Table Scraps: An Actionable Framework for Multi-Table Data Wrangling From An Artifact Study of Computational Journalism

Stephen Kasica; Charles Berret; and Tamara Munzner, *Senior Member*, *IEEE*

Abstract—For the many journalists who use data and computation to report the news, data wrangling is an integral part of their work. Despite an abundance of literature on data wrangling in the context of enterprise data analysis, little is known about the specific operations, processes, and pain points journalists encounter while performing this tedious, time-consuming task. To better understand the needs of this user group, we conduct a technical observation study of 50 public repositories of data and analysis code authored by 33 professional journalists at 26 news organizations. We develop two detailed and cross-cutting taxonomies of data wrangling in computational journalism, for actions and for processes. We observe the extensive use of multiple tables, a notable gap in previous wrangling analyses. We develop a concise, actionable framework for general multi-table data wrangling that includes wrangling operations documented in our taxonomy that are without clear parallels in other work. This framework, the first to incorporate tables as first-class objects, will support future interactive wrangling tools for both computational journalism and general-purpose use. We assess the generative and descriptive power of our framework through discussion of its relationship to our set of taxonomies.

Index Terms—Computational journalism, Data journalism, Data wrangling.

1 Introduction

Data wrangling is an exploratory, iterative process of auditing and transforming data, encompassing tasks such as cleaning, integrating, and transforming datasets as an often necessary precursor for data analysis [18]. It is also an arduous process comprising a significant portion of effort data analysis and data warehousing projects [7, 19, 25]. Programmatic wrangling is typically carried out with general-purpose scripting languages such as R or Python, often augmented with supplemental libraries [52, 53]. In addition, several interactive tools have been designed to support data wrangling among data-literate non-programmers [4,15,18,43,46], incorporating visualization to assist in data auditing and evaluation of data transformations.

Observational studies of the data wrangling process could guide the design and evaluation of wrangling support systems, both programmatic and interactive. The interview study of enterprise data analysts [19] was a useful start, but many questions remain open. We choose to study a specific domain that is an microcosm of data wrangling: computational journalism. Journalists have two amenable characteristics as a target population: a clear need to wrangle and a culture of extensive process documentation. First, the need for transforming and cleaning raw data has been identified as a preeminent challenge [49]. Second, the culture of journalism valorizes transparency and providing evidence for reported conclusions [1, 36]. Many data-driven articles conclude with a methodology statement, informally known as the "nerd box," that typically includes a link to publicly released code to replicate the entire underlying analysis process, including all wrangling operations.

We conduct a technical observation study [30] of how professional journalists use scripting languages to wrangle data in the wild, from code repositories (repos) that journalists have made publicly available in conjunction with published articles. We study the actions of journalists who program as a first step in understanding the essential operations that should be supported for both programmers and non-programmers. By studying the data wrangling actions performed with the power and flexibility of code, we can better understand what an interactive data wrangling interface should provide to non-coding journalists.

Our work at the intersection of computer science and journalism is descriptive of computational journalism but not exclusive to it. Journalists created the artifacts used in our study with common data science tools, and we find that they encounter issues similar to users in other domains, such as enterprise data analysis [17].

Our observational study results in two descriptive taxonomies of data wrangling in computational journalism. These taxonomies, grounded in data, are created by the bottom-up method of open and axial coding on the technical artifacts of programming scripts and computational notebooks from a set of repos that we curated. One taxonomy describes the actions journalists took while wrangling their data, and the other features our interpretation of their wrangling processes.

The most novel finding of our study is the extent to which journalists make use of multiple tables in their wrangling activities, in contrast to previous wrangling frameworks that emphasize wrangling operations conducted within a single table. To fill this gap, we present an actionable framework for multi-table wrangling. The framework is designed with a concise structure that provides generative power to serve as the basis for building future interactive tools. We synthesize the framework through a top-down approach where we consider tables themselves as first-class objects, with equal footing to its rows and columns. We cross-check the coverage of the framework by ensuring it covers the many multi-table operations observed in our action taxonomy that do not fit in existing frameworks for data wrangling and harmonize with concepts and vocabulary from the existing literature.

We present two primary contributions: two detailed descriptive taxonomies of data wrangling in computational journalism and a concise framework for multi-table data wrangling designed to be an actionable basis for developing future general-purpose tools.

We also present two secondary contributions of document corpora as supplemental material. The first corpus provides links to the curated set of 225 repos. The second corpus is the subset of 50 repos used in our study, fully annotated with open codes to provide transparent supporting evidence of our analysis process [24]. These secondary contributions may also prove useful for researchers interested in better transformation recommendations in data wrangling [17].

2 Related Work

We review related work on characterizing the data wrangling process in journalism and computer science.

2.1 Data Wrangling in Journalism

A small body of work characterizes wrangling challenges faced by journalists, often in the form of retrospective articles about the data

- *Stephen Kasica and Tamara Munzner are with the University of British Columbia, Department of Computer Science. E-mail: {kasica,tmm}@cs.ubc.ca.*
- *Charles Berret is with the University of British Columbia, School of Journalism, Writing, and Media. E-mail: cberret@mail.ubc.ca.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

analysis process for individual stories as well as processes used by the news organization in general. Nguyen [27] details how a team of journalists at *ProPublica* used Google Refine to perform entity resolution on data underlying an investigation into payments to US doctors from pharmaceutical and medical device companies. Rogers [34] outlines a general data journalism workflow where common data cleaning tasks constitute early steps, estimating that data journalists spend the majority of their time engaged in cleaning and merging datasets [33]. The *Quartz* Bad Data Guide [10] enumerates many real-world data issues faced by journalists. While these works are largely issue focused, concentrating on common data errors and issues, our descriptive taxonomy is action focused, concentrating on ways to resolve data issues.

Although the majority of material on wrangling in journalism comes via popular articles, there have been a few academic and industry studies. Cohen et al. [6] asserts that cleaning and verifying the contents to merge multiple data sources is a common task in data journalism. A recent survey of data journalists found that about half of respondents reported creating data-driven stories in a day or less, including time spent wrangling [35]. A report from The American Press Institute characterizes the skill levels for common wrangling operations supported by spreadsheet applications [44].

Although many journalists use general-purpose wrangling tools, the tools they build for themselves also provide insight into issues they face. The command-line tool `csvkit` [12] and its Python equivalent `Agate` [11] provide functionality for wrangling tasks such as unique key generation, pivoting a table by one or more columns, and deriving additional columns. Workbench [55] is a data journalism platform built around community-contributed modules for wrangling, analysis, and visualization. While the process of eliciting design requirements for these tools has not been clearly documented, our work provides a systematic characterization to inform future tool design in this area.

