer:innen, sondern auch ihre Ausführung als welterzeugende Praxis, die jene Objekte, Entitäten oder Akteur:innen, auf welche die Prozesse Bezug nehmen, mit hervorbringen.

Diese Situiertheit ist allerdings nicht einfach zu fassen. Neben der hermeneutischen Interpretation des Codes selbst wird hier die Genealogie des Codes in seiner Entstehungsgeschichte relevant: Die verschiedenen Versionen, Überarbeitungen und Weiterentwicklungen ermöglichen es, die Situiertheit – zumindest in Teilen – auch im Nachhinein nachzuvollziehen (Kitchin 2017: 9).

Auf eine andere Weise legt Adrian Mackenzie (2005) seinen Schwerpunkt auf die Performanz, die auch in dem steckt, was häufig als ‚Produkt' verstanden wird: Software, bestehend aus Code, wird dabei als Objektivierung einer Praxis verstanden. Dies setzt Mackenzie in Bezug zur Sprechakttheorie von Paul Austin (1962), die bereits in vielfältigen Bezügen auf Performativität hin weitergedacht und von Text auf andere Entitäten erweitert wurde (wie etwa bei Judith Butler oder Bruno Latour). Computercode ist in diesem Zusammenhang „objectification of a linguistic praxis" (Mackenzie 2005: 76), Ausdruck einer Praxis, mit der Welt performativ erschaffen wird.

Gerade die Veränderbarkeit von Code ist es dann, die seine Performanz ausmacht. Sie ist aufbauend auf vorhergehenden performativen Akten und ohne diese nicht oder nur teilweise verständlich. Mackenzie macht dies am Beispiel des Betriebssystems Linux deutlich, wo verschiedene Distributionen der Software aufeinander aufbauen (ebd.: 78). Performativ werden in dieser „collective agency" Computercode und das Softwaresystem konstituiert (ebd.: 73), wobei auch technische Objekte eine Performativität haben und sich somit gleichzeitig soziale Praxis objektiviert. Algorithmische Handlungsmacht schlägt sich also in Strukturen nieder. Genau an dieser Schnittstelle spielt dann Computercode eine zentrale Rolle in der Analyse.

Einen weiteren Ansatz beschreiben Katrin Amelang und Susanne Bauer, die einer konkreten Risikobewertungssoftware beziehungsweise einem Algorithmus in seinen unterschiedlichsten Formen folgen: dem sogenannten arriba-Rechner, der bei der Entscheidungsfindung in medizinischen Kontexten unterstützen soll. In Anlehnung an Susan Leigh Stars Ethnografie der Infrastrukturen (1999) folgen sie dabei konkret den (menschlichen und nicht-menschlichen) Akteur:innen in den alltäglichen Translationen und Zirkulationen des

Algorithmus (Amelang & Bauer 2019: 480). Ihr methodisches Vorgehen beschreiben sie dabei mit Hugh Gusterson (1997) als „polymorphous engagement" (Amelang & Bauer 2019: 481), welches wir im Folgenden im Originalwortlaut der Autorinnen wiedergeben möchten:

> „Between spring 2014 and winter 2015/16, we followed the algorithm in order to examine its encounters with medical practitioners, patients, health insurers, software developers and epidemologists. Repeatedly locating the algorithm in the diverse sites involved in its production and circulation led us to, for example, doctor-patient consultations in primary care in Berlin and the Frankfurt Rhine-Main area, telephone conversations with many kinds of health-professionals, software interfaces, a crash-course in epidemiology for qualitative social scientists working in public health, and a university class for medical students. We participated in and observed various courses, conferences as well as with three practitioners in their patient consultations. Moreover, we followed the algorithm through various kinds of documents: scientific journals, medical guidelines, risk charts, health information leaflets, professional journals associated with health insurance companies and doctors' associations, websites, online-discussion forums and extracts of computer code" (ebd.: 481–482).

Durch dieses Following-the-Algorithm-Vorgehen decken die Autorinnen nicht einen, sondern multiple arriba-Algorithmen auf, welche jeweils diverse verschiedene Probleme lösen und verschiedene Akteur:innen wiederum unterschiedlich in Beziehung zueinander bringen (ebd.: 495). Auch hier ist die Performanz eine entscheidende analytische Perspektivierung.

Dabei bietet sich die Methode der Walkthroughs an, in welcher gemeinsam mit den Programmierenden Code betrachtet und besprochen wird. Ben Light, Jean Burgess und Stefanie Duguay (2017) haben diese Walks für die Erforschung von Apps in die Methodendiskussion eingebracht, die sich besonders eignen, um die Handlungen am Beispiel nachzuvollziehen und die Interviewten zum Reflektieren ihrer eigenen Praxis anzuregen. In Weiterentwicklung der Methode von Light, Burgess und Duguay hat Amelang

> „drei Vorschläge für App-spezifischere Forschungsinteraktionen [gemacht]: Apps via App-Stores zu besichtigen und so in ihrer Vielfalt und ihren Verbindungen zu kartieren; sie via ihrer Bedienoberflächen autoethnografisch testend zu durchlaufen; und sie als mitgestaltendes, kollaboratives Werkzeug im Interview zu nutzen." (Amelang 2023: 27)

Dies lässt sich auf Computercode übertragen, indem man ihn in seiner veröffentlichten Form analysiert, in seiner Performanz untersucht und – in enger Anlehnung an die Walkthroughs, die auch Amelang nennt – aktiv in die ethnografische Erhebung einbezieht.

Ein weiterer möglicher Ansatzpunkt besteht in der Beschäftigung mit den vielfältigen Translationsprozessen, die bei der Interaktion mit Software über Interfaces (auch Schnittstellen) ablaufen – ob bei der Verwendung eines Betriebssystems, der Nutzung einer Online-Plattform, eines Fitness-Trackers oder bei der Interaktion mit Augmented-Reality-Anwendungen. Hier kann auf Studien aufgebaut werden, die sich explizit mit Interfaces beziehungsweise Interfacing beschäftigen. Die Verbform betont den performativen und prozessual gedachten Charakter des Konzepts (Nake 2008; Galloway 2012; Shah 2019).

Ein Beispiel liefert Benjamin Lipp mit seiner Studie zu Praxen des „Human-Machine Interfacing[s]" (2022). In seiner Forschung zu Informatiker:innen, die am Einsatz von Robotern in der Pflege arbeiten, zeigt Lipp wie Roboter, Menschen und auch deren räumliche Umgebungen kontinuierlich und situativ überarbeitet werden, um füreinander verfügbar zu sein

(Lipp 2022: 10). Im Zuge seiner Forschung fertigte Lipp Feldnotizen an, führte Interviews und bezog Dokumente ein, die aus dem Projekt hervorgingen, das er teilnehmend beobachtete. Lipp zog ergänzend die Methode der Videoethnografie hinzu, die er als „auxiliary tool to accompany and reflect on my ethnographic observation" (ebd.: 5) nutzte.

Ethnografisches Beispiel: Entwicklung von interaktiven Technologien

In Rahmen ihrer Promotionsforschung zur Entwicklung von interaktiven Technologien im Spannungsfeld von wissenschaftlicher Wissensproduktion, Innovationsversprechen und Zukunftspraxen hat Sarah Thanner über drei Jahre hinweg ein medieninformatisches Forschungsprojekt begleitet, in dem auf den Alltagsgebrauch ausgerichtete Augmented-Reality-Anwendungen für interaktive Tische entwickelt werden. Dabei hat sie den Entwicklungsprozess ethnografiert, kollaborativ mit dem Entwickler:innen-Team zusammengearbeitet und sich etwa an der Konzeption und Durchführung von Prototypenausstellungen beteiligt, in deren Rahmen partizipative Designmethoden zum Einsatz kamen.

