Volume 12 Number 3 April 2024

Digital Journalism

DIGITAL JOURNALISM

Routledge Taylor & Francis Group

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/rdij20

Analyzing Code: What a Critical Code Studies Approach Reveals About the Epistemology of Data Journalism

Aman Abhishek & Lucas Graves

To cite this article: Aman Abhishek & Lucas Graves (02 May 2024): Analyzing Code: What a Critical Code Studies Approach Reveals About the Epistemology of Data Journalism, Digital Journalism, DOI: 10.1080/21670811.2024.2345205

To link to this article: https://doi.org/10.1080/21670811.2024.2345205

Published online: 02 May 2024.

DIGITAL JOURNALISM
https://doi.org/10.1080/21670811.2024.2345205

Routledge
Taylor & Francis Group

RESEARCH ARTICLE

Analyzing Code: What a Critical Code Studies Approach Reveals About the Epistemology of Data Journalism

Aman Abhishek and Lucas Graves

School of Journalism and Mass Communication, University of WI-Madison, Madison, WI, USA

ABSTRACT

The last decade has seen consolidation of data journalism—defined broadly as reporting involving the collection, analysis, and presentation of quantitative datasets—as an established subfield of journalism. While programming code underlies much of data journalism, few studies focus on it, and almost none examine the code itself. Informed by Critical Code Studies, this study departs from previous efforts by examining the role of *both* quantitative data and computational analysis in data-driven reporting. Using a sample of 234 GitHub repositories of published news stories in the US, we conduct a content analysis of three separate elements of each story: code, data, and the published article. We then use hierarchical clustering to triangulate across these characteristics and identify groups of stories for qualitative review. Our analysis demonstrates the breadth of data journalism—from major investigations to quick statistical reports—enabled by the combination of abundant data and computational techniques to collect and process that data. We highlight the continued dependence on government data, but also the labor involved in scraping web data, and the use of sophisticated reverse-engineering methods to scrutinize tech corporations. Finally, we make several observations regarding studying GitHub as a data journalism infrastructure.

KEYWORDS

Data journalism; code; open source; GitHub; investigative journalism; algorithmic accountability; methodology

Introduction

An increasing number of news organizations have been using platforms like GitHub to publish programming code and datasets used in their reporting, serving as an appendix for the published news story (Haim and Zamith 2019). In this article, we construct a sample of 234 GitHub repositories and the published news stories accompanying them. Informed by Critical Code Studies (Marino 2006, 2020), we carry out a content analysis of each repository and the published news report to explore editorial facets of code and data—meaning *the role code and data play in constituting a news report*. As Kitchin (2017) argues, to study code, we need to unpack the wider sociotechnical assemblages in which it is embedded to understand how programs "work in the world". In this paper, we aim to understand the computational methods

CONTACT Aman Abhishek aabhishek@wisc.edu

that journalists are using to assemble and analyze datasets, and how this work relates to the published news report. To our knowledge this is the first study to analyze journalism code at this scale.

Literature Review

The last decade has seen a pronounced "quantitative turn" in professional journalism (Coddington 2015), characterized by the increasing prominence of reporting that centers on the collection, analysis, and presentation of quantitative data sets. This shift has precursors dating to the 1970s and 1980s, when newsrooms began to integrate computers, databases, and social scientific methods into reporting work (Anderson 2018). The current quantitative turn in news reflects wider social changes including the shift to digital distribution across media industries; the advent of "open source" software production and culture; and, crucially, the availability of new kinds and vast quantities of data as a byproduct of the embrace of digital communication and sociality—the "datafication" of social life (Baack 2015; Loosen, Reimer, and De Silva-Schmidt 2020).

A wealth of recent scholarship addresses the rise of "data-driven" reporting and storytelling, which comprises actual news practices and products but also an emerging set of professional discourses (Anderson 2018; De Maeyer et al. 2015). At least three strands stand out in this literature. One strand offers conceptual maps or taxonomies of the emerging landscape (Borges-Rey 2020; Coddington 2015), which attempt to define and distinguish labels such as data journalism, computational journalism, and the older computer-assisted reporting (CAR). While details vary, a key axis in many of these maps differentiates between journalist-led and programmer-led arenas—what Borges-Rey (2020) contrasts as "newshound" and "techie" approaches.

Another group of studies approaches the landscape of data journalism explicitly as a site where different epistemologies or discourses intersect, and thus where journalists may absorb values and practices from the programming, open-source and open-data communities (Baack 2018; Boyles 2020; De Maeyer et al. 2015; Haim and Zamith 2019; Parasie and Dagiral 2012; Usher 2016). Finally, a third strand of research focuses on evaluating or typologizing the actual stories produced under the rubric of data journalism, typically with content analysis (Stalph 2018; Zamith 2019). Notably, with very few exceptions (Weber and Kosterich 2018), large-scale content analyses have not examined the actual code produced by news organizations.

This study addresses that gap in research on actually existing data journalism, asking whether and how data-driven reporting projects rely on programming code to develop the story. Drawing on a sample of news-linked repositories on GitHub (repositories associated with a published news report), we analyze both code and data features of published projects in a way that takes seriously the editorial roles of programming—the idea that, extending the argument made by Parasie and Dagiral (2012, p. 862), coding becomes "a legitimate dimension of newsmaking" (see also Anderson 2018; Usher 2016). That is, our content analysis of programming code aims to help flesh out the broad consensus among scholars of data journalism that its rise has enabled "people with computer skills to claim a more active role in the production of editorial content" (Parasie 2022, 21).

Studying Data Journalism in Practice

Empirical studies of quantitative and/or computational reporting increasingly focus on "data journalism" as the "central term for this area of journalistic activity" (Coddington 2018). However, practitioners themselves understand and do data journalism in different ways, across occupational lines—reporters, programmers, data specialists, etc.—but also depending on the news organization, area of coverage, and so on (Coddington 2018; Stalph 2018). As a "socio-discursive practice," De Maeyer et al. (2015) argue, data journalism is constituted in an ongoing way by discourse among journalists but also professional organizations, open data advocates, foundations, researchers, and other engaged actors.

This ambiguity forces difficult choices on empirical, "bottom up" studies of data journalism. Many analyses of content, discourse, or practice in data journalism have focused on the uppermost tier of the field: award entries, projects or teams of leading newsrooms, high-profile hackathons, etc. This results in an unrepresentative picture that fails to capture "general" (Uskali and Kuutti 2015) or "everyday" (Loosen, Reimer, and De Silva-Schmidt 2020) or "quick turnaround" (Borges-Rey 2016) data journalism that typically falls short of ideals espoused by data journalism evangelists (Zamith 2019).

However, widening the focus distorts the picture in a different way, because some actors and projects matter more than others in defining data journalism, as underscored by both field (Baack 2018; Lowrey, Sherrill, and Broussard 2019) and actor-network (De Maeyer et al. 2015; Parasie 2022) perspectives on the phenomenon. This results from the field-defining status enjoyed by award-winning projects, for example, but also because major efforts embody data journalism more fully: they tend to use the tools and face the issues that make the subfield distinct. In the same way, investigative journalism can be defined more or less broadly, but routine investigations are less likely to involve the resources, special training, legal protections, and other features that make investigative journalism a separate subfield.

This empirical analysis seeks to develop a picture of how code and data combine in data-driven reporting, rather than to characterize the field of data journalism broadly. As detailed below, we develop a non-representative sample designed deliberately to surface stories which rely on code to create and/or analyze data sets. At the same time, our initial sample (participants in a major conference) aims to preserve diversity of projects and organizations. In essence, we want to get a "bottom up" understanding of data journalism by studying how datasets are assembled and analyzed (its practice) without focusing exclusively on award-winning projects or top-tier newsrooms.

Taking Code Seriously

To investigate editorial facets of newsroom programming we draw on Critical Code Studies (CCS), a humanistic approach grounded in the view that "we can read and explicate code the way we might explicate a work of literature" (Marino 2006). The central tenet of CCS is that code must be analyzed in social, material, and historical context (in the same way that Critical Legal Studies rejects narrow legalistic readings of law in favor of critical, historically situated analysis). The need for a contextual

approach has arguably been clearest in studying algorithms, whose operations and impact depend on being "embedded within complex socio-technical assemblages" including hardware, data, and human behavior (Kitchin 2017). CCS argues that every piece of code is a "social text" whose meaning reflects the circumstances of its production—who created it, under what conditions, for what purpose—but also "develops and transforms as additional readers encounter it over time and as contexts change" (Marino 2020).

In terms of practical methodology, CCS suggests that interpreting code contextually requires considering the code in relation to other texts that define it, such as documentation, comments, program structure, and other ancillary material. Our focus on news-linked GitHub repositories embodies this contextualist, intertextual approach. Specifically, as detailed in the next section, we consider three interrelated texts—the code (and associated documentation), the data, and the published story—in order to highlight how programming code serves editorial purposes. We use "editorial" here in the widely accepted sense of the line between newsroom and business functions—as in "editorial routines," "editorial judgment," "editorial analytics," etc. (e.g. Parasie 2022; Petre 2021)—where the "editorial side" encompasses gathering information, analyzing it, and producing compelling stories from it in a given medium, but not for instance distribution operations or advertising sales. CCS's contextualist approach thus aligns perfectly with an emphasis on editorial facets of programming code.