2.2 Data Wrangling in Computer Science

Tools for wrangling data fall into two categories: scripts written in various programming languages and interactive applications.

Scripting languages such as Python and R have commonly been used to wrangle data [18], often with supplemental libraries to further facilitate the process. Pandas [32], plyr [50], dplyr [52], and tidyr [53] are widely used for data wrangling in a programming environment. These packages incorporate structures for representing heterogeneous data within their environment as analogous to tables. Both Python and R have tools to instantiate the wrangling design principles of tidy data [51] and the split-apply-combine strategy for data analysis [50]. Our work synthesizes and incorporates these ideas in our multi-table framework in a way that can be applied to interactive interfaces.

While general-purpose tools like Microsoft Excel implicitly support many common wrangling operations, several interactive tools designed explicitly for wrangling offer an expanded set of operations accessible to empower data-literate non-programmers. OpenRefine [15], Trifacta Wrangler [46], and Tableau Prep Builder [45] enable users to clean, edit, and merge data using a menu-based GUI and application-specific custom functions. These systems leverage data visualization in an iterative, human-in-the-loop process of data auditing and transformation. Trifacta Wrangler (also known as Google Cloud Dataprep) and its predecessor Wrangler [18] incorporate transformation recommendations throughout the wrangling process. These applications invoke a spreadsheet or matrix view where the table is synonymous with the environment. While they may support operations to merge multiple data sources, they do not do so by treating the table as a first-class citizen in the wrangling environment.

While many existing wrangling applications do support some operations that involve multiple tables, none support the whole breadth of possible operations within this space. Data integration is a subproblem within wrangling [18] and mashup applications are often cited as examples of interfaces that address this aspect of wrangling [17, 18]. However, these applications are primarily concerned with extracting data from HTML web pages and secondarily incorporating extracted data into other structured data. Many of them support JOIN-like operations [47, 54] common in Structured Query Language (SQL) or perform data integration without the notion of a table [16, 21]. None of these applications support the kind of operations that map one table to many tables like we observed users performing in our technical observation study.

Some research addresses wrangling network data. However, we did not observe journalists extensively performing network analysis techniques, and previous work addresses just how rarely network analysis is used by journalists [42]. Ploceus [23] and Orion [14] implicitly support basic graph editing operations. Origraph [4] explicitly supports network wrangling, expanding upon these wrangling operations. Concepts from these papers inspire some of the operations in our multi-table framework; however, these systems do not directly support multi-table wrangling of tabular data.

There are only a few observational studies of the data analysis process that includes data wrangling. Kandel et al. [19] performed an interview study of enterprise data analysts. Muller et al. [25] perform an interview study with data scientists, and describe five ways data science workers engage with data. Our work differs in both domain and methodology. We focus on wrangling by computational journalists using programming languages, and we use observation of technical artifacts as our data collection method.

3 Process and Methods

The research questions addressed in this paper are:

Q1: *What are the wrangling practices of data-literate journalists with programming skills?*

Q2: *Which practices align with or diverge from existing characterizations?*

Q3: *How to re-characterize wrangling to match the observed practices?*

Our process has three phases, one for each question (Figure 1). The first phase studies the wrangling processes of journalists using the qualitative method of open and axial coding on technical artifacts. These artifacts document the analysis underpinning articles and are published in conjunction with them in publicly available repositories. The product of this phase is an initial bottom-up, descriptive taxonomy of data wrangling in computational journalism. The second phase employs an interdisciplinary literature search to compare our taxonomy to the existing literature on process theories of data wrangling. We conduct The third phase via reflective synthesis, to create a concise framework for multi-table wrangling with 21 operations. We then check that this cross-cutting framework fully covers all of the wrangling actions we observe in the study and document in the bottom-up taxonomy.

3.1 Phase One: Qualitative Coding Study Overview

In the first phase we addressed Q1 through qualitative coding of programming scripts and computational notebooks supporting published articles. We systematically searched GitHub and ObservableHQ to identify an initial corpus of more than 1,000 journalistic code repositories related to journalism. From these initial results, we inspected each repository to identify those containing data analysis, resulting in a set of 225 repositories.

This curated corpus served as the basis for the data in our technical observation study. Through an iterative process of manually selecting repos according to criteria that ensure diversity of both individuals and organizations, we produced a final set of 50 annotated repos documenting journalists' data analysis process with open codes. Through axial coding, we produce a descriptive, bottom-up taxonomy of wrangling in computational journalism grounded in this observational data.

Studies of provenance in E-Science make a distinction between whether records are data or process oriented [29, 38, 39]. We also distinguish between data and process in the qualitative coding portion of the first phase. The cross cutting nature of our taxonomy occurs along two dimensions: wrangling *actions* performed by the journalists upon the data, which are orthogonal to descriptions of the wrangling *process*.