Bei der Untersuchung der Herstellungsseite interaktiver Technologien als Mensch-Technologie-Relationen im Werden (Thanner & Vepřek 2023), spielte in Thanners Forschung auch die Performanz des Computercodes der im Projekt entwickelten Software-Prototypen eine Rolle. Zentral war hierbei das Wirken des Codes in seiner Ausführung im Zusammenspiel mit menschlichen Akteur:innen, RGB- und Tiefenkameras, Tischoberflächen, Lichtverhältnissen und digitalen Projektionen als im zeitlichen Fluss miteinander verwobene Handlungslinien. Um sich diesem Zusammenspiel methodisch anzunähern, hat Thanner einerseits den Entwicklungsprozess teilnehmend beobachtet. So war sie etwa bei Projektmeetings anwesend, hat die Arbeit der Entwickler:innen an verschiedenen Prototypen und deren fortlaufende Weiterentwicklung beobachtet und nahm aktiv an öffentlichen Demonstrationen der Prototypen im Rahmen der gemeinsam organisierten partizipativen Ausstellungen und deren Vorbereitung teil. Andererseits führte sie qualitative Interviews mit den Entwickler:innen. Thanner, die selbst keine Programmierkenntnisse besitzt, ließ sich dabei von den Programmierenden die von ihnen im Code implementierten Abläufe und getroffenen Designentscheidungen erläutern und im Zuge von Walkthroughs ausgewählte Code-Abschnitte und deren Ausführung zeigen. Somit sichtete und analysierte sie Code-Fragmente, die die Entwickler:innen als relevant erachteten, um das Funktionieren der von ihnen entwickelten Augmented-Reality-Anwendungen und deren Interaktionsdesign zu erklären. Dies ermöglichte einen Zugriff auf im Code eingeschriebene Vorstellungen von User:innen-Körpern, zum Beispiel die Übersetzung von Fingern in geometrische Formen oder von Berührungsereignissen in Tiefenwerte im Kontext von Algorithmen der Objekterkennung.

Bei diesem methodischen Vorgehen ist der Einbezug und die Selektion der zu analysierenden Code-Abschnitte also stark durch die Programmierenden selbst beeinflusst und es kann passieren, dass Abschnitte, die vielleicht relevant für die zugrundeliegende Fragestellung gewesen wären, nicht gezeigt werden.

Bei den Interviews hat Thanner auch die Prototypen selbst als Artefakte einbezogen – die über dem Tisch aufgehängten Kamera-Projektor-Systeme und angeschlossenen Computer, auf denen der Code lief, waren auch in den Interviews präsent, wodurch es möglich war, diese gemeinsam auszuprobieren und damit auch nicht-sprachliches Handeln in die Inter-

viewsituation einzubeziehen. Die Performanz des Codes war so für Interviewerin und Interviewte direkt erfahrbar – auch dann, wenn etwas nicht funktionierte. Ein solcher Zugang bringt den Vorteil mit sich, dass auch dem sinnlich-ästhetischen und körperlichen Erfahren der Performanz von Computercode methodisch nachgespürt werden kann. Das verdeutlicht auch, dass Teilnehmende Beobachtung und Interviews hier situativ fließend ineinander übergehen können.

Darüber hinaus fertigte Thanner Feldnotizen und Fotografien an, in denen sie zum Beispiel die fortwährende (Re-)konfigurationsarbeit der Entwickler:innen dokumentierte, die das Installieren der Prototypen in unterschiedlichen situativen Kontexten erforderte. Dieses Vorgehen erlaubte, die Praxen des Infrastrukturierens und des Interfacings etwa hinsichtlich Raumbeschaffenheiten und wechselnder Lichtverhältnissen zu untersuchen, die fortwährende Anpassungsleistungen im Code erforderlich machten. Hierbei erlaubte der Einbezug von Meeting-Protokollen, gezeichneten Storyboards sowie Fachartikeln aus dem Feld der Human-Computer-Interaction, die sich mit Technologien der kamerabasierten Objekterkennung beschäftigen, auch eine methodische Annäherung an die Modi wissenschaftlicher Wissensproduktion, in die Programmierpraxen und das Wirken des Computercodes im untersuchten Gegenstandsfeld eingebettet waren.

Programmieren als Praxis: Entwickeln, Basteln, Debuggen und Hacken

Nähern wir uns Coden oder Programmieren als Praxis, verschiebt sich der Fokus weg von Computercode als Objekt mit eigener Handlungsmacht hin zu den Praxen derjenigen, die Code entwickeln und bearbeiten. Selbstverständlich können und sollen diese an und mit dem Code auch in ihrer Performanz nachvollzogen werden, was erneut aufzeigt, dass die einzelnen Dimensionen von und Perspektiven auf Computercode nicht klar voneinander getrennt werden können. An dieser Stelle liegt der Schwerpunkt jedoch vermehrt auf der Frage danach, wie Code hergestellt wird und nach den damit einhergehenden sozio-materiellen Praxen der Entwickler:innen. Diese bestehen aus dem Programmieren, dem Schreiben und Umschreiben von Code oder dem Hinzufügen von Codefragmenten Anderer. Weitere Praxen an und mit Code, die über die direkte Entwicklung von Software hinausgehen, sind zum Beispiel Modding, also das Austesten von Lücken, sowie Hacking, also die Problemlösung auf nicht vorgesehene Weise. Bei Letzterem werden oft Lücken oder Fehler im Code oder Schnittstellen ausgenutzt, um beispielsweise ein System zum Absturz oder zur Ausführung nicht-intendierter Aktionen zu bringen.

Coden als Praxis besteht bei genauerem Hinsehen nicht nur aus dem neuen Implementieren von Algorithmen, sondern ebenso aus dem Importieren bestehender Programmbibliotheken, dem Verknüpfen bestehender Einzelteile oder auch dem Recherchieren von Lösungen, die andere bereits für ein bestimmtes Problem entwickelt haben. Häufig ist die Arbeit am Code dabei in spielerisch-explorative Praxen eingebunden, die Annemarie Mol als *tinkering* beschreibt (Mol et al. 2010; zu *tinkering* im Coden auch Coleman 2009). Darüber hinaus muss Computercode dokumentiert werden, damit er mit anderen geteilt und gemeinsam sinnvoll an diesem gearbeitet werden kann. Praktiken des Dokumentierens von Code sind dabei selbst vielfältig und begleitet von Normen und Taktiken. Beispielsweise kann das bewusste Erstellen von schlechter oder lückenhafter Dokumentation (*Sloppy Documentation*) als

ein „subversiver Weg [verstanden werden], um Arbeitsverträge zu verlängern“ (Amrute 2016: 101), indem sich Programmierer:innen unentbehrlich machen.

Auch hier gilt es, Programmierpraxis nicht auf das Tippen von Zeichen in eine Kommandozeile oder auf die Handlungen in einer integrierten Entwicklungsumgebung (*Integrated Development Environment*, IDE) zu reduzieren. Im Mittelpunkt dieser Perspektivierung stehen also alle Handlungen, die Menschen mit Computercode tun und die sie vornehmen, um Code überhaupt erst erzeugen und instandhalten zu können. Auch wenn wir hier den Fokus auf das Programmieren im Allgemeinen legen, zählen zu den Praxen ebenso das Debuggen und Testen von Code, welches selbst in unterschiedlichen Formen stattfindet. Forschende entscheiden mit der Berücksichtigung dieser Dimension, vor allem die Menschen und ihre Intra-aktionen mit Code zu untersuchen.

Was beispielsweise ein Softwareprodukt können und wie es strukturiert sein soll, wird häufig im Rahmen von zahlreichen Teambesprechungen ausgehandelt und ist damit auch Teil von Arbeitskulturen, die oft in spezifischen, agilen Projektzusammenhängen entstehen und kollaborative Werkzeuge zentral nutzen, um Wissen zu managen und gemeinsam Probleme zu lösen (Tischberger 2021). Genauso können Diskussionen über Programmierprobleme auf Plattformen wie Stackoverflow oder Github dabei eine wichtige Rolle spielen. Darüber hinaus muss die materielle Umgebung miteinbezogen werden – von den Räumlichkeiten und Eingabegeräten wie Tastaturen und Computermäusen über Projektpläne oder Whiteboards und vieles mehr. Ronja Trischler (2022) hat beispielsweise mit ihrer Studie zur Produktion von Videoeffekten gezeigt, inwiefern die arbeitsteiligen Herstellungspraxen an Materialitäten wie Bürostühle, Bildschirme oder Sichtungszimmer gekoppelt sind.

Dabei ist zu beachten, dass die Praxis des Programmierens in der Regel nicht durch Einzelpersonen erfolgt, sondern kollaborativ von Gruppen realisiert wird, die sich vor Ort oder über digitale Kommunikationskanäle, oft asynchron, miteinander abstimmen: Algorithmen bestehen dann also „of original formulations mashed together with those sourced from code libraries, including stock algorithms that are re-used in multiple instances“ (Kitchin 2017: 7). Als Praxen versteht Seaver auch die kulturellen Interaktionen, an den Code teilnimmt, „culturally enacted by the practices people use to engage with them“ (Seaver 2017: 5). Hier wird deutlich, dass die Grenzen zur Dimension von Code als Teil einer Assemblage nicht immer eindeutig sind.