Some previous research has studied GitHub repositories as context for understanding data journalism (e.g. Boyles 2020; Haim and Zamith 2019; Weber and Kosterich 2018). Notably, by analyzing listed "project objectives" on GitHub, Haim and Zamith (2019) find the bulk of news-related repositories focused on general tools for news distribution, production, or consumption. (While that study does not highlight repositories linked to published stories, 15% of those analyzed were used to release "news production materials like copies of data sets.") Weber and Kosterich (2018) analyze programming code itself, rather than descriptions of repositories, but focus on news distribution apps unrelated to reporting work.

To our knowledge, no previous research has focused on the potential *editorial* dimensions of programming code in gathering, processing or analyzing the datasets at the center of published news stories. Two research questions guide our investigation:

RQ1: How do the uses of code and data relate to the claims in published news reports?

RQ2: What does CCS's contextual approach (analyzing code, data, repository comments, and the published story together) reveal about the methods data journalists are using to assemble, analyze, maintain, and share data?

Methodology

This analysis focuses exclusively on cases where a published news story is accompanied by a GitHub repository containing relevant data and/or code (what we call "news-linked repositories"). In keeping with Critical Code Studies, we consider these elements together in order to analyze the code in light of the specific journalistic work it supports. That is, we interpret code contextually by considering it in relation

Table 1. Summary of the variables (sample size = 234).

Variable name	Description	Mean	Sum
Analyzing Data			
FOIA	1 if any data was obtained through a FOIA request 0 otherwise.	0.15	34
Leaked data	1 if any leaked data was used (governmental or non-governmental source) 0 otherwise.	0.00	0
Government data	1 if any government data was used 0 otherwise.	0.64	150
Data present	1 if the repository contains any data 0 otherwise.	0.91	214
Data featured	1 if the news story highlighted the creation/sharing of data 0 otherwise.	0.38	88
Analyzing Code			
Scraping	1 if the code is used to access multiple datasets iteratively 0 otherwise; NA if no code is shared.	0.15	27
Analysis	1 if the code is used for statistical analysis or mathematical manipulation 0 otherwise; NA if no code is shared.	0.94	173
Quantification	1 if the code converts qualitative information (such as text) to numerical data 0 otherwise; NA if no code is shared.	0.05	9

to other texts that define it—specifically documentation, comments, program structure, published news reports, and data in the repository. Table 1 offers an overview of how we classified data and code with reference to the published story, detailed below under "content analysis."

A quick note on GitHub terminology: each user's or organization's GitHub account has a list of coding projects called "repositories" (Figure 1). A repository contains some combination of code, data and documentation (Figure 2).

Sampling

We proceeded in three steps to create a sample of data journalism stories with an accompanying GitHub repository. Our final sample ranges from Pulitzer Prize-winning investigations to a 150-word story visualizing data on bike rental rides.

Sampling news organizations: We began with the list of 101 institutions which participated in NICAR 2022, a data journalism conference organized by the US-based Investigative Reporters and Editors (IRE). This approach was intended to create a more diverse set than sampling from award-winning projects (Loosen, Reimer, and De Silva-Schmidt 2020) or major news organizations (Zamith 2019).

Identifying GitHub accounts: We searched for the GitHub account of each news organization using the GitHub API. From the results, we removed irrelevant GitHub accounts, accounts without a data journalism repository, and accounts without a weblink to the news organization (Figure 1), yielding a total of 52 GitHub accounts owned by news organizations. Figure 1 illustrates the account information we accessed using the GitHub API.

Identifying GitHub repositories: We imposed two conditions to identify relevant repositories within these 52 accounts: 1) The repository must contain a weblink to the published news story[1] and not have an empty "about" field (this resulted in 1609 repositories). 2) The repository's "about" field must contain relevant terms such as "data," "story," "methodology," or "analysis" (this reduced the sample to 457 repositories) (see Appendix A for

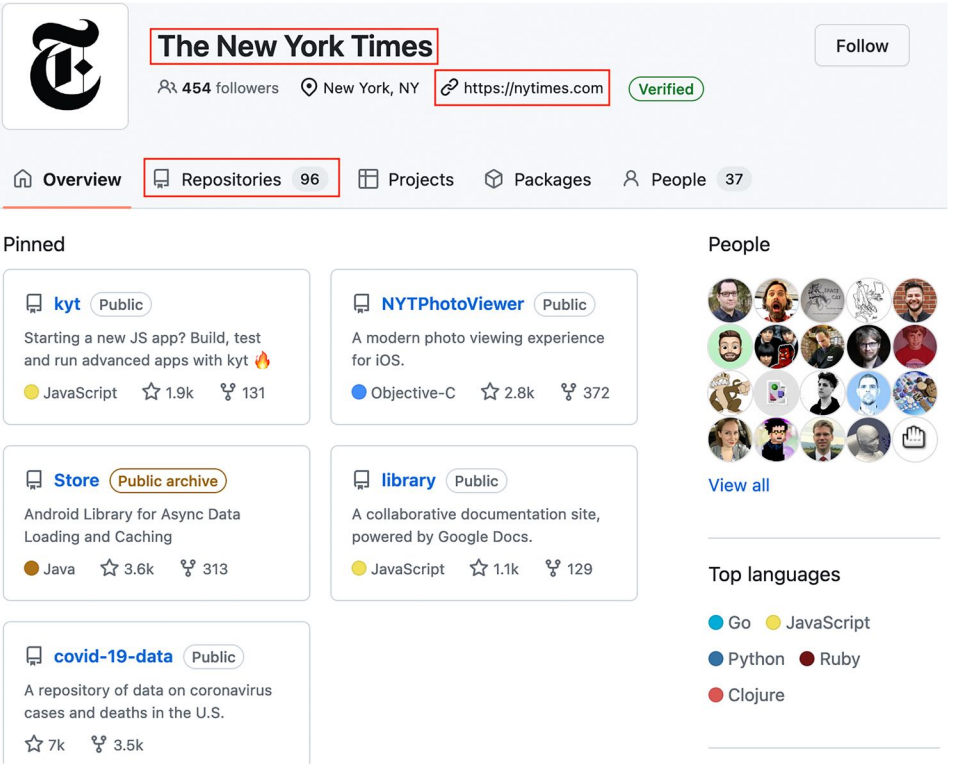

Figure 1. The GitHub account of *The New York Times*, containing weblink to the *The Times* website, a description field, list of repositories and additional information. Accessed on June 14, 2023.

details). Figure 2 illustrates these fields, which we accessed through the GitHub API. Two project assistants then manually reviewed repositories and removed remaining irrelevant cases, producing a final dataset of 234 repositories from 23 news organizations.

Content analysis

Two project assistants conducted a content analysis of the code, documentation (README files, comments within code etc.) and data within the GitHub repositories, as well as the published news reports using a "classification schema" (we are not calling this "coding sheet" to avoid multiple meanings of the term "code"). That is, we defined variables to capture various features of (1) data in the repository, (2) code in the repository, and (3) the published news story (see Table 1). Below, we describe how these variables were defined and recorded during the content analysis, and then explain how hierarchical clustering of these variables assisted our qualitative analysis.

Data

To characterize the use of data in our sample we consider its source, how it was obtained, and how it is used in the story. We classify data with respect to the repository as well as the published story.

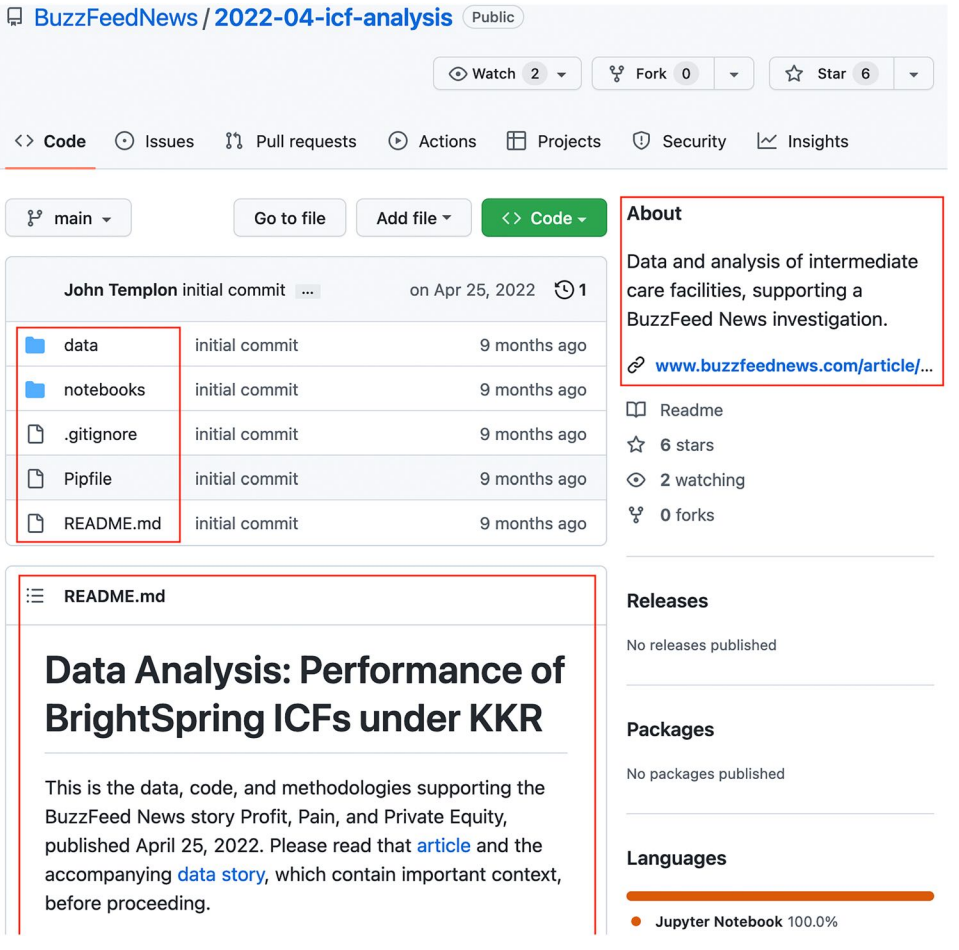

Figure 2. *BuzzFeed News'* repository for the news story "Profit, Pain, and Private Equity." the "about" field contains a brief project description as well as a weblink to the news story. The code is in the "notebooks" folder. The documentation is usually in the README.md file, but it can also appear as comments within code. Accessed on Feb 5, 2023.