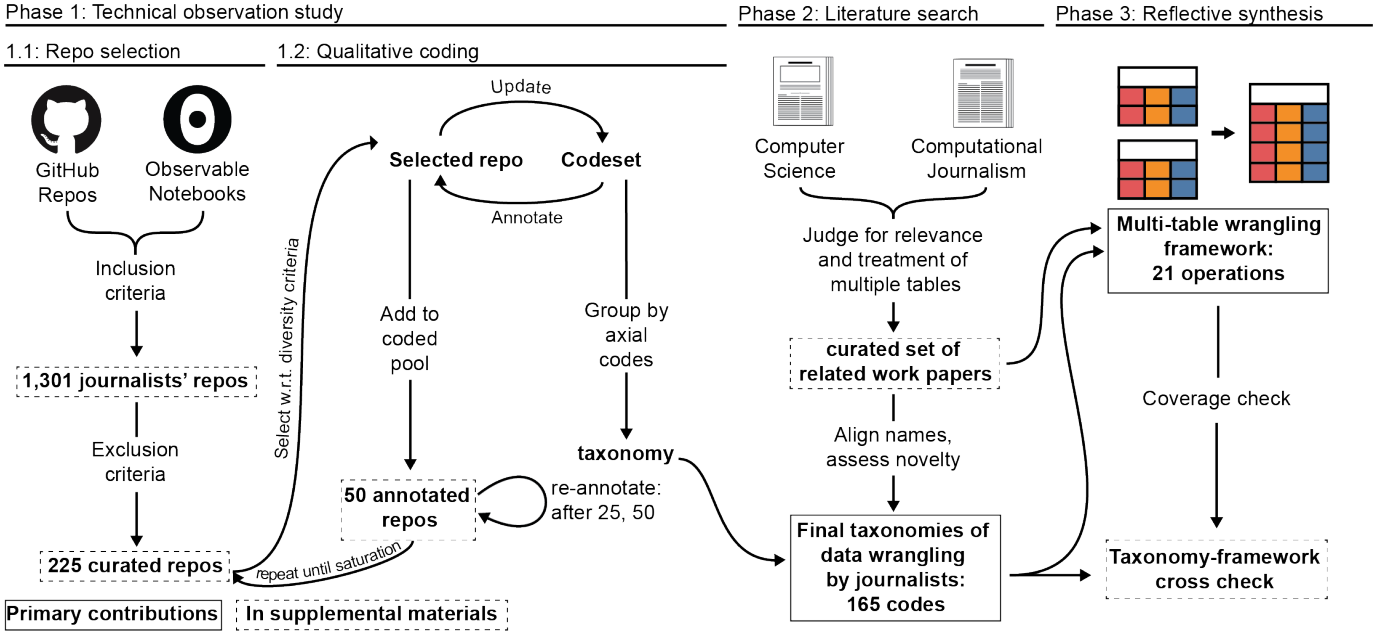


Fig. 1. Three-phase process: observation study of technical artifacts conducted through qualitative coding of journalist repos, resulting in two initial bottom-up taxonomies of 165 open and axial codes; literature search to align naming and assess novelty; reflective synthesis to create a concise top-down multi-table wrangling framework with 21 operations.

In related work on data wrangling, journalists are considered a non-technical user group [18], but there is a data-literate subset of journalists who routinely practice technical tasks such as data wrangling, database management, statistical analysis, and data visualization. These data-literate journalists can be further classified by how much of this work is performed with programming languages versus interactive tools. The repo authors in this corpus are a sample of technically savvy data journalists fluent in various programming languages. Although this group does constitute the minority of data journalists [26, 44], we expect theories generated from studying the practices of data-literate programmer journalists will transfer to the larger population of data-literate non-programmer journalists. We conjectured that these programmatic approaches to wrangling would be more expressive than the operations supported by interactive applications, which aim to democratize the tools and techniques for data manipulation [4, 18].

3.1.1 Observation of Technical Artifacts

We employ the term *technical observation study* from the software engineering literature to denote a data collection strategy of observing user-generated technical artifacts, such as source code [30]. This approach is similar to indirect observation, as both methods involve mediated, *post hoc* analysis of a user group. Whereas indirect observation studies collect data through researcher-developed tools like keystroke logs or transcriptions of audio and video recordings [37], technical observation data is an artifact of the phenomenon itself.

Wrangling is especially well suited to this technical-observational approach. The chief product of wrangling is not only the transformed and cleaned data, but also the record of the transformations applied to the raw data [17, 18]. Programming scripts and computational notebooks constitute an auditable and reproducible transformation record. As a result, this approach is positioned to produce theories of *how* users wrangle their data with strong ecological validity. Thus, when forming our taxonomy, we implemented this bottom-up approach of qualitative coding by annotating journalists' scripts and notebooks with open and axial codes, grounding our findings in these transformation records.

Technical observation allows us to quickly and easily analyze data with high ecological validity with no demands on the target population's time. This approach does have limitations. We limit our claims to focus on *how* journalists wrangle data, with limited conjecture into *why* they perform these actions. While code comments and the before-and-after state of a table may provide some sense of the user's motivation, qualitative research methods such as interviews and direct observation are better suited to this question [37]. Although previous work notes that those engaged in data analysis often explore alternatives [22], these repos typically contain a straightforward pipeline from the raw data to its final form, and may omit false starts and dead ends. Moreover, our collection process filters out all instances of unsuccessful wrangling. Our study may thus paint a simpler view of wrangling than the reality.

3.1.2 Repository Selection

We compiled an initial corpus of public code repositories from GitHub and ObservableHQ by two inclusion criteria: being relevant to journalism and being written in a common programming language used for data wrangling. We use the term *repo* to refer to any collection of related materials in either platform. This process concluded with a set of curated repos containing journalists data analysis, which we include in supplemental materials.

On GitHub, we identified journalistic repos through two avenues. We conducted a programmatic search using the platform's Search API, parameterized by topic, owner, and programming language. We satisfied the relevance to journalism criterion by identifying repos with a *journalism* or *data journalism* topic tag. We also referenced `@NewsNerdRepos`, a Twitter bot that posts new repos published by journalists. Any GitHub user or organization monitored by this account[1] satisfied the journalism-relevance criterion. We satisfied the wrangling language criterion by restricting our search to those repos where the predominant programming language in the repo is R, Python, or Jupyter Notebook. We chose R and Python due to their inclusion in previous wrangling papers [4, 18] and added Jupyter due to its rising popularity for data analysis. In order to gather R Markdown files, an idiosyncrasy of GitHub forced us to include HTML in the search parameters, leading to the inclusion of many web applications such as front-end visualizations or news applications that do not contain examples of data wrangling. We manually inspected the contents of each repo to exclude irrelevant ones, yielding a much smaller corpus of curated repos.

ObservableHQ is a computational notebook environment similar to Jupyter, where Observable notebooks are created using the front-end web development tools HTML, CSS, and JavaScript. We included a total of 36 repos from three journalists at major daily newspapers

[1] github.com/silva-shih/open-journalism

who were panel members of a conference workshop on this topic [5], automatically satisfying the relevance criterion. The portion of these repos applicable to wrangling are written in JavaScript, adding the diversity of another wrangling language to our corpus of coded repos. After similar manual filtering, we retained 34 repos from this set.

3.1.3 Qualitative Coding

We select a single repo at a time from the curated corpora to analyze through a manual selection procedure using two inclusion criteria: an explicit connection to a published article and increased diversity in experience or practices. By deliberately selecting only repos with corresponding articles, we ensure that our results are based on work where journalists were successful in wrangling raw data for their analysis or visualization needs. We deliberately select repos authored by different journalists at different news organizations to ensure a wide range of tools, practices, and experience. With each repo, we add new codes to our codeset, refine existing codes, and opportunistically apply new codes to previous notebooks. We performed axial coding in batches, splitting and consolidating code groups as needed. The intent of previously encountered behavior often became clearer as the coder encountered similar examples. In addition to opportunistic, retroactive application of codes, the coder also systematically re-coded earlier work after completing 25 and 50 repos, at which point we reached theoretical saturation, concluding the first phase.

Once collecting more data resulted in diminishing returns with respect to information that influenced our newly constructed theories, we determined that our study had reached saturation and required no further data. Two factors signaled reaching this point. First, the cardinality of our codeset began to level off, and the internal structure ceased to change. Adding new codes did little to modify the internal structure of axial codes. Thus, our taxonomies adequately described new phenomena. Second, the first author drew upon previous experience as a journalist and assessed that our theory conveyed a thorough understanding of major themes; sustained engagement in a field can help researchers achieve theoretical saturation [8]. At the conclusion of this stage, the open and axial codes were exported as the initial taxonomies.