Wenn die Praxis des Programmierens im Fokus ethnografischer Forschung steht, kann und muss dies je nach Fragestellung auch bedeuten, die Alltagswelten der Programmierenden in den Blick zu nehmen und ihnen über ihre Zeit am Computer hinaus zu folgen. Dies hat etwa Sareeta Amrute (2016) in ihrer Studie über IT-Arbeiter:innen indischer Herkunft in Berlin gezeigt. Gabriella Coleman (2013) beschäftigte sich mit den Ethiken und Selbstverständnissen von Hacker:innen in der freien und offenen Software-Entwicklung und deckte dabei auch deren Widersprüche und Vielfältigkeiten auf.

Existierende Ansätze nähern sich Programmierpraxen methodisch meist durch die Teilnehmende Beobachtung mit Entwickler:innen, die etwa in Meetings mit anderen Entwickler:innen, Kolleg:innen oder Kund:innen oder aber quasi zwischen Bildschirm und Stuhl umgesetzt werden. Als Ethnograf:innen können wir physisch präsent gemeinsam mit Programmierer:innen vor dem Bildschirm sitzen oder auf Distanz über einen geteilten Bildschirm beobachten.

Roman Tischberger (2020) geht im Zusammenhang mit der bereits genannten agilen Arbeit in der Softwareentwicklung der Alltäglichkeit von Fehlern und dem Umgang mit diesen

nach. Seine Ergebnisse basieren dabei besonders auf Beobachtungsprotokollen, die im Rahmen seiner Teilnehmenden Beobachtung in einer Softwarefirma entstanden sind, sowie auf qualitativen Interviews. Durch das Begleiten von Entwickler:innen bei der Erledigung von Programmieraufgaben kann Tischberger Alltäglichkeiten aufzeigen, die uns durch das Interpretieren des Textes von Computercode oder der performativen Ausführung dessen verborgen bleiben würden. Die vielschichtigen Praxen im Entstehungsprozess des Codes sind es, die erst durch die ethnografische Beobachtung und Teilnahme zugänglich werden. Denn Code wird in der Regel mehrfach modifiziert, Fehler werden beseitigt und immer wieder neue Elemente eingefügt, bevor Code in ein Softwareprodukt integriert oder auf einem Repository zur Nachnutzung veröffentlicht wird. Er ist somit als Work-in-Progress zu verstehen. Den damit verbundenen Prozessen folgend, können ethnografische Methodenkombinationen besonders gewinnbringend sein.

Einen Schritt weiter geht das autoethnografische Vorgehen, in dem Forschende nicht nur beobachten, sondern selbst programmieren (lernen) und teils aktiv an der Produktion von Code mitarbeiten.

Ethnografisches Beispiel: Veränderung wissenschaftlicher Forschungsprozesse

In ihrer Forschung zur Veränderung wissenschaftlicher Forschungsprozesse durch digitale Methoden beforschte Lina Franken verschiedene Wissenschaftler:innen, die sich den Digital Humanities oder den Computational Social Sciences zugehörig fühlen. Hier wird Computercode in die eigene wissenschaftliche Praxis eingebunden – Code wird etwa genutzt, um große Mengen Text zu untersuchen (vgl. zu Anwendungen in qualitativer Forschung den Beitrag von Franken et al. in diesem Band und Vepřek in diesem Band). Im Rahmen ihrer Forschungen begleitete sie zwei Projekte über mehrere Jahre, nahm aktiv an den Forschungen teil und untersuchte dabei auch selbst, welche Rolle digitale Methoden für ihre Fragestellungen spielen oder spielen können.

Zu Beginn des ersten Projektes besuchte Franken gemeinsam mit Feldpartner:innen einen Einführungskurs ins Programmieren mit Python und diskutierte in den folgenden Monaten die Weiterentwicklung der dort erlernten Grundlagen mit verschiedenen Akteur:innen. Dafür setzte sie sich mit den Personen zusammen und ließ sich Code zeigen, der selbst geschrieben worden war. Das gemeinsame Sitzen vor einem Bildschirm erwies sich dabei als hilfreich, wobei hier nicht im engeren Sinne Walkthroughs erfolgten, sondern die gemeinsame Bearbeitung von Datensätzen mit dem Ziel der wissenschaftlichen Analyse im Mittelpunkt stand. Diese Diskussionen entwickelten sich immer wieder auch zu Formaten der gemeinsamen Reflexion. Sie wurden jeweils im Nachgang in Feldnotizen festgehalten. Auch im zweiten Projekt erfolgten gemeinsame Sichtungen des erstellten Codes anhand unterschiedlicher Datensätze, in diesem Fall durch Videokonferenz vermittelt.

Auf Teile des Codes hatte Franken auch im Nachhinein Zugriff. Sie probierte die Skripte selbst mit ihrem Material aus und stellte Rückfragen, die in Treffen, per E-Mail und telefonisch besprochen wurden. Der Code wurde so zwar nicht selbst programmiert, aber in seiner Ausführung nicht nur gesehen, sondern auch selbst ausgeführt. Auch diese Schritte wurden in Feldnotizen festgehalten, wobei sich im Nachhinein besonders die Irritationen und Missverständnisse als produktiv erwiesen, da sie weitere Erklärungen und Diskussionen hervorriefen. Dabei ermöglichte die autoethnografische Komponente dieses Vorgehens zudem, das

Selbsterfahren der Praxis als Instrument analytischer Reflexion nutzbar zu machen und durch die geteilte Erfahrung eine Gesprächsbasis für die sich daraus ergebenden Diskussionen mit Forschungspartner:innen. In einigen Fällen führten diese Diskussionen und gemeinsamen Code-gestützten Untersuchungen von Quellenmaterial schließlich auch zu Interviews, in denen die Forschungspartner:innen weiterführend über ihre eigene Rolle und ihr methodisches Vorgehen reflektierten und auch (Zwischen-)Ergebnisse der Analyse diskutiert wurden. Dieser Zugang erlaubte es zudem, das *tinkering* mit dem Code einzubeziehen. Auf diese Weise wurden etwa auch die beim Programmieren gemachten Fehler sichtbar sowie die verschiedenen Versionen des Codes, die in späteren Veröffentlichungen nicht mehr nachvollziehbar gewesen wären.

Die Entwicklung von digitalen Methoden in den Geistes-, Sozial- und Kulturwissenschaften und deren Etablierung als eigenständige Forschungsrichtungen oder Disziplinen geht darüber hinaus mit Praxen des Infrastrukturierens durch vielfältige Akteur:innen einher: Forschende, wissenschaftspolitische Vertreter:innen, Verbände, Institutionen wie Bibliotheken oder Forschungs- und Servicezentren, Universitätsgremien und viele mehr. Methodischen Zugang zu dieser Perspektive gewährten insbesondere öffentliche Veranstaltungen, die zum Beispiel über einschlägige E-Mailverteiler beworben wurden, oder auch auf Empfehlung von Forschungspartner:innen ausfindig gemacht werden konnten. Ergänzenden Einbezug fanden auch verschiedene Dokumente und Websites. Das auf diesem Wege generierte Material gewährte Aufschluss über die mit Code einhergehenden und durch ihn geprägten Politiken und Fachdiskurse. In diesem Sinne wurde die ethnografische Perspektive also um eine diskursanalytische ergänzt, um gouvernementale Ordnungen analytisch nachzuvollziehen.

Infrastrukturieren mit und durch Computercode

Eine weitere Dimension von Computercode tritt hervor, wenn wir uns der Rolle von Computercode bei der Hervorbringung von (digitalen) Infrastrukturen zuwenden. Computercode ermöglicht unterschiedlichste in unser Alltagshandeln eingebettete Infrastrukturen, die ohne den zugrundeliegenden Code nicht funktionieren würden und trägt zu deren kontinuierlicher Reproduktion bei. Gleichermaßen wird Computercode selbst erst durch bestimmte Infrastrukturen hervorgebracht. Beschäftigen wir uns aus Infrastrukturperspektive mit Computercode, so rücken die relationalen Bezüge von Computercode zur Herstellung und Verhandlung von technischen Strukturen und Arbeitsroutinen in den Vordergrund. Aber auch die organisationalen, technologischen sowie ökologischen Ressourcen und deren Gebrauch durch heterogene soziale Gruppen werden relevant. Forschungsentscheidungen gehen dabei weg vom konkreten Code und den Praktiken des Code-Herstellens und hin zu den techno-sozialen Zusammenhängen, in denen diese eingebunden sind und durch die sie hervorgebracht werden. Somit wird das alltägliche Gemacht-Werden von Infrastrukturen betont, die oftmals nur unbewusst wahrgenommen werden oder die besonders dann hervortreten, wenn sie versagen oder zusammenbrechen (Star 1999; Star & Ruhleder 1996). Auch Computercode beziehungsweise die auf diesem aufbauende Software bleibt in der Nutzendenperspektive in der Regel unsichtbar: Die Benutzeroberflächen werden genutzt, ohne dass sich Gedanken über die Algorithmen gemacht werden. Erst wenn etwas nicht funktioniert, wird diese dahinterliegende Infrastruktur für die Nutzenden offensichtlich, insbesondere wenn sie sich mit den etwaigen Problemen beschäftigen müssen, die durch den

Zusammenbruch für sie entstehen (Mackenzie 2005: 72). Digitale Infrastrukturen müssen dabei immer in ihren relationalen Zusammenhängen untersucht werden (Koch 2017). Um diese eigentlich unsichtbaren Phänomene zu beforschen ist es notwendig zu verstehen, dass „the nature of infrastructural work involves unfolding the political, ethical, and social choices that have been made throughout its development" (Bowker et al. 2010: 99). Die Beschaffenheiten und Wirkweisen von Computercode prägen die Möglichkeiten und Unmöglichkeiten der Inter- beziehungsweise Intra-aktion und ko-konstituieren dabei performativ auch die beteiligten Akteur:innen und Materialitäten auf spezifische Weise. Auch die Software Studies haben auf die materielle Dimension von Computercode hingewiesen und stellen damit den Zusammenhang zu jenen Infrastrukturen her, über die Software mit Gesellschaft verbunden ist (Reichert & Richterich 2015).