Previous studies of data journalism have found widespread and heavy reliance on government data (Loosen, Reimer, and De Silva-Schmidt 2020; Stalph 2018; Zamith 2019). This reflects professional journalism's systematic privileging of official voices and frames (Gans 2004) but also the particular ability to quantify reality and create statistical data as a facet of state and institutional power (Porter 1995; Scott 1998). Journalists' reliance on government data has been understood as an "information subsidy" that both enables and influences news coverage (Gandy 1982). Meanwhile, studies of data journalism have paid less attention to the work required to obtain government data or render it usable. Government data may be released in response to Freedom of Information (FOIA) requests, or leaked. Datasets may also be scattered across disparate files or released in file formats inconvenient for data analysis, perhaps intentionally (e.g., tabular data in a PDF format).

Project assistants recorded the dummy variables *FOIA* (Krippendorff's alpha = 1.0), *leaked data* (Krippendorff's alpha = 1.0) and *government data* (Krippendorff's alpha = 1.0) in our

classification schema based on reading the news report and the documentation in the repository, and searching for phrases like "open data," "FOIA," "leak" etc. These variables represent non-exclusive categories (e.g., *leaked data* may also be *government data*).

We also noted the availability of journalistic datasets in the GitHub repository, independently of the data's source. Established practice in data journalism emphasizes making available the datasets that underlie analysis; scholars understand this as reflecting norms of transparency as well as civic engagement, where providing data may "facilitate various forms of interactivity, engagement, participation and collaboration" (Gray and Bounegru 2019). As Coddington (2015) puts it, data journalism can "allow the public to analyze and draw understanding from data themselves, with the data journalist's role being to access and present the data on the public's behalf".

Project assistants marked the variable *data present* as "1" if the repository included at least one dataset (Krippendorff's alpha = 1.0). Locating data files in a repository is usually straightforward due to their distinct file extensions like "csv" or "pdf". We did not track the completeness of data in the repository (whether it allowed the entire analysis to be replicated) because it was unfeasible to do so.

Code

Our variables used to classify programming code focus on how code was used to gather, create, or analyze data. This reflects data journalism practice and the fact that our sample only consists of news-linked repositories (and does not include, for example, code for mobile apps).

Scraping: Our first code-related variable highlights the labor of accessing and working with unstructured data from government or other sources, rarely highlighted by scholarship on computational and data journalism. This labor includes, e.g., assembling data scattered across different sites or sources, checking for missing or incorrectly recorded data, and converting data into formats compatible with analysis tools. These steps feature prominently in data journalism textbooks and tutorials, referred to as "data scraping," "data wrangling," "preprocessing," "data cleaning" and similar terms (Houston 2019; Nguyen 2010).

Importantly, for this analysis scraping is understood as a technique not just to access but to create datasets, a distinction which becomes blurry in practice. For example, *The New York Times'* award-winning COVID-19 dataset relies almost exclusively on health data which was publicly available from various agencies—but which gained tremendous value when systematically cleaned and collated by *The Times*. In other cases, scraping specific items from a large number of sources can be used to create a new dataset, a technique used for instance in social media analysis.

The dummy variable *scraping* (Krippendorff's alpha = 0.81) captures instances where code is used to automatically collect and collate data or datasets from one or more web pages. Examples range from code which iterates over web addresses to collect specific data (e.g. people's names) from each, to downloading scattered datasets (Figure 7). *Scraping* was marked as "1" if the repository contained such scraping code ("0" otherwise).

Quantification and *analysis*: Two additional code-related variables highlight challenges that Porter (1995), in his work on quantification and objectivity, distinguishes as "the contact between the numbers and the world" (here, *quantification*) and "the methods

of processing and analyzing numerical information" (or *analysis*). Our initial review of programming code in news-linked repositories indicated that such code may be used to generate quantitative data from qualitative observations (*quantification*) or to conduct mathematical operations on quantitative data once assembled (*analysis*).

The dummy variable *analysis* was marked as "1" if at least some code in the repository included statistical analysis or other mathematical manipulation (Krippendorff's alpha = 1.0). *Analysis* is independent of other code variables – independent of whether the dataset was scraped or created by journalists. The variable *quantification* (Krippendorff's alpha = 0.76) refers narrowly to cases where the repository includes code which converts textual data to numeric data, for example, measuring "negative sentiment" in a text (see Figure 8).

News Story

Content analysis of each repository in our sample included a close reading of the linked news story, which informed code- and data-related variables. For instance, the GitHub repository as well as the published story was used to ascertain whether government data and/or leaked data was used. One dummy variable, *data featured* (Krippendorff's alpha = 0.76), refers specifically to whether the published article highlights that journalists are creating and sharing a dataset. This is distinct from the previous variable *data present* which noted whether the repository contains a dataset; this may or may not be highlighted in the news article. In keeping with recent scholarship, *data featured* helps to identify cases where the news organization highlights the creation and publication of datasets as public-facing newswork in itself (Coddington 2015), and could potentially mobilize "data publics" (Gray and Bounegru 2019) which put that data to new uses. *Data featured* greatly overlaps with *data present*, but the latter also includes cases where data is shared via the repository—for transparency and potentially replication—but not in the published story.

Hierarchical Clustering and Qualitative Analysis

We used the variables in Table 1 for hierarchical clustering to assist the subsequent qualitative analysis informed by CCS. The clustering embodies CCS because it places the characteristics of the programming code directly in relation to the characteristics of other primary "texts" accompanying it – the data and published news story. The resulting clusters do *not* represent a definitive typology of data journalism. Since our variables are not completely independent of one another (e.g. *data present* and *data featured*), the clusters are also not conceptually orthogonal or mutually exclusive. Rather, the clustering helped us to identify groups of repositories which were similar across the variables we have defined, as a starting point for qualitative analysis. Further, there were no instances of leaked data in our sample, so we excluded the variable *leaked* in clustering. We also excluded the variable *FOIA* – but not from the discussion which follows – because it created many redundant subclusters. The details of clustering are in Appendix B.

We undertook a close re-reading of the news reports and the repositories within each cluster to identify common themes, which we present below. The clusters are

Table 2. Summary statistics of the clusters. The values are the averages of the dummy variable for the repositories within that cluster. Superscripts in the first column indicate the two cluster pairs which were merged.

		Function of the code in the repository			Details of data in the repository		
Cluster No.	Reference name	(1) Scraping	(2) Quantification	(3) Analysis	(4) Data present	(5) Government data	(6) Data featured
1	Data Archival	0.00	0.00	0.00	1.00	1.00	1.00
2*	Analyzing Government Data	0.00	0.00	1.00	1.00	1.00	0.00
3*		0.00	0.00	1.00	1.00	1.00	1.00
4	Data Wrangling	1.00	0.00	0.84	0.95	1.00	0.42
5°	Tech Reporting	0.00	0.00	0.00	1.00	0.00	1.00
6°		0.33	0.20	1.00	1.00	0.00	1.00
7	Social Analysis	0.00	0.20	1.00	0.97	0.07	0.03

summarized in Table 2; the cluster labels are a shorthand for referring to the key insights gleaned from them.

Findings and Analysis

Theme 1: Creating Databases as an End Goal of Data Journalism

Our analysis found that simply sharing datasets is a primary use of GitHub by data journalists: repositories in the sizable "data archival" and "tech reporting" clusters include datasets but no programming code (Table 2, columns 1-3). Importantly, projects in these clusters may have relied on code which journalists chose not to share—indeed, we find this is most often the case (see examples below). The clear implication is that sharing datasets via GitHub was considered a worthy goal in itself, as further reflected in the high value of *data featured* in these clusters (Table 2, column 6) indicating discussion of dataset creation in the published story.

As noted previously, this finding agrees with existing work on journalistic uses of GitHub (Haim and Zamith 2019) and on wider norms of data journalism, whereby journalists aim to "allow the public to analyze and draw understanding from data themselves" (Coddington 2015) and publish datasets "in order to facilitate various forms of interactivity, engagement, participation and collaboration" (Gray and Bounegru 2019). Below, we offer examples from the data archival cluster which help to substantiate this conclusion; we will discuss the tech reporting cluster later (the two differ in sources of data; see Table 2 column 5).