All repos from the curated corpora were analyzed by a single coder, the first author. We use established qualitative criteria: transparent reporting of our procedure, thick description of sample data, and triangulation with related research. In contrast, quantitative analysis with a closed codeset often involves a positivist approach with multiple coders who converge above a threshold for inter-coder reliability to demonstrate replicability [37], but we felt the interpretive agenda of qualitative research was a better match with our goals.

While coding the first initial notebooks, we began with a single taxonomy, but further into this process, we saw two different types of code emerging, regardless of the level at which they describe phenomena. The first comprises *actions* journalists made upon their data in which we could reasonably infer their motivation based on the consequence of their operations and the semantic APIs of certain coding-based wrangling tools. The second concerned our own observations about the wrangling *process* at a level higher than just short sequences of transformation operations. Thus, we retroactively pivoted one taxonomy into two separate taxonomies, one for *actions* and one for *processes*.

Data Flow Sketches: We quickly noticed that journalists frequently employ multiple tables. To facilitate our own understanding of their activity, we sketched table-based data flow diagrams of how raw data is transformed through the wrangling environment when tables are used in complex ways. Figure 2 shows an example. These diagrams were instrumental for the central finding, reflected in our taxonomies, that journalists often employ many tables in ways not addressed by previous characterizations of wrangling operations.

3.2 Phase Two: Literature Review

In the second phase, we address Q2. Through a search of relevant literature, our goals are to reconcile the labels of our codes with terminology from the research literature and to assess the novelty of phenomena observed in the previous phase.

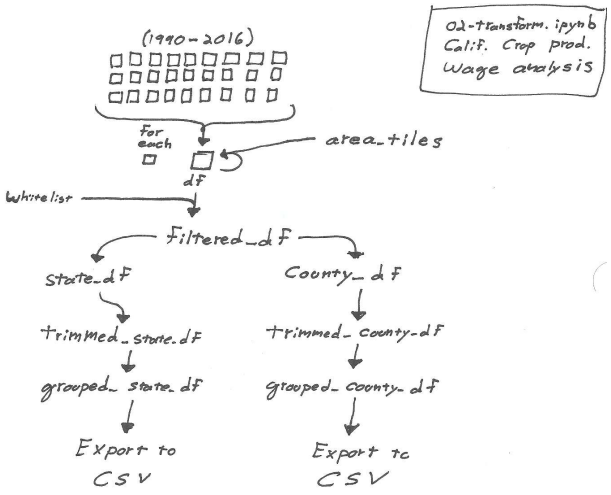


Fig. 2. A sketch of data flow through a notebook authored by journalists at the Los Angeles Times shows a wrangling process using more than two dozen tables before exporting two datasets for analysis and visualization.

We conduct our literature search in two stages, both performed by the first author. The first stage involves assembling a set of seed papers from two research domains: data wrangling in the computer science literature and journalistic data analysis in both popular and academic literature. We identify seed papers through targeted searching informed by the authors' background knowledge. Following the creation of preliminary observational taxonomies, we expand this set by following citations of papers that address the role of multiple tables and data sources in the wrangling process, using these papers' discussion of data integration, mashups, or network wrangling as inclusion criteria. We review this expanded set of seed papers, noting those with similarities and differences to observed phenomena. We harmonize the nomenclature in our taxonomy to align with previous usage to create the final version of the taxonomy, resulting in dozens of name changes. For example, *Create a Semi-Unique Key* was an early code describing users generating keys with no guarantee or assessment of uniqueness, such as concatenating given names and family names. We rename this code to *Create Soft Key* to align with literature in data cleaning [7].

3.3 Phase Three: Reflective Synthesis

In the final phase, we address Q3. By reflecting on other taxonomies of data wrangling operations [4, 14, 18] in relation to our observational findings, we synthesize a design space for multi-table data wrangling. Many interactive wrangling applications specify a design space through a domain specific language or enumerate supported operations. While these design spaces are informed by work in data transformation languages or personal experience in data wrangling, our work is grounded in observational data gathered from users wrangling data in the wild with the full flexibility of scripting languages. The result of this process is our Framework for Multi-Table Wrangling, described in Section 5.

The motivation for this second framework is our interest in finding a process model to guide the design of existing and future wrangling tools. The utility of a conceptual framework for guiding designers can be evaluated in terms of both descriptive and generative power [2]. Our set of descriptive taxonomies is centered on users, organized around actions, and observed in wrangling processes. This set of taxonomies provides a rich vocabulary for describing the actions and processes of journalists. However, they lack generative power: it would be very difficult for a designer to use them for guidance in building a system.

In contrast to the user-centered focus of our taxonomies, this framework centers on the state of the data itself, defining transformation operations by the cardinality and type of its inputs and outputs. One hurdle when navigating the design space of wrangling transformations is the inconsistent nomenclature used across computational, statistical, database, and data wrangling literature. A data-centered categorization of wrangling operations abstracts away this terminology and will help

designers build better tools.

4 Descriptive Taxonomies: Actions and Processes

Our bottom-up taxonomies of data wrangling in computational journalism provide richly detailed descriptions of the *Actions (A)* journalists took while wrangling their data and our interpretation of their wrangling *Processes (P)*. Table 4.1 provides an overview of the whole taxonomy down to a depth of two. Unabridged versions of these taxonomies are provided in Supplemental Materials, with shortcode, name, and description for each code. Below, each code name is followed by the shortcode in parentheses to facilitate its lookup.

These two taxonomies thematically organize 165 open and axial codes. Both taxonomies are hierarchically organized into wide trees with a maximum depth of three. The *Actions (A)* taxonomies and *Process (P)* taxonomies contain 66 open codes with 25 axial codes and 56 open codes with 18 axial codes, respectively. The tree is root-justified not leaf-justified: open-code leaves may occur at any level.

4.1 High-Level Overview

Our taxonomy exists in two orthogonal and cross-cutting dimensions: wrangling *Actions (A)* performed on the data and descriptions of the wrangling *Process (P)*.

Actions	Process
Import	**Source**
Fetch	Collect Data
Create	Acquire Data
Load	**Workflow**
Clean	Annotations
Remove	Comp. Processes
Replace	Toggle Operation
Replace NA Values	**Cause**
Edit Values	Downstream Input
Resolve Entities	**Themes**
Standardize Cat Vars	Divide and Conquer
Scale Values	Join Aggregate
Reformat	Create a Frequency Table
Merge	Trim Fat
Union Datasets	Align Variables
Inner Join	**Analysis**
Supplement	Interpret Model
Cartesian Product	Compare Groups
Self Join Dataset	Identify Extreme Values
Profile	Show Trend Over Time
Run a Test	Calculate a Statistic
Check Results	Count the Data
Summarize Dataset	...
Derive	**Management**
Detrend	Object Persistence
Consolidate Variable Values	Data Quality
Generate Unique Identifiers	**Pain Points**
Subset the Dataset	Fix Incorrect Calculation
Formulate Perf Metric	Repetitive Code
Transform	Make an Incorrect Conclusion
Reshape	Post-Merge Clean Up
Modify Variables	Post-Aggregate Clean Up
Summarize	Data Too Large for Repo
Sort	Schema Drift
Export	Data Type Shyness

Table 1. Abridged version of our two descriptive taxonomies of data wrangling in computational journalism. The unabridged taxonomies extend to five levels, and are provided as supplemental materials.