Dabei werden Infrastrukturen selbst nicht als statische Entitäten, sondern als emergente Phänomene gefasst, die zwischen Stabilität und Fragilität oszillieren und stets sowohl materiell als auch sozial sind (Niewöhner 2015; Bowker & Star 2006). Um dieses praxisorientierte Verständnis zu betonen, bietet es sich hier zudem an, von Infrastrukturieren als Verb zu sprechen, statt das Nomen Infrastruktur(en) zu verwenden (Niewöhner 2015: 9). In diesem Sinne werden Computercode, Betriebssysteme, Protokolle oder Datenbanken durch kontinuierliches Infrastrukturieren wirksam (Mackenzie 2005: 74).

Jörg Niewöhner (2015: 9–11) legt unter Verweis auf Geoffrey Bowker et al. (2010) dar, dass für die Analyse von Infrastrukturen im Sinne sozio-materieller Praxen ein multimodaler und multiperspektivischer Zugang erforderlich ist, bei dem unterschiedliche Methoden kombiniert werden, die wiederum unterschiedliche Daten hervorbringen (zum Beispiel dichte Beschreibungen, Narrative, technische Daten, Karten, usw.). Dabei geht es jedoch weniger darum, die Ergebnisse verschiedener Methoden zu einem integrativen Gesamtbild zusammen zu fügen, sondern „rather to explore how they relate to each other, always only partially connected, and how the constant failure of any perspective to capture something whole and complete starts to produce something interesting when put next to other failures" (Niewöhner 2015: 9). In ihrer bereits im Abschnitt zur Performativität erwähnten Studie zur Risikobewertungssoftware arriba-Rechner ließen Amelang und Bauer in ihrem Follow-the-Algorithm-Vorgehen von der Infrastrukturethnografie inspirieren (Amelang & Bauer 2019: 480) und zeigen auf, wie der Algorithmus infrastrukturierend wirkt, indem er Gespräche zur Vorsorge strukturiert. Gleichzeitig war und ist schon die Erzeugung der Software infrastrukturierend, denn in diese sind nicht nur konkrete Algorithmen, sondern auch durch Forschung entwickelte Risikofaktoren eingegangen. Letztere wurden etwa in Form von Tabellen hinterlegt, die mit Star (1999: 381) als in die Software eingebettete „Embodiment[s] of standards" (ebd.) betrachtet werden können. Die Darstellung der Risikofaktoren in der Infrastruktur wurde von den Entwickler:innen diskutiert und angepasst (hin zu Balkendiagrammen) (ebd.: 485). Darüber hinaus findet durch den Algorithmus ein „infrastructuring of public health in a much broader sense" (ebd.: 495) statt.

Governance und Gouvernementalität: Regieren mit und durch Computercode

Eine hier abschließend zu betrachtende Fokussierung bei der Analyse von Computercode nimmt Fragen von Gouvernementalität in den Blick und wendet sich den Regierungstechniken zu, in die Code eingebunden ist oder durch welche Code auf Gesellschaft und die Herausbildung von Subjektivitäten einwirkt.

Der Begriff der Gouvernementalität geht auf Michel Foucault (1978) zurück, der damit Macht-Wissen-Komplexe und Formen des Regierens beschreibt. Neben Regierungsformen, die sich zwischen Staat und Individuum oder in Institutionen, wie zum Beispiel der Schule finden, zählen für Foucault hierzu auch Techniken des Selbst (Foucault 1982), d.h. Regierungsformen, denen Individuen sich selbst unterwerfen und die Individuen dazu anrufen, fortwährend an sich selbst zu arbeiten.

Dabei legt das Konzept der Gouvernementalität den Fokus darauf, wie Macht in Handlungszusammenhängen immer wieder neu durch eine Pluralität von Akteur:innen hergestellt wird (Introna 2016: 27) – und weniger auf vermeintlich feststehende Formen des Regierens durch Institutionen oder Hierarchien. Wohl aber zeigt sich Gouvernementalität nicht nur in einzelnen Akteur:innen, sondern auch in den diese regierenden Institutionen. Damit geraten auch die mit diesen Regierungstechnologien verbundenen Machtverhältnisse in den Blick ethnografischer Betrachtung, denn der strukturelle Deutungsanspruch von Macht wird in der algorithmischen Gegenwart auch durch und mit Computercode realisiert (ebd.). Studien mit Fokus auf diese Dimension interessieren sich also zum Beispiel dafür, auf welche Weise Computercode vor dem Hintergrund spezifischer Machtkonstellationen an der Hervorbringung von Individuen als Subjekte beteiligt ist.

In Bezug auf Algorithmen ist es mit diesem Ansatz sowohl möglich zu untersuchen, wie regierende menschliche Akteur:innen Computercode kontrollieren, als auch wie Code selbst als Akteur zur Umsetzung von Regierungstechnologien eingesetzt wird. Schließlich können Computeralgorithmen als Teil der Gouvernementalität im Sinne Foucaults eingesetzt werden und müssen dann auch in den entsprechenden Studien berücksichtigt werden (ebd.: 28–30). Code ist in das Geflecht von Machtstrukturen und -mechanismen ebenso wie andere Akteur:innen verstrickt und wirkt damit auf die Hervorbringung von Handlungsmöglichkeiten sowie die Disziplinierung von Individuen ein (Lemke 2001: 113).

Ethnografische Studien, welche die gouvernementale Dimension von Computercode berücksichtigen, fragen danach, welche Rolle diesen in Regierungstechnologien zukommt. Die damit einhergehenden Forschungsentscheidungen fokussieren also gesellschaftliche Setzungen, die von und mit Code etabliert, verstärkt oder verändert werden und nähern sich diesen mit Blick auf vielfältige unterschiedliche Dimensionen von Computercode an.

So untersucht etwa Lucas Introna (2016), anknüpfend an Foucaults Gouvernementalitätskonzept, Regierungspraktiken von und durch algorithmische Akteure und verweist hierbei insbesondere auch auf den performativen Charakter des gouvernementalen Wirkens von Algorithmen und algorithmischen Systemen: „the governance of algorithms, or through algorithms, must itself be understood as practices of governmentality in order to understand the doing of governance!“ (Introna 2016: 30). Dabei zeigt er, wie Regierungstechnologien mit Wissensregimen verknüpft sind und so spezifische Subjektivitäten hervorbringen (ebd.). Auch hierbei ist es wichtig darauf hinzuweisen, dass die Perspektiven auf und Dimensionen von Computercode nicht trennscharf voneinander zu unterscheiden sind. Introna etwa

spricht von „algorithmischen Assemblages“ (ebd.: 17; Übersetzung der Autorinnen), in welchen selbstverständlich die in den vorhergehenden Kapiteln beschriebenen Dimensionen ebenfalls eine Rolle spielen. Der Schwerpunkt dieser Perspektivierung liegt allerdings im Blick auf die sich verändernden Machtstrukturen, weswegen – so Introna – der methodisch-analytische Fokus weniger auf dem Interpretieren einzelner Codezeilen liegt als vielmehr auf der Analyse des performativen Zusammenwirkens und der relationalen Verwobenheit mit den Doings algorithmischer Akteure und ihrer berechnenden Praktiken. Die Performanz von Computercode wird in dieser Perspektivierung also insbesondere mit Blick auf das gouvernementale Wirken des Codes untersucht.