Most stories in the data archival cluster appear to rely on programming code not shared on GitHub. For example, *The Washington Post*'s project "Fatal Force" has been tracking fatal police shootings in the US since 2014, winning Pulitzer and Peabody awards. *The Post* has shared this dataset of killings on GitHub, and highlighted it in its news reports. The dataset is created by "scrubbing the web" for relevant news reports; a GitHub bot account updates the data regularly, but the scraping code itself is not shared. (This could reflect a desire to maintain a competitive advantage, or to ensure that data is assembled responsibly; it may also be that the project relies on access to proprietary news databases.) While code-sharing transparency is lacking, the web interface of "Fatal Force" allows readers to get the news snippet corresponding to each fatal shooting datapoint.

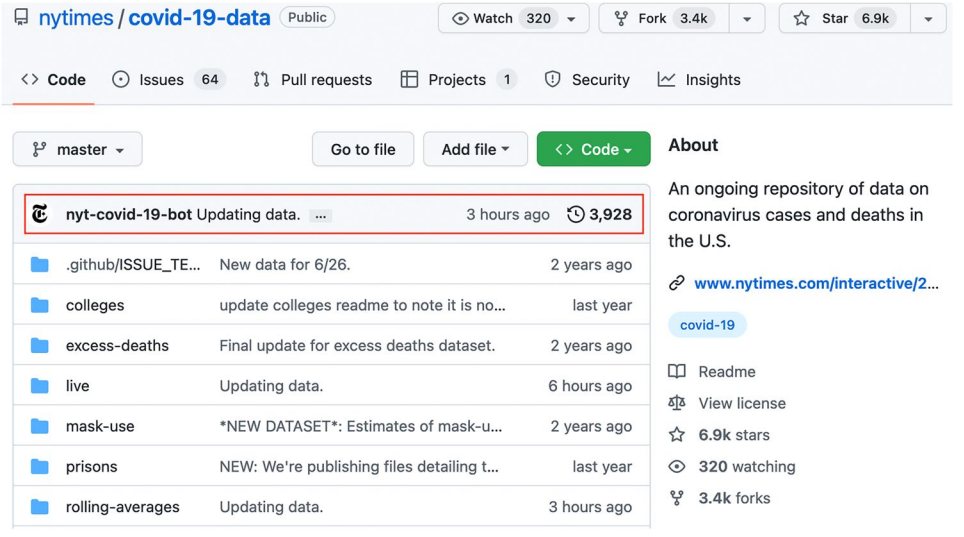

Figure 3. *The New York Times* Repository containing its COVID-19 dataset. The folders only contain data and no code. The repository has been updated 3,928 times in total, mostly by the bot (including the most recent update). Accessed on Feb 5, 2023.

Another example is *The New York Times* dataset on COVID-19 cases and deaths, which was shared on GitHub and updated daily by a bot account (Figure 3). Here too, the code which automatically compiles data from different sources was unavailable. The value of *The Times'* data archival work was widely recognized; its pandemic reporting won a Pulitzer Prize for filling "a data vacuum that helped local governments, healthcare providers, businesses and individuals to be better prepared and protected" (Pulitzer.org 2021).

A small subset of stories in the data archival category appear to not rely on programming code, using manual analysis rather than computation (or perhaps a combination of the two) to generate datasets. For example, in the investigative series "Haiti's Debt,"[2] *The New York Times'* used historical archives and secondary sources to track the reparations that Haiti was forced to pay to its French enslavers over the years. This investigation shared a tabular dataset on GitHub and also received international attention. Similarly, *The Washington Post*'s high-profile story on slave-owning legislators in US history[3] used census records and historical documents to compile a list of these congressmen, soliciting readers' help with 677 inconclusive cases. More than a hundred entries were updated in the dataset based on crowdsourced inputs. As scholars have noted, "databases can be used as crowdsourcing mechanisms to enrich existing data" (Gray and Bounegru 2019), although this method often proves unsuccessful (Borges-Rey 2020, 925–926).

In this way, a CCS-informed reading of news reports and their associated GitHub repositories (RQ2) reaffirms and adds nuance to the conclusion that creating and sharing databases becomes an end goal in itself in data journalism. In each of the examples cited here, underlying datasets were released on GitHub *in addition* to appearing via interactive charts and/or tables in the published stories. The data-sharing ethic is unmistakable when the GitHub architecture is used to release only datasets, without any scraping or analysis code. Some of these examples complicate the usual

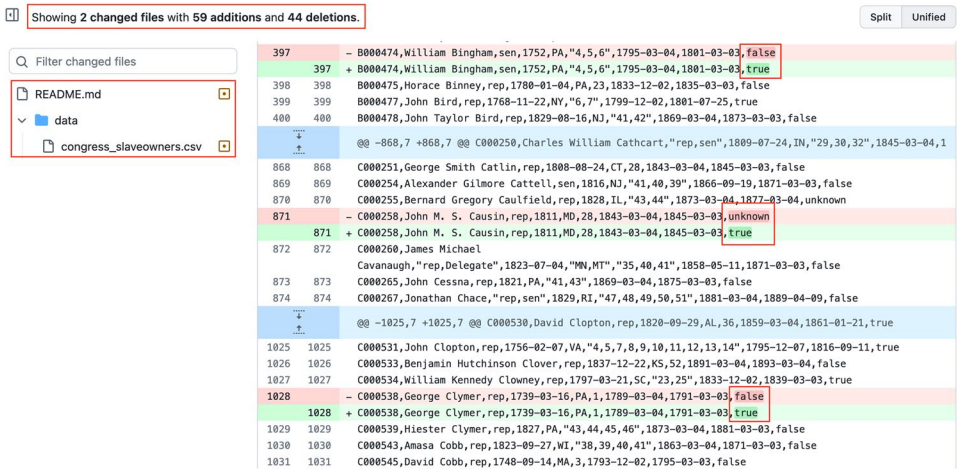

Figure 4. Changes made to *The Washington Post*'s dataset on slaveowners based on inputs from the readers.[21] Red indicates text deletion; green indicates addition. The labels "true," "false" and "unknown" indicate whether a Congressman was a slave owner. This repository contents and organization can be seen on the left.

distinction between "data" and "computation," with hundreds of hours of manual labor by journalists to create datasets (e.g. from historical documents) which are then showcased as end products of the investigation. Regardless of dataset replicability, using GitHub to share datasets fulfills key editorial functions in data journalism. First and foremost, publishing underlying data bolsters the claims made in the news report (RQ1), as anyone can "see it for themselves". In some cases, the dataset becomes the stepping stone for other reportage and research (e.g. *The Times'* COVID-19 dataset).

It should be emphasized that using GitHub offers specific advantages over other platforms for sharing datasets (such as the publisher's site or a conventional file-hosting service like DropBox). First, GitHub provides version tracking, and therefore fosters transparency regarding how crowdsourcing was used to update/modify the dataset (Figure 4). Second, GitHub permits automated updates to the dataset (Figure 3), as in the police shooting and COVID-19 data. Third, GitHub brings together a programming community and provides publicly visible engagement metrics, such as "star," "fork" and "watch"; high engagement can be a marker of prestige. In fact, a news report containing a weblink to GitHub might suggest to readers that the news organization is transparent and embraces an open source ethos—which is a performative aspect of transparency. As Boyles (2020) notes, "GitHub accounts may operate similar to newsprint mastheads, branding the outlet's prestige" in open source software spaces.

Theme 2: Government data as an Anchor of Investigative Data Journalism

We find a particular and highly consistent epistemological model by reviewing stories in the "analyzing government data" cluster (Table 2, columns 3 and 5). As noted, several clusters feature government data prominently. However, stories in this cluster follow a specific formula: journalists use government data (sometimes in combination with other datasets) to perform statistical analysis that reveals a systemic issue of public concern.

The authority vested in official data allows journalists to question those responsible—including public officials—while pointing to the results from the statistical analysis.

This trend has been noted in different ways by scholars of data journalism. For example, Parasie and Dagiral (2012, p. 859) write that "in this model, data have no journalistic value on their own. The reporter has to find the hidden 'story' in the data". Our cluster analysis highlights that, in addition to relying on government data, these projects tend to share both data *and* the code used to analyze it via GitHub (Table 2, columns 3-4; we consider clusters 2 and 3 together here.)

Several compelling illustrations appear in our sample. In 2018 *The Los Angeles Times* published an investigating showing that, despite public rhetoric against 'criminalizing' homeless people, "thousands are jailed for minor offenses"[4] (Figure 5). The analysis used data from the Los Angeles Police Department (LAPD); reporters then asked the LAPD to respond to the findings, and strongly challenged police officials by showing that they were contradicting the department's own data. Similarly, a 2019 *The Center for Public Integrity* story[5] used FOIA data to analyze the US government's justifications to reject 800,000 federal disaster relief loan applications, and found evidence of discrimination against the poor and immigrants. The straightforward analysis in these two cases, relying mainly on summary statistics, made it relatively uncontroversial for reporters to undercut official claims without being challenged.