Actions: We record seven high-level groups within the actions taxonomy. *Import (A.I)* captures how data is introduced to the wrangling environment. The *Clean (A.C)* branch addresses actions for well-known data quality issues, such as entity resolution, deduplication, and addressing missing or incomplete data. We record operations that combine multiple tables together under *Merge (A.M)*. Journalists often inspect the state of a table either before or after a transformation under the category *Profile (A.P)*. The *Derive (A.D)* branch contains codes for operations that transform observations and variables within a dataset, but do not necessary address data quality issues. Operations such as aggregating or reshaping a table fall under *Transform (A.T)*. Finally, journalists often *Export (A.E)* their data at the conclusion of the process.

Processes: Processes reflect our interpretations of the wrangling process. We record seven high-level descriptive categories reflecting our interpretations of the wrangling process. First, we take note of how journalists acquire data whenever such information is apparent or explicitly mentioned under *Source (P.S)* of the dataset. The category *Workflow (P.W)* captures user behavior as it relates to additional wrangling architecture within the environment. Although our analysis focuses on how data is wrangled, where we can confidently infer motives, we record these observations in the *Cause (P.C)* branch. We record several high-level patterns for how data is transformed in *Themes (P.T)*. While the scope of this project focuses on the pre-analysis activities of data journalists, we also include some observations about high-level *Analysis (P.A)*. Analyzing multiple notebooks reveals a few recurring methods for the *Management (P.M)*. Finally, we infer *Pain Points (P.P)* encountered during the wrangling process.

4.2 Low-Level Group Structure

The fourteen axial codes described above comprise only the top two levels of the taxonomies. Within each of these top levels, our high- and medium-level axial codes tend to converge upon high-level wrangling tasks addressed in related work, while our lower-level axial codes and open codes illustrate salient and nuanced differences within these well-known categories. For example, *Resolve Entities (A.C.b.3* refers to the common task of entity resolution, reconciling separate and distinct entries for the same real-world entity [3]. Other codes in the same group all address how users *Replace (A.C.b)* values in saliently different ways, such as: *Replacing NA Values (A.C.b.1)*, *Editing Values (A.C.b.2)*, and *Scaling Values (A.C.b.5)*. Likewise, *Replace (A.C.b)* belongs to a group of codes generally concerned with *Cleaning (A.C)* datasets of errors that also include *Remove (A.C.a)* and *Reformat (A.C.c)*.

5 A Multi-Table Framework for Data Wrangling

A key finding from the first two phases of our work is the discrepancy between journalists' frequent use of multiple tables and the single-table emphasis of most wrangling frameworks. We even found examples of journalists using multiple tables when wrangling a single data source. Many programming languages and packages support the concept of a table as a first-class object containing heterogeneous data, such as data frames in R and Pandas for Python. However, this convention is largely absent from GUI-based wrangling tools. Most interactive wrangling applications, such as Wrangler, OpenRefine, and Workbench, support only what we call a single-table wrangling context: the interface is designed around a single matrix where the table constitutes the environment, with rows and columns as the objects being wrangled within that environment. Motivated by the finding that wrangling across multiple tables is an unmet need, we present a concise multi-table framework for data wrangling. It features a small but complete set of operations that could serve as the formalism that underlies an interactive wrangling application.

5.1 Framework Overview

As we illustrate in Table 5.1, the design space of our framework is structured by two primary dimensions. The first dimension is the type of data object, with three different types: rows, columns, and tables. While tables are collections of rows and columns, we count them as distinct entities in order to describe table-based operations that operate on rows and columns collectively. The second dimension comprises operation categories, with five top-level classes: create, delete, transform, separate, and combine.

These five class groupings are based on the cardinality of the set of input and output objects according to three bins: zero, one, and many. Create has no inputs and one output; delete is the inverse. Transform is one-to-one. Separate has one input and many outputs, and

combine is the inverse. For simplicity, we do not include many-to-zero operations as these can be described as repeated one-to-zero operations. We also restrict operations to stay within data types. For example, a transformation operation that takes one table as input could not output two columns; it would output one table.

We designed this framework to be as concise as possible, and so that all intersections between the five top-level operation classes and the three data types would be semantically meaningful. The simple create and delete operations suffice for all three data types, and the other three categories have one level of further subdivision into two or three operations each, yielding 21 operations in total.

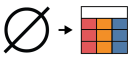
Op Class	Sets	Expanded Operations
Create	0:1	T/C/R: `Create`
Delete	1:0	T/C/R: `Delete`
Transform	1:1	T: `Rearrange`, `Reshape`; C/R: `Transform`
Separate	1:N	T: `Subset`, `Decompose`, `Split`; C/R: `Separate`
Combine	N:1	T: `Extend`, `Supplement`, `Match`; C: `Combine`; R: `Summarize`, `Interpolate`

Table 2. Multi-table framework for data wrangling. One axis of the design space is the type of data: Tables (T), Columns (C), and Rows (R). The 5 classes of operations comprise a second axis, based on cardinalities of the input and output sets: zero, one, and many.

We now describe each class of operations in the framework and discuss its connection to taxonomy classifications, database operations, and previous wrangling frameworks.

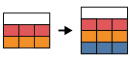
5.2 Create Operations

An operation that transforms zero objects into one or more is effectively creating data objects inside the wrangling environment. We consolidate both zero-to-one and zero-to-many operations into this class because we conceptually view the latter as repeating the former.



Create Tables: Users in our technical observation study defined tables in three distinct ways. First, tables can be *Fetched (A.I.a)* from an external source, such as a HTTP request to publicly accessible API. Second, tables can be *Created (A.I.b)* directly in the wrangling environment. Third, tables can be *loaded (A.I.c)* into the environment locally from a file or database residing on the user's hard drive.

Create Columns: While new columns can be added to a table by merging another table or transforming existing columns within a table, column creation involves adding columns to tables without these sources. We observed a salient instance in our technical observation study: the code *Generate Dataset Identification (A.D.c.2)* describes when a journalist defines a column of constant values, such as the year of the data or the file name. This dataset variable serves as a unique identifier for the table, and this action often occurs prior to merging tables together row-wise, *Union Datasets (A.M.a)*.