Das Regieren von Anderen spielt beispielsweise bei Computeralgorithmen eine Rolle, die Empfehlungen geben und so das Handeln von Menschen, oft implizit und für User:innen unbewusst, beeinflussen. Es finden sich eine Reihe von Studien zu spezifischen Plattformen, die mit Blick auf Regierungstechniken untersucht wurden, wie etwa Spotify (Kropf 2019; Werner 2020), AirBnB (Frisch 2019) oder Dating-Plattformen wie Tinder (Peetz 2021). Besonders Studien zu Predictive Policing, also der vorhersagenden Polizeiarbeit, untersuchen Regierungstechnologien (Reigeluth 2014; Weber 2018; Vepřek 2018).

Auch das Regieren des Selbst wurde bereits in verschiedenen Aspekten untersucht. Hervorzuheben sind hier etwa Studien zu Phänomenen, die unter den Begriffen des Selbsttracking oder Quantified Self gefasst werden. Diese befassen sich mit auf Optimierung, Steuerung und Kontrolle abzielenden Selbstführungstechniken als algorithmisch strukturierte Teile gouvernementaler Normierungen. Während Carsten Ochs und Barbara Büttner (2019) beispielsweise eine Fitness-Plattform auf die Verknüpfung von Daten, Körpern und Alltagspraxen hin untersuchen, beforscht Lisa Wiedemann (2019) die Verwobenheiten von Körper- und Gesundheitspraxen mit digitalen Geräten im Kontext von Diabetes. Dabei zeigt sie, wie Fremdführung und Selbstführung miteinander verbunden sind: Durch die Geräte und die ihnen eingeschriebenen Algorithmen werden einerseits gesundheits- und biopolitisch relevante Daten erhoben, die etwa zur Risikofrüherkennung genutzt werden und auf diese Weise potenziell Einfluss auf gesundheitspolitisches Handeln nehmen können. Andererseits dient diese „Form der numerisch angeleiteten Selbstführung“ (ebd.: 53), so Wiedemann, Diabetiker:innen im alltäglichen Umgang mit ihrer Erkrankung als Hilfestellung, zum Beispiel durch das Zählen von Kalorien bei der alltäglichen Nahrungsaufnahme. Durch diese Codegestützte Form der Selbstkontrolle werden „zyklische Feedbackschleifen […], die nahtlos und automatisch im Hintergrund mitlaufen“ (ebd.: 54) produziert, in die Individuen durch Machtstrukturen und Politiken verwoben sind. Zu ähnlichen Ergebnissen kommt Hannah Rotthaus (2020) in Bezug auf die Verwendung von Verhütungsapps.

Weitere Beispiele für Studien, die aus gouvernementaler Perspektive auf algorithmische Systeme blicken, finden sich bei Introna (2016; 2017). Er beschäftigt sich dabei unter anderem mit der Anwendung der Plagiatssoftware Turnitin und deren Algorithmen sowie den durch sie hervorgebrachten Berechnungspraktiken im Sinne von Regierungstechniken im Kontext akademischen Schreibens (2016: 33). Introna stellt aus gouvernementaler Perspektive die Frage, inwiefern die Übernahme von Textpassagen Anderer im Kontext akademischen Schreibens als Betrug verstanden wird und welche Rolle das Wirken von Algorithmen der erwähnten Plagiatssoftware hier spielt. Der verwendete Algorithmus ermöglicht den Vergleich von Textpassagen in zeichengenauer Übereinstimmung. Die Ähnlichkeit wissenschaftlicher Texte wird hierbei also auf Zahlenwerte heruntergebrochen, die Aufschluss über den prozentualen Anteil von potenziell plagiierten Anteilen geben. Da allerdings nur zei-

chengenaue Textübernahmen gefunden werden, werden aus fremden Texten übernommene, aber sprachlich leicht veränderte Textteile hingegen nicht als Plagiat gewertet. Die Nutzung des Algorithmus ist es dann, durch die ein Regime der legitimen und illegitimen Praktiken des wissenschaftlichen Arbeitens mithervorgebracht wird. Der Algorithmus selbst ist damit also nur spezifischer Ausdruck einer Gouvernementalität (ebd.: 38).

Ebenfalls fruchtbare Ansätze zur Beschäftigung mit dem gouvernementalen Wirken von Computercode finden sich in der Untersuchung der Voreingenommenheit (auch Bias) von Algorithmen und algorithmischen Systemen. Dabei sind es von Menschen generierte Daten, die in engem Zusammenhang mit ebenfalls von Menschen generiertem Computercode dazu führen, dass bei der Ausführung dieses Codes Voreingenommenheit (re-)produziert wird. Ntoutsi et al. (2021) unterscheiden hierbei zwischen Studien, die versuchen die Voreingenommenheit nachzuvollziehen und zu verstehen, Studien, die um die Abschwächung dieser Bias bemüht sind, und Studien, die aufzeigen, dass das Wissen über Voreingenommenheit bei der Erstellung von Code einbezogen werden muss (ebd.: 3).

Diesen Studien ist gemeinsam, dass sie die Relevanz von Algorithmen in der Analyse zentral setzen, sich dabei jedoch nicht oder kaum mit dem Computercode direkt als Text beschäftigen, sondern sich diesem in seiner politischen Relevanz nähern. Angèle Christin legt etwa dar, dass Forscher:innen häufig von Designkritik und *close readings* von „industry publications, promotional material, and journalistic articles about algorithms" (Christin 2020: 4) ausgehen, „which they mobilize to connect recent incidents within longer historical trajectories" (ebd.). Christin beschreibt außerdem die Methode „algorithmischer Audits", die nur die In- und Outputs algorithmischer Systeme mittels computerbasierter und statistischer Methoden untersuchen (ebd.: 3). Ein Fokus liegt dabei besonders auf diskriminierenden Auswirkungen. Allerdings reproduzieren diese Vorgehensweisen Algorithmen als Black Box, die nur durch ihre In- und Output-Relationen erfassbar sind (Seaver 2017). Jenna Burrell (2016) verwendet den gleichen Begriff des Audits in ihrer Forschung zur Undurchsichtigkeit von Machine-Learning-Algorithmen, setzt sich methodisch jedoch direkt mit diesen auseinander. Zum Beispiel untersucht sie, wie diese Algorithmen zu bestimmten Klassifizierungen, etwa von Spamnachrichten, kommen und macht so die internen Logiken dieser Modelle, zumindest bis zu einem bestimmten Grad, nachvollziehbar. Schließlich verweist Amrute in ihrer Forschung zu indischen Programmierer:innen in Berlin auf deren Kritik an unternehmerischen Programmierpraktiken, die einzig darauf abzielten, Kundenwünsche zu erfüllen und nicht dabei helfen, politische Grenzen aufzulösen und gemeinsam Probleme zu lösen (Amrute 2016: 21).

Computercode in Assemblagen ethnografisch beforschen: methodische Zugänge in ihrer Bandbreite

In den vorangegangenen Abschnitten haben wir uns einigen der vielen Dimensionen, denen wir in ethnografisch-kulturwissenschaftlicher Forschung begegnen können, angenähert und mögliche methodische Vorgehensweisen erläutert. Die vorgestellten Dimensionen von und damit einhergehende Perspektiven auf Computercode sollen jedoch keinesfalls als sich gegenseitig ausschließend verstanden werden. Vielmehr bilden sie mögliche Schwerpunktsetzungen oder Ausgangspunkte für den Einbezug von Computercode in kulturwissenschaftlich-ethnografisches Forschen, die relational aufeinander verweisen. Computercode muss stets als eingebettet in weitere soziotechnische Assemblagen verstanden werden, die Introna

(2016) auch als *algorithmic assemblages* versteht, in denen „many of the traditional institutional actors (…) have become malleable and interconnected" (ebd.: 19). Kitchin und Lauriault (2018) arbeiten bezogen auf Daten mit einem vergleichbaren Konzept, dem der *data assemblages*, „that produce, circulate, share/sell, and utilize data in diverse ways" (ebd.: 6). Ein ähnliches Konzept entwickelt auch Christin mit den „algorithmic enrollments" (Christin 2020: 913). Für die Analyse dieser Assemblagen oder *enrollments* ist ein multiperspektivisches und multidimensionales Vorgehen grundlegend, das Assemblagen als „always in a state of becoming" (Kitchin & Lauriault 2018: 7) versteht und die Situiertheit von Code in der Analyse berücksichtigt. Kitchin weist darauf hin, Code sei

> „embedded within complex socio-technical assemblages made up of a heterogeneous set of relations including potentially thousands of individuals, data sets, objects, apparatus, elements, protocols, standards, laws, etc. that frame their development." (Kitchin 2017: 7)

Auch Nick Seaver (2017) fordert, Algorithmen als Teil von Kultur und durch kulturelle Praxen hervorgebracht zu betrachten und damit das Augenmerk auf ihre Heterogenität und Multiplizität zu legen. Ihm zufolge gilt es, jene alltäglichen Praxen und deren sozio-materielle Einbettung zu untersuchen, die Computercode und Algorithmen hervorbringen, Instandhalten und immer wieder aufs Neue rekonfigurieren (ebd.).