More complicated analyses can potentially invite counter-claims. The case of the 4,600-word investigation published by *ProPublica* titled "Machine Bias" is a good illustration. Using FOIA requests and publicly available data from law enforcement agencies, *ProPublica* analyzed a proprietary algorithm called "COMPAS" sold by the company Northpointe. *ProPublica* found that COMPAS, which was being used in numerous state and local jurisdictions across the US, systematically exaggerated Black Americans' reoffending rate. The analysis extended far beyond simple statistical comparison (for instance, using a specialized regression called the Cox proportional hazards model

Finding: In 2011, one in 10 people arrests citywide were of homeless people; in 2016, it was 1 in 6

Use the grouped dataframe `arrest.totals` to calculate percentage of homeless arrests by year

In [44]:

```
arrest.totals %>%
    mutate(arrests_percent = arrests_number / sum(arrests_number) * 100) %>%
    filter(homeless == 1)
```

arrest_year	homeless	arrests_number	arrests_percent
2011	1	10496	10.85408
2012	1	11837	11.69271
2013	1	12237	12.47070
2014	1	12622	13.41667
2015	1	13418	15.41112
2016	1	14011	16.75797

Figure 5. A section of the analysis code from *the Los Angeles Times* repository. The code is in the gray box; the output is below it. The calculation is relatively simple: percentage arrest rate for each year.

Calculating expected deaths

To estimate expected deaths for each jurisdiction and week, we trained models on the 2015-2019 weekly deaths data, accounting for long-term demographic changes using a linear component and using a smoothing spline to account for seasonal variation. This is the same approach used by the New York Times in its accounting for excess deaths in the US through the COVID-19 pandemic

```
# data frame to hold observed and expected deaths data
mortality_working <- tibble()

# fit models and predict expected deaths
for (j in unique(mortality$jurisdiction)) {
  tmp1 <- mortality %>%
    filter(jurisdiction == j & weekendingdate < "2020-01-01")
  model <- lm(allcause ~ weekendingdate + bs(mmwrweek, knots = quantile(mmwrweek, seq(0,
1, 0.1))), data = tmp1)
  tmp2 <- mortality %>%
    filter(jurisdiction == j & weekendingdate >= "2020-01-01")
  tmp <- bind_rows(tmp1,tmp2)
  tmp3 <- as_tibble(predict(model,tmp, interval = "prediction"))
  tmp <- bind_cols(tmp,tmp3)
  mortality_working <- bind_rows(mortality_working,tmp)
}
```

Figure 6. A section of the analysis code from *BuzzFeed News* which calculates unreported deaths from a storm in Texas by predicting what would be the "normal" trend. The choice to define "model" in a relatively complex way is justified by citing the methodology used by *The New York Times*.

and a predictive statistic called the Kaplan-Meier estimator). After publication, Northpointe disputed the results by publishing its own analysis:

> "Our detailed review of how *ProPublica* conducted their analysis revealed several statistical and technical errors such as misspecified regression models, mis-defined classification terms and measures of discrimination, the incorrect interpretation and use of model errors, and more. These errors led to a false conclusion of racial bias; we do not believe the conclusions drawn are in fact supported by the data samples used in any way." (equivant 2018)

ProPublica published a rebuttal, stating that "We re-examined the data, considered the company's criticisms, and stand by our conclusions" (*ProPublica* 2016). Since the original analysis and associated data was published on GitHub, it could be replicated and scrutinized by a number of scholars and legal professionals; over 200 academic articles have cited it, with no clear consensus (Washington 2018, 135).

News organizations may take specific steps to avoid potential controversies when they undertake complex data analysis. For example, a 2019 *Los Angeles Times*' story that alleged discriminatory policing—"LAPD searches blacks and Latinos more. But they're less likely to have contraband than whites"—relied on open records requests to obtain police data, but outsourced the analysis to a data scientist at Harvard University, who performed specialized statistical modeling. *The Washington Post* used data from the US scientific agency NOAA to report on the climate crisis[6] and was awarded the Pulitzer Prize; a climate scientist provided the initial analysis "at *The Post's* request."[7] A 2021 *Buzzfeed News* story[8] used CDC data to calculate that the number of deaths in the Texas winter storm was likely 4 to 5 times greater than official estimates. Comments in the analysis code justified the statistical methodology by stating that *The New York Times*' had used the same approach to calculate excess deaths due to COVID-19 (Figure 6).

RQ1 asked how code and data relate to the claims in the news report. The examples above illustrate a consistent epistemological model in data journalism: performing statistical analysis of government data to reveal or clarify a systemic issue, and then using this evidence to question relevant public figures. This complicates the criticism that powerful institutions provide an information subsidy for data journalism—in the examples above, journalists' interpretation of data is often adversarial to the institutions which provide that data. As has been noted, access to government data can weaken the government's "interpretive monopoly" by allowing for alternative interpretations (Baack 2015). At the same time, as the "Machine Bias" example showed, using sophisticated statistical techniques may invite criticisms of methodology. When using such techniques is necessary, journalists strengthen their claim and mitigate risks by relying on external expertise—and by having published datasets in a way that allows outsiders to replicate and confirm their analysis.

Theme 3: Data wrangling as Essential Work of Data Journalism

Our analysis underscored the immense importance of data scraping and "wrangling" in data journalism. As noted, the variable *scraping* notes the presence of code designed to collect data across disparate web pages or other sources. Scraping code in the sample was frequently accompanied by code designed to convert files to more convenient formats and structure the dataset. We refer to this process of scraping, formatting and restructuring collectively as "data wrangling," a term used by practitioners themselves. Few scholars have addressed the importance of scraping and wrangling in data journalism: Borges-Rey (2016) observes that web scraping "provides a wider range of alternative sources of data," which helps in "overcoming the topical saturation of open/FOI data." Lowrey and Hou (2021) note that such methods might be "shifting some control from institutions to everyday citizens" by providing alternative avenues for original data collection.

All repositories in the "data wrangling" cluster contain some scraping code (Table 2, column 1). The length and complexity of this code reflects the effort and expertise involved in scraping data, converting it to a structured format, and standardizing and merging data scraped from different sources. Scraping code often requires regular updating because the format or location of the target webpage might change. For example, *The New York Times* explained the following about its COVID-19 database:

> "We learned early that individual scrapers needed to be strategically fragile: We wanted them to break when the source website changed or failed to respond, to alert us so that our team could figure out whether the issue was with our code or the source website's changes. In the early days of the pandemic, a few notable sites changed several times in just a couple of weeks, which meant we had to repeatedly rewrite our code." (Fehr and Williams 2021)

Our qualitative review of stories in the "data wrangling" cluster suggested two separate approaches to applying this code in data journalism. One set of cases emphasizes the creation and publication of a database, similar to the data archival cluster discussed previously. (There, scraping code also appeared to be used in many cases but was not shared in the repository.) For example, *The Los Angeles Times* maintains a repository containing the scraper code for its California COVID-19 database, which fetches data from government institutions including hospitals, schools and prisons.[9] The final database is published on *The LA Times'* website as well as on GitHub. Scraped data is often

unstructured, requiring additional labor to convert it to formats suited to computational analysis. The README file in a repository maintained by *The Seattle Times* states[10]:

> "…scraping PDFs should never—emphasis on NEVER—be a data journalist's first approach. In the ideal world, this information would have also been neatly available in a structured format, such as an API. The decision to dive into a PDF ocean should only be taken after it has been confirmed beyond doubt that is the only way to obtain the information contained in them…"

In a second set of data wrangling cases, code generates "clean," structured data primarily as an input for further analysis, rather than to publish a dataset (as reflected by the low value of *data featured* in Table 2, column 6). For example, in *The Seattle Times'* repository[11] shown in Figure 7, data wrangling code scrapes data on nursing home staffing from web pages and PDF files from multiple government sources. The code then analyzes this data to reveal staffing shortages in nursing homes, and the news story highlights the analysis results rather than the dataset itself. A similar case is *Buzzfeed News*' 2018 analysis[12] of gentrification—titled "They Played Dominoes Outside Their Apartment For Decades. Then The White People Moved In And Police Started Showing Up."—which used scraping code to compile nearly a decade of complaints from one

C. The long looping road to downloading

In [10]:

```
# As we dowload each report, we will save some of its metadata in this new df_list:
df_metadata = pd.DataFrame(columns = ['fain_id', 'nf_name', 'pdf_name',
                                      'rep_type', 'rep_date', 'url','download_record'])


# Each iteration of this first loop correspond to a single facility
for index, row in df_list.iterrows():

    print(index, '|', row['nf_name'])

    # Send a URL request to the site that contains all the links to each of the reports for a single facility.
    page = requests.get(row['reports_location'])
    soup = bs4.BeautifulSoup(page.text, 'html.parser')

    # The list of links to each of the reports is contained it a 'div' table with id='content_results'
    table = soup.find('div', {"id": "content_results"})

    # Each of the following table elements contains the URL of each PDF report
    ls_li = table.find_all('li')

    # Each iteration of this second loop corresponds to a single pdf report (from a single facility)
    for li in ls_li:
        # We are only looking for enforcement letters
        if 'enforcement' in li.find('a').contents[0].lower():

            # Obtain some metadata from the report
            pdf_url = li.find('a').get('href')
            pdf_name = pdf_url.split("/")[-1]
            rep_type = re.search('\d{2}\/\d{4}\s-\s(.*)$', li.find('a').contents[0]).groups()[0]
            rep_date = re.search('(\d{2}\/\d{4})', li.find('a').contents[0]).groups()[0]

            # Save down the pdf
            pdf = requests.get('https://fortress.wa.gov' + pdf_url)
            open(download_path + pdf_name, 'wb').write(pdf.content)

            # Save the report's metadata in 'df_metadata'
            new_record = {'fain_id':row['fain_id'],
                          'nf_name':row['nf_name'],
                          'pdf_name':pdf_name,
                          'rep_type':rep_type,
                          'rep_date':rep_date,
                          'url':'https://fortress.wa.gov' + pdf_url,
                          'download_record':time.ctime()}
            df_metadata = df_metadata.append(new_record, ignore_index=True)
            del(new_record)

        time.sleep(0.5)
```

```
0 | AVALON CARE CENTER - OTHELLO, LLC
1 | PRESTIGE CARE & REHABILITATION - CLARKSTON
2 | LIFE CARE CENTER OF KENNEWICK
3 | LIFE CARE CENTER OF RICHLAND
4 | Regency Canyon Lakes Rehabilitation and Nursing Center
5 | RICHLAND REHABILITATION CENTER
6 | Cashmere Care Center
```

Figure 7. An example of a relatively elaborate scraping code from *The Seattle Times*.[22] Lines preceded by "#" are comments; the text within the gray box is the code; the text below that box is the output. The output shows the data sources as they get converted into a format appropriate for data analysis.