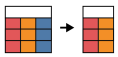


Create Rows: While row creation is not a common class of data wrangling operations, users may need to do so in order to enter missing observations from a dataset obtained by other means, which we coded as *Construct Data Manually (A.I.b.1)*.

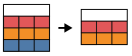
5.3 Delete Operations

Functions that map one or more objects to zero objects.

Delete Tables: Tables can be deleted either explicitly or implicitly. When merging multiple tables together, constituent tables may be explicitly removed after the operation to clean up the wrangling environment. In a wrangling environment where tables are first-class objects, filtering a table by rows and columns can be conceptualized as a composite task in which they `Separate` one table into two tables and implicitly `Delete` either one.



Delete Columns: One-to-zero operations with columns deleted from the table [18]. Columns may be irrelevant, incomplete, or duplicate variables. Journalists in our technical observation study frequently *Removed Variables (A.D.d.1)*. *Merging (A.M)* datasets together may result in duplicate variables, and journalists may choose to *Remove Duplicate Variables (P.P.d.4)*. Journalists also *Trim Fat (P.T.d)*, removing many variables at the initial portion of wrangling. Many interactive wrangling applications [15, 43, 46] support dropping multiple columns at a time. However, we omit a many-to-zero column operation category since this action is conceptually a composite of many one-to-zero column operations.

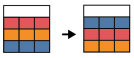


Delete Rows: Maps one or more rows to zero rows [18]. It is an essential class of wrangling operations in data preparation tasks, such as filtering. As with columns, deletion is a means of addressing irrelevant, incomplete, or duplicate observations in a dataset.

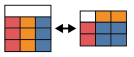
5.4 Transform Operations

Functions involving a one-to-one mapping of inputs and outputs.

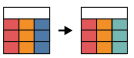
Transform Tables: A rich class of operations involving structural changes to a table of varying complexity. On the simple end of the spectrum is `Rearrange`, or operations that transform the table without modifying the fundamental structure of the dataset, and on the complex end of the spectrum is `Reshape`, operations that modifying the fundamental schema of the dataset.



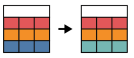
`Rearrange` operations transform a table without fundamentally modifying the underlying table schema. Example operations in this category are *sort* [18] and rearrange columns. While less semantically meaningful as rearranging rows, some informal conventions may facilitate presentation of a dataset, such as moving unique identifiers of an observation to the far left side of a table.



`Reshape` operations change the fundamental structure of variables and observations within the dataset [18]. *Unfold* and *fold* are two examples of `reshape` operations. *Fold* collapses a column into key-value sets, which can be used to restructure a table into *tidy format* [51]. *Unfold* casts level values in columns of categorical variables as table columns from data values, a common method to *Cross Tabulate (A.T.a.2)* data.



Transform Columns: Map the values from one column into another column by various means, including extract, cut, and split [18]. Transformation can be a means to a multitude of data wrangling ends. For example, *Resolving Entities (A.C.b.3)*, the process of reconciling separate and distinct entries for the same real-world entity [3], can involve mapping a column of categorical variables with duplicate levels for the same real-world entity to a smaller, unique set of levels.



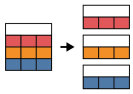
Transform Rows: Transform one row into another because data is either incomplete or inaccurate. We observed journalists manually editing individual rows when raw data contained clerical errors, coded as *Impute Missing Data (A.I.b.4)*. Errors arising from human data entry are a common issue for journalists [10].

5.5 Separate Operations

Transform one object in the wrangling environment into many, by which we mean strictly greater than one. While our framework has parallels to operations for combining tables using database tools, these tools lack high-level support for separating tables by value.

Separate Tables: Our framework is the first to address operations in this category separating one table into many. We classify operation in the **Separate** category into three subcategories: `Subset`, `Decompose`, and `Split`.

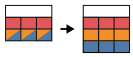
`Subset` operations describe dividing a table row-wise into two tables. Filtering a table is a composite task combining `Subset` and `Delete`. After separating a table, the subset of non-matching rows is deleted from the wrangling environment.



`Decompose` operations are similar to `Subset` except a single table is partitioned into any number of constituent tables based on the values and data type of a single table column. If this column represents a categorical variable, then constituent tables are divided by unique level value. In the case of boolean variables, this operation outputs two tables for true and false values. If the variable is quantitative or ordinal, the constituent table is arbitrarily divided into disjoint sets within the variable's range.

`Split` operations describe mapping one table to many by dividing a table column-wise. While these operations essentially produce two sets of disjoint columns, one duplicate key column always remains in order to maintain continuity between observations in the two tables.

Separate Columns: Separate a composite column into its atomic components, a necessary step for anomaly detection [31]. One of the most common issues with messy data is for multiple dataset variables to be stored in a single column [51].

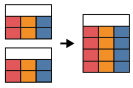


Separate Rows: Separating one row into many can address more fundamental data parsing issues that arise in data wrangling [18]. OpenRefine [15] enables users to work with observations in multiple rows as *records* in the application. This class of operations could be useful for initially parsing data stored in files with idiosyncratic structuring.

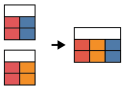
5.6 Combine Operations

Many-to-one transformations of objects in the wrangling environment effectively combine objects.

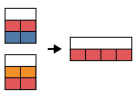
Combine tables: Data integration, constructing a single schema for unified access to multiple data sources [40], is often cited as a sub-process of data wrangling [4, 17, 18], but details of transformations during integration are minimally addressed in existing process theories. Informed by the results of our technical observation study, we offer three subcategories of operations that combine multiple tables into one: `Extend`, `Supplement`, and `Match`.



`Extend` operations describe the row-wise merger of multiple tables into one table, similar to a `UNION` operator in SQL. These operations can be significantly complicated by *Schema Drift (P.P.g)*, in which periodically published datasets change over time. While enterprise data analysts have been documented encountering this issue through redundant columns containing the same variable, our technical observations show that the levels of categorical variables often also change over time.



`Supplement` operations incorporate data from other tables through the column-wise merger of multiple tables where tables are matched on corresponding key columns [18], similar to an `OUTER JOIN` operation in SQL. There is a bijective relationship between levels in the key column of the main table and the supplementing table, distinguishing it from `Match`. One common application of supplement involves *Creating Lookup Tables (P.A.m)*. Hence, using a lookup table can be a table-focused way of transforming columns.