Die vorgestellten Herangehensweisen sollen somit in erster Linie mögliche Wege methodischer Hinwendungen zu Computercode als Teil des ethnografischen Methodenrepertoires aufzeigen, die dazu beitragen, Code erkenntnisbringend in die eigene Forschung einbinden zu können. Auch wenn dieser Einbezug keiner grundsätzlich neuen Methoden bedarf, so möchten wir Leser:innen doch dazu ermuntern, mit neuen methodischen Zugängen zu experimentieren. Experimentelle Ansätze können das bestehende facettenreiche Methodenspektrum unseres Fachs ergänzen, um eine stets dynamisch zu verstehende Kombination und Konfiguration verschiedener Perspektiven und mit ihnen einhergehender methodischer Zugänge. Amelang (2023) schlägt vor, für neue Phänomene wie Smartphone-Apps, aber auch für Algorithmen, auf bewährte, ‚alte' Methoden zurückzugreifen. Sie plädiert dafür, weniger neue Methoden in den Mittelpunkt der Überlegungen zu stellen, sondern vielmehr für einen „erweiterten Einsatz von Feldforschungsmethoden in digitalen Settings" (ebd.). Dies erscheint uns auch für den Einbezug von Computercode zielführend. Computercode kann also als neue Perspektive und auch Quellengattung in unseren Methodenbündel aufgenommen werden. Allerdings brauchen wir eine gesteigerte digitale Literalität und ein damit einhergehendes Verständnis für (das Wirken von) Computercode, aber auch für seine theoretisch-konzeptionellen Hintergründe, um diesen zum Forschungsgegenstand zu machen.

Im Mittelpunkt steht also vor allem die Erweiterung unseres Quellenspektrums mit Blick auf die Analyse des Wirkens von Computercode auf Alltagskulturen. Hierfür ist es notwendig, von einer prozessualen, nie abgeschlossenen Assemblage auszugehen und jeweils individuelle „*agentische Schnitte*" (Barad 2015: 109; Hervorh. i.O.) im eigenen Forschungsprojekt zu setzen. In diesem Schnitt ist „die Uneindeutigkeit nur temporär und kontextuell entschieden (…); und zwar auf eine Weise, die bestimmten Konzepten unter Ausschluss anderer Bedeutung verleiht" (ebd.: 55). Welche Ausschnitte wir in der betreffenden Assemblage betrachten, ist eine aktive Setzung von uns als Forschenden (vgl. Carlson et al. 2021). Anders laufen wir Gefahr, den Code in seiner Einzigartigkeit zu fetischisieren (Kitchin 2017: 12) und dessen Verwobenheit zu vernachlässigen.

Da Computercode oftmals – und dabei besonders weltstrukturierende Computeralgorithmen – nicht öffentlich zugänglich ist, bietet sich die Erforschung von Computercode besonders für kollaborative und interdisziplinäre Vorgehen an. Denn bei proprietärer Software, in der Code durch Schutzrechte nicht öffentlich ist, stoßen viele Vorgehensweisen an ihre Grenzen, da sie das Problem der Zugänglichkeit nicht lösen können. Bei der Analyse von Code als Text zum Beispiel, sind Such- und Sortierungs-Algorithmen oft nur über ein Reverse Engineering nachzuvollziehen und um die Programmierung solcher Software zu beforschen, bedarf es Zugang zu Institutionen oder Unternehmen. Für entsprechende Einblicke, die partiell bleiben, sind zudem Informatikkenntnisse notwendig, die interdisziplinäre Zusammenarbeit naheliegend machen. Nicht nur können so Zugänge geschaffen werden, sondern auch Schwerpunkte im individuellen Wissen und Fachlogiken im gemeinsamen Austausch erarbeitet werden. Das Zusammenarbeiten mit unterschiedlichen Epistemologien bringt jedoch auch Herausforderungen mit sich, wie beispielsweise Diana Forsythe bereits in den 1980er und 1990er Jahren in ihren Forschungen in US-amerikanischen KI-Laboren anschaulich beschrieben hat (Forsythe 2001).

Das weite ethnografische Methodenspektrum kann also gewinnbringend für die Analyse von Computercode genutzt werden: von diskursanalytischen Untersuchungen über den Einbezug historischer Archivalien (jüngeren Entstehungsdatums) bis hin zur Teilnehmenden Beobachtung, Befragung und Autoethnografie können alle Methoden, und gerade ihre Kombination, hilfreich sein. Zentral ist die Klärung der jeweiligen Perspektivierungen, mit denen wir auf unterschiedliche Dimensionen von Computercode blicken können. Es braucht also vielmehr neue Zuschnitte bestehender Methoden auf Code als Untersuchungsgegenstand als neue Zugänge zu diesem. Oder, um es mit Kitchin zu sagen: „We can therefore only know how algorithms make a different [sic!] to everyday life by observing their work in the world under different conditions“ (Kitchin 2017: 26).

In diesem Artikel haben wir ethnografische Begegnungen mit Code in fünf verschiedenen Dimensionen zusammengefasst. 1) Code als Text und seine Einschreibungen, 2) Computercode in seiner Performanz und Ausführung, 3) Programmieren als Praxis: Entwickeln, Basteln, Debuggen und Hacken, 4) Infrastrukturieren mit und durch Computercode, und 5) Governance und Gouvernementalität von Computercode: Regieren mit und durch Computercode. Diese nacheinander besprochenen Dimensionen, die keineswegs einer logischen als vielmehr rein durch die Textform einer linearen Ordnung folgen, spielen sich auf unterschiedlichen Skalen von Code ab, nehmen verschiedene Akteur:innen (sowohl menschliche als auch nicht-menschliche) in den Blick und bewegen sich zwischen Mikropraktiken und politischen Diskursen. Während diese Aufteilung für den vorliegenden Text sinnvoll und für eine ethnografisch-methodische Auseinandersetzung hilfreich ist und in der Metapher unterschiedlicher Zwiebelschichten auch grafisch aufgegriffen wurde, wären andere Aufschlüsselungen von Code ebenso denkbar. Beispiele für alternative Aufteilungen sind etwa bei Kitchin (2017) und Christin (2020) nachzulesen.

Ziel dieses Beitrages war es zu zeigen, dass Computercode für ethnografische Forschungen keine Black Box ist und sein sollte, sondern vielmehr über unterschiedlichste Zugänge eingebunden werden kann (und sollte). Die verschiedenen Dimensionen sollen als Hilfestellung dienen, um die Assemblagen, in welche Code verstrickt ist, in der Forschungspraxis zuzuschneiden, methodisch untersuchen zu können und bewusste Entscheidungen für entsprechende Schwerpunkte in der Erhebung und Analyse zu treffen. Dabei sollten allerdings deren Verwobenheiten und Übergänge nicht vergessen werden. Denn Code begegnet uns nicht nur im Alltag, sondern ebenso in der Forschung in unterschiedlichster Weise.

Anmerkungen

Die Forschung von Libuše Hannah Vepřek ist Teil des Forschungsprojekts „Spielend in the Loop: Neue Mensch-Software Relationen in Human Computation Systemen und deren Auswirkungen auf Sphären des Alltags“, gefördert durch die Deutsche Forschungsgemeinschaft (DFG) – 464513114.

Literatur

Amelang, Katrin (2023): Wie Apps erforschen? Zum Zusammentreffen neuer Forschungsgegenstände und alter Methoden. In: Hamburger Journal für Kulturanthropologie 16, 11–28.

Amelang, Katrin & Susanne Bauer (2019): Following the Algorithm. How Epidemiological Risk-Scores do Accountability. In: Social Studies of Science 49/4, 476–502. https://doi.org/10.1177/0306312719862049.

Amrute, Sareeta Bipin (2016): Encoding Race, Encoding Class. Indian IT Workers in Berlin. Durham, London: Duke University Press. https://doi.org/10.1215/9780822374275.

Austin, John L. (1962): How to Do Things with Words. The William James Lectures delivered in Harvard University in 1955. Cambridge Massachusetts.

Barad, Karen (1996): Meeting the Universe Halfway: Realism and Social Constructivism without Contradiction. In: Lynn Hankinson Nelson & Jack Neslon (Hgs.), Feminism, Science, and the Philosophy of Science. Dordrecht, Boston, London: Kluwer Academic Publishers, Springer, 161–94.