Harlem neighborhood to New York City's 311 hotline, revealing a spike in "quality of life" complaints.

As noted, variables in our classification schema are not mutually independent, and the resulting clusters overlap. Some 84% of cases in the data wrangling cluster also include code to analyze the assembled data (Table 2, column 3); these stories mostly share the epistemological model of investigative data journalism discussed previously. (Cluster 6, another analysis-oriented cluster reviewed below, also includes some scraping code.)

Our CCS-informed approach thus highlights the importance of data wrangling in assembling datasets (RQ2), which has often been overlooked in existing scholarship. In the examples above, scraping allows journalists to gather institutional data on their own terms, again pushing against the dynamics of information subsidy; it is counterintuitively but unmistakably *editorial* effort, as reflected in the occasional published accounts of this work. Data wrangling is ubiquitous and labor intensive but celebrated only rarely, even among the practitioners. We observed that it is more likely to be highlighted in news reports when the resulting database is itself newsworthy, but not in numerous other cases where the data is "consumed" within the analysis.

Theme 4: Abundance of Data Drives "Quick Turnaround" Data Journalism

Despite the oft-noted reliance on the "information subsidy" of government data in journalism generally and data journalism in particular, our cluster analysis also highlights data-driven stories that create or obtain datasets from diverse sources including social media platforms, independent surveys, and miscellaneous online datasets. Qualitative review underscores that these are generally not investigative reporting, but instead focus on what we call "social analysis," taking advantage of widespread datafication to uncover interesting cultural or political trends. As Lowrey & Hou note, data journalism's "relevance and proliferation will certainly co-evolve with the increasing datafication of society as a whole" (2021, p. 16).

One set of stories in the "social analysis" cluster rely on social media data, often from Twitter. For example, a 2018 NPR story[13] used textual analysis to explore how President Trump's use of words like "fake" and "phony" was increasing over time.[14] *Buzzfeed News* published a more elaborate story[15] in 2018, titled "All The President's Tweets—And Every Lawmaker's Too," used the Twitter API to retrieve politician's tweets and explore temporal patterns in tweeting, interaction and engagement, including sentiment analysis (Figure 8). These stories mirror the trend in social sciences – for instance, in the field of mass communication – where the availability of social media data can shape lines of inquiry (boyd and Crawford 2012).

Another set of stories use a variety of private or academic databases to conduct similar analyses. For example, a 2018 BuzzFeed story[16] titled "'I Have The Best Words.' Here's How Trump's First SOTU Compares To All The Others," obtained transcripts from the American Presidency Project; the analysis code included calculating Flesch-Kincaid reading grade level, conducting sentiment analysis, etc. The same dataset drove a 2014 story, "No, Obama's Pronouns Don't Make Him A Narcissist," which compared Obama's use of first-person pronouns to previous presidents.[17] Similarly, during the #MeToo movement in 2017, *Buzzfeed News* published an analysis of how cable news

We ran a sentiment analysis on original tweets only, excluding retweets, using the lexicon developed by Bing Lui of the University of Illinois at Chicago and colleagues. When matching words in the tweets to the lexicon's list of words linked positive or negative emotions, we filtered out a set of custom stop words: "Trump," classed as positive in this lexicon, obviously had to be removed; "critical" and "issue(s)" are classed as negative in this lexicon, but are used differently by politicians. Values for negative words are shown as percentages of all positive and negative words.

```
# load lexicon from https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html
bing <- get_sentiments("bing")

# custom stop words, to be removed from analysis
custom_stop_words <- data_frame(word = c("trump","critical","issues","issue"))

# sentiment by mentioning trump or not
sentiments_trump_mentions <- tweets %>%
  filter(isRetweet == FALSE) %>%
  mutate(trump_mention = ifelse(grepl("@realdonaldtrump",text), "Y", "N"),
         text = str_replace_all(text, replace_reg, "")) %>%
  unnest_tokens(word, text, token = "regex", pattern = unnest_reg) %>%
  filter(str_detect(word, "[a-z]")) %>%
  # match to lexicon and remove custom stop words
  inner_join(bing, by = "word") %>%
  anti_join(custom_stop_words)
```

Figure 8. *BuzzFeed News*' analysis code uses a sentiment classifier to numerically measure the "negativity" in politicians' tweets.[23]

covered sexual misconduct allegations, relying on data from the Television News Archive, the GDELT Project etc.

These news stories based on textual analysis differ from the investigative analysis projects highlighted previously, and rarely showcase the dataset as a journalistic goal (note the low value of *data featured*). As noted, one can argue that these relatively minor stories exist as a byproduct of widespread datafication. Stories in the social analysis cluster fit the descriptions of what Borges-Rey calls "a daily, quick turnaround, generally visualised, brief form of data journalism" which is often editorialised and entertaining (2016, p. 841). The code in these stories qualifies as quantification under our classification schema (Table 2, column 6): it converts qualitative data (such as text) to numeric data (such as a numerical measure of sentiment), as shown in Figure 8.

A third set of cases we observed center on analyzing survey data. For instance, a 2016 report from *Buzzfeed News*, titled "This Is How 23 Countries Feel About Transgender Rights," features a survey conducted in partnership with a polling firm and the University of California, Los Angeles. Both the survey data and the analysis code are available in the repository. Although it does not appear in our sample, the well-known, award-winning site FiveThirtyEight epitomizes this style of survey-driven data journalism. A few scholars have also noted this trend of data journalism relying on surveys to generate original data (Loosen, Reimer, and De Silva-Schmidt 2020, 9; Lowrey and Hou 2021, 13). The small number of such cases in our sample suggest that surveys—historically a major source of data in quantitative social science—play a secondary role in data journalism relative to other data sources.

RQ2 asked what a CCS approach reveals about novel methods to assemble datasets. The examples above illustrate that data journalists are leveraging widespread datafication to collect their own data, going beyond traditional sources such as public data or FOIA requests; this dimension is further developed in the discussion of "algorithmic accountability reporting" below. Meanwhile, RQ1 focused on how the data and analysis relate to claims in the news report. The examples above highlight how—in a parallel to critiques of "big data" in academia (boyd and Crawford 2012)—an abundance of data has invited a new genre based on relatively low-effort data mining to uncover interesting sociopolitical trends.

Theme 5: Algorithmic accountability reporting

The cases in the previous section reflect how social media data and other non-governmental data supports quick-turnaround data journalism. We now describe examples from two clusters which we collectively call "tech reporting." Projects in these clusters also do not rely on government data, but do highlight dataset creation in published stories (*data featured*, Table 2, column 6). Qualitative review reveals another crucial distinction: Projects in these clusters tend to harvest data from platforms and other major tech firms using specialized methods, and use the created datasets to scrutinize the firms' policies and opaque algorithms. The two clusters differ only in whether the code they use is shared on GitHub; both share the epistemic model of reverse-engineering computational artifacts (Diakopoulos 2015).

A prominent case in our sample is *The Markup*, a newsroom which states on its website: "Big Tech Is Watching You. We're Watching Big Tech." In 2021, *Markup* launched an investigative series called "Citizen Browser" which scrutinizes Facebook's algorithms. To retrieve data needed to reverse-engineer the platform's algorithms, the outlet paid volunteers to run an application it created to monitor which groups and pages Facebook recommended to its users. The app, though based on an open-source framework and audited by a third-party security firm, is not shared on GitHub (likely due to privacy concerns). Below are several illustrative news stories based on the analysis of the collected data:

1. "Facebook Isn't Telling You How Popular Right-Wing Content Is on the Platform"
2. "Facebook Said It Would Stop Recommending Anti-Vaccine Groups. It Didn't"
3. "How Big Pharma Finds Sick Users on Facebook"
4. "Facebook Promised to Remove 'Sensitive' Ads. Here's What It Left Behind"

Numerous *Markup* projects which follow the same model make their code available (cluster 6). A story titled "Google Has a Secret Blocklist that Hides YouTube Hate Videos from Advertisers – But It's Full of Holes" offers an instructive example of reverse-engineering. According to *Markup*, the "goal was to examine whether [YouTube] was limiting or facilitating the placement of ads on hate content"[18] by exploring how comprehensive its secret list of blocked keywords was. To uncover the blocklist, *Markup* located an undocumented and unpublicized YouTube API; systematically varying inputs to the API (random strings, hate speech, and uncontroversial words) revealed four classes of API outputs (no results, full results, blocked keywords, and partially blocked keywords), as shown in Figure 9.