`Match` operations also concern the column-wise combination of many tables into one table; however, there exists an injective relationship between the matching keys of the two tables similar to an *INNER JOIN* operation in SQL. Hence, some rows purposefully do not have corresponding matches, excluding them from the output table, also known as filter joins [52]. We do not specify an inverse operation for match because rows are excluded from this process by definition. To `Match` tables when the intent was to `Supplement` can cause rows to be dropped from the output table unknowingly. We code this phenomenon as a *Lossy Join (P.P.d.3)*.

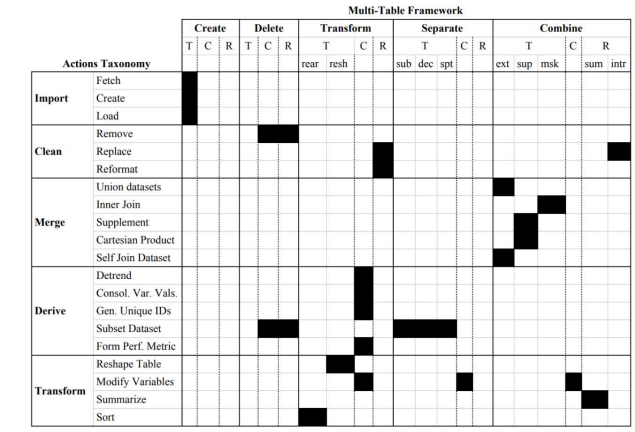
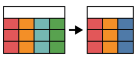


Fig. 3. We cross check the descriptive power of our multi-table framework for data wrangling by comparing against the high-level axial codes in our descriptive action taxonomy. We only include Actions codes that correspond with table operations, excluding codes in the Profile branch.



Combine Columns: Map many columns into one column. While a column containing a decomposed variable, such as an address, may need to be separated in order to perform operations on its constituent components, the data wrangler may prefer for these components to be *combined* into a single column. This example is essentially a *split, compute, and merge* strategy for data cleaning [50] applied to wrangling.

Combine Rows: Take multiple rows as input, then output one row. We make a distinction between two combine operations within this category: `Summarize` and `Interpolate`.

`Summarize` operations combine rows, grouping observations by a categorical variable and applying an aggregate function to another variable. Aggregation effectively coarsens the granularity of the observations in a dataset. While this operation could be construed as a one-to-one transformation of tables, categorizing it as an operation upon rows better captures this coarsening effect.

Many data analysis packages support aggregations grouped by the levels of one or more categorical variables. Aggregation alone does not completely describe this common use case. We consider this particular operation a composite operation consisting of three separate operations, similar to the Split, Apply, Combine strategy for data analysis [50]. The same operation can be described in this framework by `Decompose`, `Summarize`, and `Extend`.

`Interpolate` operations are another common way to combine rows. Also known as *fill* [18], one output row is calculated based on the values of multiple input rows. This operation can be used to address issues with missing data.

5.7 Assessment

We developed this second model for data wrangling in pursuit of greater generative power. The composability and conciseness of this 21-operation framework the 165-code set of descriptive taxonomies reasonably indicates our success. Individual operations can be arranged to create new multi-table wrangling processes and describe existing ones. For example, the popular Split, Apply, and Combine strategy of data analysis [50] can be translated into a sequence consisting of table separation, column transformation, and table combining operations. However, our framework preserves the multi-table context of the "split" and "combine" operations and the within-table context of the "apply" operations. Figure 3 illustrates similar areas of descriptive overlap be-

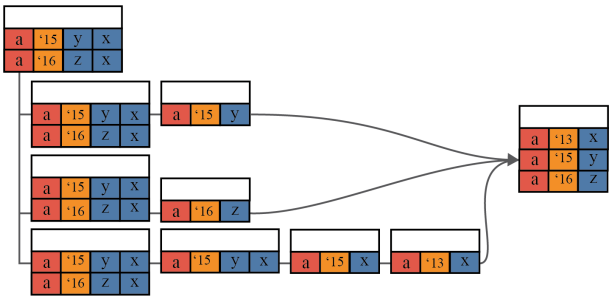


Fig. 4. Journalists at the Los Angeles Times employ multiple tables to wrangle water usage data into tidy format. With water usage amounts in a separate column, common reshape operations that operate within the context of one table fail on this table.

tween these two conceptual models. We cross-checked them to ensure our multi-table framework adequately covers wrangling observations performed by journalists during our technical observation study.

6 Critical Incidents

This section presents two case studies taken from the corpus of 50 repos to highlight specific critical incidents. The first incident illustrates how journalists overcome challenging data wrangling tasks that cannot be accomplished with existing interactive wrangling systems. The second incident demonstrates that common mistakes when wrangling data with multiple data sources may lead to factual errors in published stories. We explain how we arrived at some of the open codes in the taxonomies.

6.1 A Multi-Table Success Story

The state of California enacted mandatory water use restrictions from 2015 to 2016, following years of record drought. *Using open-government data portals (P.S.b.5)* published monthly by California's Water Resources Control Board, reporters at the Los Angeles Times found the majority of water districts increased their usage after the state relaxed restrictions in 2016 when compared to water use in 2013, before the governor proclaimed a state of emergency due to drought.

In "The drought eased up, and these Californians turned on the spigot" reporters analyze this data and rank water supply agencies by a conservation-consumption score, a composite metric considering overall water savings and per-capita water use. Reporters specifically *Compare Groups (P.A.b)* along a common metric to evaluate water usage in June, July, and August between 2013, 2015, and 2016.

One significant wrangling challenge involved tidying [51] the data. Figure 4 illustrates a simplified representation of the data schema at the beginning of this process, the state of intermediate representations, and the final tidy format. The dataset includes the following variables: supplier name, month and year of the water usage, and the amount of water used. This abstract dataset could easily lend itself to a three-column table. However, the actual table representation has a fourth column: the amount of water consumed in 2013. The most complicated form of messy data has different variables stored in both rows and columns, such as a cross-tabulated table [51]. In this example, the same variable (year of recorded water production) exists in both rows and columns.

In order to tidy this dataset, the reporters implement a *Divide and Conquer (P.T.a)* strategy, splitting this table into three tables, then filtering the table upon a facet. With values for 2013 in a separate table, journalists were able to apply typical wrangling operations common in single-table contexts. A new date column can be derived from the old one, unilaterally replacing any year with 2013. The column containing non-2013 values was split from the rest of the table; as was the 2013 column in the other two tables. Finally, the three constituent tables were extended together into one tidy table.

Wrangling applications that operate in a single-table context, would fail on this transformation. A logical within-table approach for this type of problem would be to *Transform (A.T)* the schema via a *Reshape (A.T.a)* operation. Reshaping generally captures transposing sections of the data from rows to columns, and vice versa. We observed many journalists performing this through *melt* and *cast* in the reshape library in R. This operation is also supported in both OpenRefine under *Transpose*, and in Google Cloud Dataprep under *pivot*, *unpivot*, and *convert columns to values*. In either approach, there would not be a straightforward way to fold 2013 values into the other columns.