Barad, Karen (2015): Verschränkungen. Berlin: Merve.

Bourdieu, Pierre (1982): Die feinen Unterschiede. Kritik der gesellschaftlichen Urteilskraft. Frankfurt am Main: Suhrkamp.

Bowker, Geoffrey C., Karen Baker, Florence Millerand & David Ribes (2010): Toward Information Infrastructure Studies: Ways of Knowing in a Networked Environment. In: Jeremy Hunsinger (Hg.), International Handbook of Internet Research. New York: Springer, 97–117. https://doi.org/10.1007/978-1-4020-9789-8_5.

Bowker, Geoffrey C. & Susan Leigh Star (2006): How to infrastructure. In: Leah A. Lievrouw & Sonia Livingstone (Hgs.), Handbook of New Media and Communication, London: SAGE, 151–162. https://doi.org/10.4135/9781446211304.

Bucher, Taina (2012): Want to Be on the Top? Algorithmic Power and the Threat of Invisibility on Facebook. In: new media & society 14/7, 1164–1180. https://doi.org/10.1177/1461444812440159.

Burrell, Jenna (2016): How the Machine ‚Thinks‘: Understanding Opacity in Machine Learning Algorithms. In: Big Data & Society 3/1, 1–12. https://doi.org/10.1177/2053951715622512.

Carlson, Rebecca, Ruth Dorothea Eggel, Lina Franken, Sarah Thanner & Libuše Hannah Vepřek (2021): Approaching Code as Process. Prototyping Ethnographic Methodologies. In: Kuckuck. Notizen zu Alltagskultur und Volkskunde 1, 13–17.

Christin, Angèle (2020): The Ethnographer and the Algorithm: Beyond the Black Box. In: Theory and Society 49/5–6, 897–918. https://doi.org/10.1007/s11186-020-09411-3.

Chun, Wendy Hui Kyong (2008): On „Sourcery,“ or Code as Fetish. In: Configurations 16/3, 299–324.

Coleman, Gabriela (2009): Code is Speech. Legal Tinkering, Expertise, and Protest Among Free and Open Source Software Developers. In: Cultural Anthropology 24/3, 420–454. https://doi.org/10.1111/j.1548-1360.2009.01036.x.

Coleman, Gabriela (2013): Coding Freedom. The Ethics and Aesthetics of Hacking. Princeton: Princeton University Press. https://doi.org/10.2307/j.ctt1r2gbj.

Collier, Stephen J. & Aihwa Ong (2005): Global Assemblages, Anthropological Problems. In: Dies. (Hgs.), Global Assemblages. Technology, Politics, and Ethics as Anthropological Problems. Malden, Oxford: Blackwell Publishing, 3–21.

Deleuze, Gilles & Félix Guattari (1987): A Thousand Plateaus. Capitalism and Schizophrenia. Minneapolis: University of Minnesota Press.

Douglass, Jeremy, Mark C. Marino & Jessica Pressman (2020): Collaborative Reading Praxis. In: Electronic Book Review. https://doi.org/10.7273/B023-FX05.

Dourish, Paul (2016): The Stuff of Bits. An Essay on the Materialities of Information. Cambridge: The MIT Press.

Forsythe, Diana (2001): Studying Those Who Study Us: An Anthropologist in the World of Artificial Intelligence. Stanford: Stanford University Press. https://doi.org/10.1515/9781503619371.

Foucault, Michel (2005 [1978]): Die Gouvernementalität. In: Daniel Defert & Francois Ewald (Hgs.), Michel Foucault: Analytik der Macht. Auswahl und Nachwort von Thomas Lemke. Frankfurt am Main: Suhrkamp, 148–174.

Foucault, Michel (2005 [1982]): Subjekt und Macht. In: Daniel Defert & Francois Ewald (Hgs.), Michel Foucault: Analytik der Macht. Auswahl und Nachwort von Thomas Lemke. Frankfurt am Main: Suhrkamp, 240–263.

Frisch, Thomas (2019): Digitale Bewertungskultur im Tourismus 2.0. Grenzüberschreitung und Normalisierungsdruck. In: Jonathan Kropf & Stefan Laser (Hgs.), Digitale Bewertungspraktiken. Für eine Bewertungssoziologie des Digitalen. Wiesbaden: Springer VS, 41–70.

Galloway, Alexander R. (2012): The Interface Effect. Cambridge; Malden: Polity.

Gillespie, Tarleton (2014): The Relevance of Algorithms. In: Pablo J. Boczkowski, Kirsten A. Foot & Ders. (Hgs.), Media Technologies. Essays on Communication, Materiality, and Society, Cambridge: The MIT Press, 167–194. https://doi.org/10.7551/mitpress/9780262525374.003.0009.

Gusterson, Hugh (1997): Studying Up Revisited. In: Legal Anthropology Review 20/1, 114–119. https://doi.org/10.1163/25891715-bja10028.

Introna, Lucas D. (2016): Algorithms, Governance, and Governmentality. In: Science, Technology, & Human Values 41/1, 17–49. https://doi.org/10.1177/0162243915587360.

Introna, Lucas D. (2017): Die algorithmische Choreographie des beeindruckbaren Subjekts. In: Robert Seyfert & Jonathan Roberge (Hgs.), Algorithmuskulturen. Über die rechnerische Konstruktion der Wirklichkeit. Bielefeld: Transcript, 41–74. https://doi.org/10.14361/9783839438008-002.

Kitchin, Rob (2017): Thinking Critically About and Researching Algorithms. In: Information, Communication & Society 20, 14–29. https://doi.org/10.1080/1369118X.2016.1154087.

Kitchin, Rob & Tracey P. Lauriault (2018): Toward Critical Data Studies. Charting and Unpacking Data Assemblages and Their Work. In: Jim Thatcher, Andrew Shears & Josef Eckert (Hgs.), Thinking Big Data in Geography. New Regimes, New Research. Lincoln; London: University of Nebraska Press, 3–20.

Koch, Gertraud (2017): Ethnografie digitaler Infrastrukturen. In: Dies. (Hg.), Digitalisierung. Theorien und Konzepte für die empirische Kulturforschung. Konstanz, München: Herbert von Halem Verlag, 107–126.

Kropf, Jonathan (2019): Recommender Systems in der populären Musik. Kritik und Gestaltungsoptionen. In: Ders. & Stefan Laser (Hgs.), Digitale Bewertungspraktiken. Für eine Bewertungssoziologie des Digitalen. Wiesbaden: Springer VS, 127–163.

Lemke, Thomas (2001): Gouvernementalität. In: Marcus S. Kleiner (Hg.), Michel Foucault. Eine Einführung in sein Denken. Frankfurt a.M., New York: Campus, 108–122.

Light, Ben, Jean Burgess & Stefanie Duguay (2017): The Walkthrough Method. An Approach to the Study of Apps. In: new media & society 20/3, 881–900. https://doi.org/10.1177/1461444816675438.

Lipp, Benjamin (2022): Caring for Robots: How Care Comes to Matter in Human-Machine Interfacing. In: Social Studies of Science. https://doi.org/10.1177/03063127221081446.

Mackenzie, Adrian (2005): The Performativity of Code. Software and Cultures of Circulation. In: Theory, Culture & Society 22/1, 71–92. https://doi.org/10.1177/0263276405048436.

Marino, Mark C. (2018): Reading Culture through Code. In: Jentery Sayers (Hg.), The Routledge Companion to Media Studies and Digital Humanities. New York, London: Routledge, 472–482. https://doi.org/10.4324/9781315730479.

Marino, Mark C. (2020): Critical Code Studies. Cambridge: The MIT Press.

Mol, Annemarie, Ingunn Moser & Jeannette Pols (Hgs.) (2010): Care in Practice. On Tinkering in Clinics, Homes and Farms. Bielefeld: Transkript.

Mousavi Baygi, Reza, Lucas D. Introna & Lotta Hultin (2021): Everything Flows: Studying Continuous Socio-Technological Transformation in a Fluid and Dynamic Digital World. In: MIS Quarterly 45/1, 423–452. https://doi.org/10.25300/MISQ/2021/15887.

Müske, Johannes (2020): Diskurs. In: Tim Heimerdinger & Markus Tauschek (Hgs.), Kulturtheoretisch argumentieren. Ein Arbeitsbuch. Münster u.a.: UTB, 100–129.

Niewöhner, Jörg (2015): Infrastructures of Society, Anthropology of. In: James D. Wright (Hg.), International Encyclopedia of the Social & Behavioral Sciences. Volume 12, Oxford: Elsevier, 119–125.