Empty response

Returned labels (JSON keys) for a given keyword, but no results.

```
{"videos": [], "channels": []}
```

Full response

Returned suggested videos and channels for the given keyword.

```
{"videos": [v1, v2, …, vN], "channels": [c1, c2, …, cN]}
```

Blocked response

Returned no labels or results.

```
{}
```

Partially blocked

Returned a single invalid video with a gray "video" thumbnail, and the search term appeared in lieu of functional video metadata. These searches returned suggested channels when available (similar to a full response), but sometimes returned only a label (similar to an empty response).

```
{"videos": ["search_term"], "channels": [c1, c2, …, cN]}
```

Figure 9. *Markup*'s method for reverse-engineering YouTube's blocklist, as explained on their website.[24] shown are four possible (simplified) responses from the API for a given input keyword.

Previous scholarship has differed in regard to this style of data journalism. Diakopoulos highlights "algorithmic accountability reporting" undertaken by leading news organizations, which uses "sophisticated reverse engineering investigations" to uncover the workings of influential algorithms (2015, p. 398). However, Borges-Rey argues in the case of the United Kingdom that despite "a remarkable understanding of the computing language and logics behind this datafication of the world" (2016, p. 833), data journalists are mostly unable to access data held by "data corporations" and must rely on "traditional methods such as leaks or whistleblowers" (2016, p. 842). These contrasting findings may suggest this is a specialized, emerging subgenre of data journalism, perhaps strengthened by heightened scrutiny of tech platforms in recent years.

A smaller set of cases within these clusters use easily accessible social media data to reveal shortcomings in platforms' content moderation policies. For instance, *Buzzfeed*

News used the Twitter API for a 2018 story[19] titled "An Inside Look At The Accounts Twitter Has Censored In Countries Around The World." *Buzzfeed* criticized Twitter, stating that the investigation "provides an unprecedented glimpse into Twitter's collaboration with national groups and governments—democratic and authoritarian alike." A story from 2017, titled "These Are 50 Of The Biggest Fake News Hits On Facebook In 2017," used data from the social analytics service Buzzsumo to question the effectiveness of Facebook's efforts to curb fake news.

To summarize, our CCS-informed reading of the repositories (RQ2) reveals a specific, emerging topical thrust or subgenre of data journalism: stories in which data journalists apply sophisticated techniques to gather original data about platforms and algorithms, interrogating the social role of major tech firms. The data and analysis is used to make claims in the news report (RQ1) in usually a straightforward way—the analysis reveals failures of tech companies' policies.

Conclusion

This paper relies on an original approach, informed by Critical Code Studies and combining quantitative and qualitative methods, to investigate the uses of data and code in real-world data journalism. Several of the conclusions reinforce and add texture to recurring findings within existing scholarship: data journalism's heavy reliance on institutional sources for data; the creation of public databases considered as a journalistic activity; and analysis of institutional data to reveal systemic issues. However our CCS-led approach, including analyzing code, also revealed major two facets of data journalism which are often overlooked in existing scholarship.

First, the heavy labor of scraping, formatting and cleaning data – to either create a public database or conduct further analysis – was highlighted when we looked at the code. This labor is addressed in the "data science" field (e.g. its textbooks) but seldom considered in journalism studies. This practice counters, in a limited way, data journalism's low capacity for original data collection and heavy reliance on institutional sources. Furthermore, we found that the institutions often provide data which is effectively unusable, perhaps intentionally so. In these cases, journalists do the labor of making this data comprehensible and easily accessible, or provide interpretations of it (e.g. COVID-19 or government budget data). Second, to hold tech companies accountable, journalists are adopting the methodology of reverse-engineering computational artifacts (Diakopoulos 2015; Kitchin 2017), which is distinct from traditional journalistic methods. Reverse-engineering is likely supplementary to traditional methods relying on whistleblowers and information from insiders.

As scholars focus to a greater degree on the role of code and AI in journalism (Simon 2023), it will be increasingly necessary to analyze programming code itself—and, we suggest, to read code holistically as demonstrated in this paper. Unlike some previous studies on journalism code, we did not confine ourselves to the framework of transparency (whether code and data were complete and the results replicable). Instead, this analysis illustrates that looking contextually at the assemblage of code, data, repository, and published news report, has more to offer. First, it can compensate for missing or inaccessible elements in journalistic code (such as a proprietary API) by providing a

generalized picture of the work that code and data are "doing". Second, a contextual reading helps to focus the analysis of code on relevant sections when it is impractical to understand every line of code in a sizable sample. However, our operationalization of CCS also has limitations. We focused on three functions of code (scraping, analysis, and quantification); future research could focus on alternative categories and in greater depth. For example, we did not formally classify the complexity within analysis and scraping code, or the use of libraries and packages for analysis. The role of code could also be understood by considering its engagement and activity metrics on GitHub.

Lastly, we wish to offer some comments on GitHub based on some unexpected observations made during our analysis. GitHub appears to fulfill two infrastructural roles within data journalism. First is transparency by sharing of analysis material as a "proof" for the findings in the news report—ordinary users are not invited to make contributions because the analysis is supposed to be complete. Second is using GitHub to crowdsource databases and track changes to it—crowdsourcing provides missing data rather than changing the interpretations of the database or claims in the news report itself[20] (e.g. Figure 4). These kinds of transparency are overlooked in scholarship focusing on GitHub and "open-source" journalism (Lewis and Usher 2013) and need further study.

Notes

1. Since we did not have the news story's weblink a priori, we imposed the less restrictive condition that at least one weblink in the repository should point to the news website's domain, such as "nytimes.com.".
2. https://github.com/nytimes/haiti-debt
3. https://github.com/washingtonpost/data-congress-slaveowners
4. https://github.com/datadesk/homeless-arrests-analysis
5. https://github.com/PublicI/sba-disaster-loans
6. https://github.com/washingtonpost/data-2C-beyond-the-limit-usa
7. https://www.washingtonpost.com/graphics/2019/national/climate-environment/climate-change-america/.
8. https://github.com/BuzzFeedNews/2021-05-tx-winter-storm-deaths
9. https://github.com/datadesk/california-coronavirus-scrapers
10. https://github.com/seattletimes/nursing_homes_staffing_20200925
11. https://github.com/seattletimes/nursing_homes_staffing_20200925
12. https://github.com/BuzzFeedNews/2018-06-nyc-311-complaints-and-gentrification
13. https://github.com/nprapps/trump-tweet-analysis
14. https://www.thetrumparchive.com/.
15. https://github.com/BuzzFeedNews/2018-01-trump-twitter-wars
16. https://github.com/BuzzFeedNews/2018-01-trump-state-of-the-union
17. https://github.com/BuzzFeedNews/presidential-language-notebooks
18. https://themarkup.org/google-the-giant/2021/04/08/how-we-discovered-googles-hate-blocklist-for-ad-placements-on-youtube
19. https://github.com/BuzzFeedNews/2018-01-twitter-withheld-accounts
20. Additional example: https://github.com/theintercept/trial-and-terror-data/commit/7295de62c72b41bc0eef1462f1fcd7e4776fe0ba
21. https://github.com/washingtonpost/data-congress-slaveowners/commit/a64e5de1e68be2dbc3cd2972b8ff9a40ae32eb5b.
22. https://github.com/seattletimes/nursing_homes_staffing_20200925/blob/master/B_scripts/A_Enf_Letters_Step1_Download_PDFs.ipynb

23. https://buzzfeednews.github.io/2018-01-trump-twitter-wars/.
24. https://themarkup.org/google-the-giant/2021/04/08/how-we-discovered-googles-hate-blocklist-for-ad-placements-on-youtube
25. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering

Disclosure Statement

No potential conflict of interest was reported by the author(s).

ORCID

Aman Abhishek http://orcid.org/0000-0003-3686-6052
Lucas Graves http://orcid.org/0000-0002-2980-7145

References

Anderson, C. W. 2018. *Apostles of Certainty: Data Journalism and the Politics of Doubt*, *Oxford Studies in Digital Politics*. New York: Oxford University Press. https://doi.org/10.1093/oso/9780190492335.001.0001

Baack, S. 2015. "Datafication and Empowerment: How the Open Data Movement Re-Articulates Notions of Democracy, Participation, and Journalism." *Big Data & Society* 2 (2): 205395171559463. https://doi.org/10.1177/2053951715594634

Baack, S. 2018. "Practically Engaged." *Digital Journalism* 6 (6): 673–692. https://doi.org/10.1080/21670811.2017.1375382

Borges-Rey, E. 2016. "Unravelling Data Journalism." *Journalism Practice* 10 (7): 833–843. https://doi.org/10.1080/17512786.2016.1159921

Borges-Rey, E. 2020. "Towards an Epistemology of Data Journalism in the Devolved Nations of the United Kingdom: Changes and Continuities in Materiality, Performativity and Reflexivity." *Journalism* 21 (7): 915–932. https://doi.org/10.1177/1464884917693864

boyd, D., and K. Crawford. 2012. "Critical Questions for Big Data." *Information, Communication & Society* 15 (5): 662–679. https://doi.org/10.1080/1369118X.2012.678878

Boyles, J. L. 2020. "Deciphering Code: How Newsroom Developers Communicate Journalistic Labor." *Journalism Studies* 21 (3): 336–351. https://doi.org/10.1080/1461670X.2019.1653218

Coddington, M. 2015. "Clarifying Journalism's Quantitative Turn: A Typology for Evaluating Data Journalism, Computational Journalism, and Computer-Assisted Reporting." *Digital Journalism* 3 (3): 331–348. https://doi.org/10.1080/21670811.2014.976400

Coddington, M. 2018. "Defining and Mapping Data Journalism and Computational Journalism: A Review of Typologies and Themes." In *The Routledge Handbook of Developments in Digital Journalism Studies*, edited by Scott Eldridge II and Bob Franklin. London: Routledge.