Ntoutsi, Eirini, Pavlos Fafalios, Ujwal Gadiraju, Vasileios Iosifidis, Wolfgang Nejdl, Maria-Esther Vidal, Salvatore Ruggieri, Franco Turini, Symeon Papadopoulos, Emmanouil Krasanakis, Ioannis Kompatsiaris, Katharina Kinder-Kurlanda, Claudia Wagner, Fariba Karimi, Miriam Fernandez, Harith Alani, Bettina Berendt, Tina Kruegel, Christian Heinze, Klaus Broelemann, Gjergji Kasneci, Thanassis Tiropanis & Steffen Staab (2020): Bias in Data-Driven Artificial Intelligence Systems—An Introductory Survey. In: Wiley Interdisciplinary Reviews Data Mining and Knowledge Discovery 10/3, 1–14. https://doi.org/10.1002/widm.1356.

Mol, Annemarie (2002): The Body Multiple: Ontology in Medical Practice. Science and Cultural Theory. Durham: Duke University Press. https://doi.org/10.1215/9780822384151.

Montfort, Nick, Patsy Baudoin, John Bell, Ian Bogost, Jeremy Douglass, Mark C. Marino, Michael Mateas, Casey Reas, Mark Sample, and Noah Vawter (2013): 10 PRINT CHR$(205.5+RND(1));:GOTO 10. Software Studies. Cambridge, Mass: MIT Press.

Nake, Frieder (2008): Surface, Interface, Subface: Three Cases of Interaction and One Concept. In: Uwe Seifert, Jin Hyun Kim & Anthony Moore (Hgs.), Paradoxes of Interactivity. Bielefeld: Transcript. https://doi.org/10.14361/9783839408421-005.

Ochs, Carsten & Barbara Büttner (2019): Selbstbestimmte Selbst-Bestimmung? Wie digitale Subjektivierungspraktiken objektivierte Datensubjekte hervorbringen. In: Carsten Ochs, Michael Friedewald & Thomas Hess (Hgs.), Die Zukunft der Datenökonomie. Zwischen Geschäftsmodell, Kollektivgut und Verbraucherschutz. Wiesbaden: VS Verlag, 181–214.

Peetz, Thorsten (2021): Digitalisierte intime Bewertung. Möglichkeiten sozialer Beobachtung auf Tinder. In: Kölner Zeitschrift für Soziologie und Sozialpsychologie 73, 425–450.

Pressman, Jessica, Mark C. Marino & Jeremy Douglass (2015): Reading Project: A Collaborative Analysis of William Poundstone's Project for Tachistoscope (Bottomless Pit). University of Iowa Press. https://doi.org/10.2307/j.ctt20p598m.

Reichert, Ramón & Annika Richterich (2015): Introduction. Digital Materialism. In: Digital Culture & Society 1, 5–17. https://doi.org/10.25969/mediarep/634.

Reigeluth, Tyler Butler (2014): Why Data is not Enough. Digital Traces as Control of Self and Self-Control. In: Surveillance & Society 12/2, 243–254. https://doi.org/10.24908/ss.v12i2.4741.

Rotthaus, Hannah (2020): Aushandlungen von Schwangerschaftsverhütung im Kontext digitaler Selbstbeobachtung. In: Hamburger Journal für Kulturanthropologie 11, 3–93.

Seaver, Nick (2017): Algorithms as Culture: Some Tactics for the Ethnography of Algorithmic Systems. In: Big Data & Society, 4/2. https://doi.org/10.1177/2053951717738104.

Shah, Nishant (2019): Interface as a Mediating Technology of Organization. In: Timon Beyes, Claus Pias & Robin Holt (Hgs.), Oxford Handbook of Media, Technology, and Organization Studies. London: Oxford University Press, 257–264.

Star, Susan Leigh (1999): The Ethnography of Infrastructure. In: American Behavioral Scientist 43, 377–391. https://doi.org/10.1177/00027649921955326.

Star, Susan Leigh & Karen Ruhleder (1996): Steps Toward an Ecology of Infrastructure. Design and Access for Large Information Spaces. In: Information Systems Research 7/1, 111–134. https://doi.org/10.1287/isre.7.1.111.

Suchman, Lucy (2007): Human-Machine Reconfigurations. Plans and Situated Actions. Cambridge u.a.: Cambridge University Press. https://doi.org/10.1017/CBO9780511808418.

Thanner, Sarah & Libuše Hannah Vepřek (2023): Imaginieren – Intraagieren – Rekonfigurieren: Mensch-Technologie-Relationen im Werden. In: Manuel Trummer, Daniel Drascek, Gunther Hirschfelder, Lena Möller, Markus Tauschek & Claus-Marco Dieterich (Hgs.), Zeit. Zur Temporalität von Kultur. 43. Kongress der Deutschen Gesellschaft für Empirische Kulturwissenschaft (DGEKW). Münster: Waxmann, 321–338.

Tischberger, Roman (2020): Computer sagt Nein. Fehlerkulturen in der Softwarearbeit. In: Stefan Groth, Sara May & Johannes Müske (Hgs.), Vernetzt, Entgrenzt, Prekär? Arbeit im Wandel und in gesellschaftlicher Diskussion – kulturwissenschaftliche Perspektiven. Frankfurt a.M., New York: Campus, 113–134.

Tischberger, Roman (2021): Wie wir coden wollen. Zum strategischen Umgang mit der Ressource Wissen in der Softwarearbeit. In: Hamburger Journal für Kulturanthropologie 13, 139–149.

Trischler, Ronja (2022): Digitale Materialität. Eine Ethnografie arbeitsteiliger Visual-Effects-Produktion. Bielefeld: Transkript. https://doi.org/10.1515/9783839457962.

Vepřek, Libuše Hannah (2018): Programmierte (Un)Gleichheiten. Mit Data Mining in die Polizeiarbeit der Zukunft?! In: Kuckuck. Notizen zur Alltagskultur 2, 18–24.

Vepřek, Libuše Hannah (2023): At the Edge of Artificial Intelligence. Intraversions in Human Computation Systems. Unv. Dissertation, LMU München.

Weber, Jutta (2018): Pleasing Little Sister. Big Data und Social Media Surveillance. In: Thorben Mämecke, Jan-Hendrik Passoth & Josef Wehner (Hgs.), Bedeutende Daten. Modelle, Verfahren und Praxis der Vermessung und Verdatung im Netz. Wiesbaden: Springer VS, 91–104.

Werner, Ann (2020): Organizing Music, Organizing Gender. Algorithmic Culture and Spotify Recommendations. In: Popular Communication. The International Journal of Media and Culture 18, 78–90. https://doi.org/10.1080/15405702.2020.1715980.

Wiedemann, Lisa (2019): Self-Tracking. Vermessungspraktiken im Kontext von Quantified Self und Diabetes. Wiesbaden: Springer VS.

Autor:inneninformation

Libuše Hannah Vepřek studierte Empirische Kulturwissenschaft sowie Informatik in Regensburg und München. Sie promoviert am Institut für Empirische Kulturwissenschaft und Europäische Ethnologie der LMU München und ist wissenschaftliche Mitarbeiterin im DFG-Projekt „Spielend in the Loop: Neue Mensch-Software Relationen in Human Computation Systemen und deren Auswirkungen auf Sphären des Alltags“. Zu ihren Forschungsinteressen zählen unter anderem die digitale und Technikanthropologie, Science and Technology Studies, Moralanthropologie, Critical Code Studies, Digitale Methoden und Erinnerungskulturen.

Sarah Thanner studierte Empirische Kulturwissenschaft und Linguistik in Regensburg und war von 2019-2023 wissenschaftliche Mitarbeiterin im BMBF-Verbundprojekt VIGITIA am Lehrstuhl für Medieninformatik der Universität Regensburg. In ihrem an der FSU Jena angesiedelten Promotionsprojekt erforscht sie das Werden von Mensch-Technologie-Relationen im Kontext der Entwicklung „smarter“ Alltagsdinge und Augmented Reality. Ihre Forschungsinteressen umfassen unter anderem Fragestellungen der digitalen Anthropologie und der Science and Technology Studies sowie partizipative, multimodale oder kollaborative Methoden und Forschungszusammenhänge.

Lina Franken, Dr. phil., ist Professorin für Digital Humanities in den Kulturwissenschaften an der Universität Vechta. Nach ihrem Studium der Volkskunde, neueren Geschichte und Medienwissenschaft in Bonn promovierte sie in der Vergleichenden Kulturwissenschaft Regensburg. Forschungsschwerpunkte sind: Methodologie und digitale Methodenentwicklung, Technisierung und Digitalisierung in Alltag und Wissenschaft, Bildungskulturen und -politik, Immaterielles Kulturerbe, Arbeits- und Nahrungskulturen.