De Maeyer, J., M. Libert, D. Domingo, F. Heinderyckx, and F. Le Cam. 2015. "Waiting for Data Journalism." *Digital Journalism* 3 (3): 432–446. https://doi.org/10.1080/21670811.2014.976415

Diakopoulos, N. 2015. "Algorithmic Accountability: Journalistic Investigation of Computational Power Structures." *Digital Journalism* 3 (3): 398–415. https://doi.org/10.1080/21670811.2014.976411

Fehr, T., and J. Williams. 2021, July 9. *Tracking Covid-19 From Hundreds of Sources, One Extracted Record at a Time*. NYT Open. https://open.nytimes.com/tracking-covid-19-from-hundreds-of-sources-one-extracted-record-at-a-time-dd8cbd31f9b4

Gandy, O. H. 1982. *Beyond Agenda Setting: Information Subsidies and Public Policy*. Norwood, NJ: Ablex Pub. Co.

Gans, H. J. 2004. "*Deciding What's News: A Study of CBS Evening News*." In *NBC Nightly News, Newsweek, and Time*. Evanston, IL: Northwestern University Press.

Gray, J., and L. Bounegru. 2019. "What a Difference a Dataset Makes? Data Journalism and/as Data Activism." In *Data in Society: Challenging Statistics in an Age of Globalisation*, edited by J. Evans, S. Ruane, and H. Southall, 365–374. Bristol University Press.

Haim, M., and R. Zamith. 2019. "Open-Source Trading Zones and Boundary Objects: Examining GitHub as a Space for Collaborating on "News." *Media and Communication* 7 (4): 80–91. https://doi.org/10.17645/mac.v7i4.2249

Houston, B. 2019. *Data for Journalists: A Practical Guide for Computer-Assisted Reporting*. New York, London: Routledge.

Kitchin, R. 2017. "Thinking Critically about and Researching Algorithms." *Information, Communication & Society* 20 (1): 14–29. https://doi.org/10.1080/1369118X.2016.1154087

Lewis, S. C., and N. Usher. 2013. "Open Source and Journalism: Toward New Frameworks for Imagining News Innovation." *Media, Culture & Society* 35 (5): 602–619. https://doi.org/10.1177/0163443713485494

Loosen, W., J. Reimer, and F. De Silva-Schmidt. 2020. "Data-Driven Reporting: An on-Going (r) Evolution? An Analysis of Projects Nominated for the Data Journalism Awards 2013–2016." *Journalism* 21 (9): 1246–1263. https://doi.org/10.1177/1464884917735691

Lowrey, W., and J. Hou. 2021. "All Forest, No Trees? Data Journalism and the Construction of Abstract Categories." *Journalism* 22 (1): 35–51. https://doi.org/10.1177/1464884918767577

Lowrey, W., L. Sherrill, and R. Broussard. 2019. "Field and Ecology Approaches to Journalism Innovation: The Role of Ancillary Organizations." *Journalism Studies* 20 (15): 2131–2149. https://doi.org/10.1080/1461670X.2019.1568904

Marino, M. C. 2006. *Critical Code Studies*. Electronic Book Review. https://electronicbookreview.com/essay/critical-code-studies/

Marino, M. C. 2020. *Critical Code Studies*. Cambridge, MA: The MIT Press.

Nguyen, D. 2010. "Scraping for Journalism: A Guide for Collecting Data." *ProPublica*. Accessed February 4, 2023. https://www.propublica.org/nerds/doc-dollars-guides-collecting-the-data.

Parasie, S. 2022. *Computing the News: Data Journalism and the Search for Objectivity*. New York: Columbia University Press.

Parasie, S., and E. Dagiral. 2012. "Data-Driven Journalism and the Public Good: "Computer-Assisted-Reporters" and "Programmer-Journalists" in Chicago." *New Media & Society* 15 (6): 853–871. https://doi.org/10.1177/1461444812463345

Petre, C. 2021. *All the News That's Fit to Click: How Metrics Are Transforming the Work of Journalists*. Princeton; Oxford: Princeton University Press.

Porter, T. M. 1995. *Trust in Numbers: The Pursuit of Objectivity in Science and Public Life*. Princeton, NJ: Princeton University Press.

ProPublica. 2016. "ProPublica Responds to Company's Critique of Machine Bias Story." July 29, 2016. https://perma.cc/FM9V-W5EU

Scott, J. C. 1998. *Seeing like a State: How Certain Schemes to Improve the Human Condition Have Failed*. New Haven: Yale University Press.

Simon, F. M. 2023. "Escape Me If You Can: How AI Reshapes News Organisations' Dependency on Platform Companies." *Digital Journalism* 12 (2): 149–170. https://doi.org/10.1080/21670811.2023.2287464

Stalph, F. 2018. "Classifying Data Journalism." *Journalism Practice* 12 (10): 1332–1350. https://doi.org/10.1080/17512786.2017.1386583

Usher, N. 2016. *Interactive Journalism: Hackers, Data, and Code*. Urbana: University of Illinois Press.

Uskali, T. I., and H. Kuutti. 2015. "Models and Streams of Data Journalism." *The Journal of Media Innovations* 2 (1): 77–88. https://doi.org/10.5617/jmi.v2i1.882

Washington, A. L. 2018. "How to Argue with an Algorithm: Lessons from the COMPAS-ProPublica Debate." *Colorado Technology Law Journal* 17: 131–160.

Weber, M. S., and A. Kosterich. 2018. "Coding the News: The Role of Computer Code in Filtering and Distributing News." *Digital Journalism* 6 (3): 310–329. https://doi.org/10.1080/21670811.2017.1366865

Zamith, R. 2019. "Transparency, Interactivity, Diversity, and Information Provenance in Everyday Data Journalism." *Digital Journalism* 7 (4): 470–489. https://doi.org/10.1080/21670811.2018.1554409

Appendix A: Filtering Relevant Repositories Using Keywords

List of keywords:

- Relevant: datum, analysis, news, article, story, data, support, report, methodology, finding, reproduce, investigation, database, dataset, track, tracker
- Irrelevant: style, template, app, application, service, server, wrapper, tool

We first removed repositories if their "about" field contained any irrelevant keywords. Out of the repositories that were left, we only kept those whose "about" field contained at least one relevant keyword.

Appendix B: Hierarchical Clustering

We used the hierarchical/agglomerative clustering algorithms in Pythons SciPy library[25]. The optimal number of clusters is exactly before the algorithm merges clusters which have a large distance between them. If this happens at multiple instances, then choosing between those instances is subjective.

In Figure B1, the distance between clusters is shown on the vertical axis. Figure B1(a) shows the number of clusters on the horizontal axis, while Figure B1(b) is a dendrogram with the repositories on the horizontal axis. The optimal number of clusters is the first point where the curve becomes flat in Figure B1(a). This happens first at 4 clusters and then at 9 clusters; we chose the latter because it captures more granular details of the data and constitutes the former. Then, we dropped two clusters (Misc#1 and Misc#2) because they were not conceptually meaningful despite being mathematically coherent, and merged two pairs of very similar clusters (Table B1).

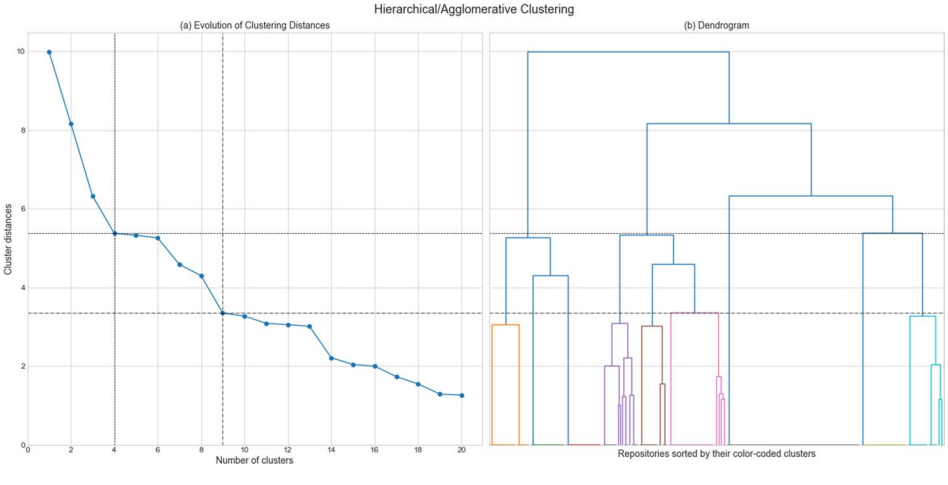

Figure B1. Hierarchical clustering.

Table B1. Cluster size distribution. Clusters marked with * were formed by merging a pair of very similar clusters. Misc#1 and Misc#2 were dropped from the analysis.

Cluster	Number of repositories	Percent within the sample (N = 234)
Analyzing Government Data*	93	40%
Tech Reporting*	34	15%
Social Analysis	30	13%
Data Archival	18	8%
Data Wrangling	19	8%
Misc#1	21	9%
Misc#2	19	8%