6.2 A Multi-Table Cautionary Tale

On November 19, 2015, BuzzFeed News published the article "Where U.S. Refugees Come From—And Go—In Charts." This piece presents the results of exploratory data analysis on a dataset related to refugee immigration to the United States from 2005 to 2015, with totals greater than 672,000 refugees. This data comes from the data portal for the US State Department's Refugee Processing Center. Like the other case study, the raw data is from an *Open-government Data Portal (P.S.b.5)*.

Because each table represents the same phenomenon, the sum of arrivals should be equal. Journalist begins with *Testing for Equality (A.P.a.2)* by summing the arrivals column of both tables and comparing the total. The religion table had four more arrivals than the destination table. No further steps were taken to address this discrepancy, coded as *Tolerate Dirty Data (P.M.b.2)*. However, our own auditing of the data reveals more nuanced differences between the two tables. When comparing arrival counts by country, the destination table has one more arrival from Iran and five fewer arrivals from Iraq than the religion table, leading to a discrepancy of six arrivals between the two tables.

Although this difference is insignificant, in theory the grouped differences between arrival totals could have been arbitrarily large and the non-grouped difference could appear small. Thus, simple column sums are a problematic way to *Test for Equality (A.P.a.2)* between two tables.

The majority of wrangling in this notebook involves transforming each table to make it suitable for different visualizations, *Wrangle Data for Graphics (P.C.a.1)*, including total arrivals over time, arrivals from a specific country, and arrivals by destination in the US. Like the previous case study, the analysis of this notebook revolves around visually *comparing groups (P.A.b)*. The final chart in this notebook compares US states by refugees admitted, normalized per 1,000 residents. The wrangling portion of this task required introducing a third table of the US state populations to the wrangling environment in order to *formulate a performance metric (A.D.e)* for a fair comparison, the normalized rate of arrivals per state population. To form this derived table, the US state population and refugee arrival datasets were merged by a *Outer Joining (A.M.c.1)* of tables. Our multi-table framework captures this kind of table transformation under *Supplement (A.M.c)*, combining two tables that significantly increases the number of columns while rows remain unchanged.

However, the number of rows from the resulting composite table was significant in this instance. The state of Wyoming was missing from the chart when the article was first published. Later that day, the state was added to the chart and a correction was issued. Because the journalist published the code underlying this article, we know that this error resulted from a subtle issue when performing outer joins between two tables. Wyoming did not have a column in the aggregated refugee arrival table because the state did not accept any refugees between 2005 and 2015. Hence, when the state population table was outer-joined to the refugee table, Wyoming was silently dropped because the corresponding key did not exist in the other table.

The Observations branch of the taxonomy contains many observations of *Pain Points (P.P)* from our technical observation study. One frequent pain point concerned data cleaning tasks that result from merging multiple datasets, *Post-merge Clean Up (P.P.d)*. This issue of silently dropping the state of Wyoming was coded as a *Lossy Join (P.P.d.3)*, when data is lost after merging two tables column wise. While the implications of this error were relatively minor in the context of the whole article, this case illustrates that issues in data wrangling can have real editorial consequences for journalists.

7 Implications for Visualization Design

Visualization is an integral component of interactive wrangling, bridging the gulf of evaluation [28] by communicating the effects of intended

operations on a dataset. Our results have design implications for visualization, including how to develop interactive interfaces, address common pain points, communicate data provenance, and recommend transformations via mixed-initiative guidance.

The clearest need is to construct an interactive visualization-based wrangling application that fully instantiates all of the operations in our multi-table framework. Although many applications support subsets, none covers them all. Combining tables is widely supported, similar to *JOIN* operations in SQL, but no application fully supports operations for separation, which would help resolve the common problem of one table with multiple types of observations [51]. Our multi-table framework specifies an underlying formalism, but interaction design is a remaining challenge. An approach similar in spirit to Polaris [41] would be suitable, with a bidirectional mapping between a formalism and actions carried out through drag-and-drop visual interface.

Through code comments authored by journalists and inferences based on our own experience wrangling data, we observed many pain points that are amenable to a visualization-based solution. Visualization could aid journalists in detecting *Schema Drift (P.P.g)* when wrangling perennially published datasets, which includes changes in dataset variable names and values over time. Combining tables may introduce errors in the wrangling process, which wranglers further address as *Post-Merge Clean Up (P.P.d)*. In our second case study, a political map of the United States could have alerted the wrangler that their exported dataset was missing a state. In general, visualization paired with semantic and accurately inferred variable types can alert wranglers to missing data in common geographic and temporal variables.

Data provenance is more complicated in multi-table wrangling processes, and visualizations of these processes as data-flow diagrams can concisely depict these complex flow networks. A major visualization challenge for the few interactive wrangling applications that incorporate this idiom [45, 46] and any future multi-table wrangling tool, is designing automatic layout algorithms to draw comprehensible data-flow diagrams for workflows as complicated as we observed among journalists. These workflows possess many qualities that contribute to visual complexity: cyclic processes, multiple sinks, dozens of sources, and even more interior nodes.

Finally, a mixed-initiative system could facilitate and expedite wrangling through automatically generated recommendations, where the user could visually preview data transformations before interactively selecting which ones are desirable. Although Wrangler [18] provides this capability for single-table wrangling, multi-table support would require not only the improved provenance diagrams mentioned above, but also the ability to show effects across multiple tables, possibly with semantic zooming or other multi-scale approaches [20].

8 Discussion

Many interactive wrangling applications specify a design space of supported transformations informed by data transformation languages [18] or personal experience in wrangling data [4]. To the best of our knowledge, no one has generated a design space of data transformations grounded in observational data gathered from users wrangling data in the wild, with the full flexibility of programming through script-based languages. We conjectured that these programmatic approaches to wrangling might be more expressive than those supported by interactive wrangling applications, and indeed we found these differences. We were especially interested in patterns of behavior where users appear to be exerting a lot of effort to accomplish a relatively simple task. Such discrepancies between the level of effort and the simplicity of the task can signal deficiencies in a particular model of wrangling.

A better understanding of journalistic data wrangling holds considerable promise for positive social impact. Journalism provides a public good, especially through investigative and public affairs reporting that uses data to hold corporations, public institutions, and elected officials accountable [13]. Better support for the time-consuming, error-prone practices of data wrangling could lead to tools that better support journalists in conducting this socially beneficial work. Even as data-driven journalism grows more important in many newsrooms, layoffs have diminished their on-the-ground reporting capacity: staff at US newspapers declined 47 percent from 2008 to 2018 [9], but better data analysis tools can help news organizations meet the unfortunate demand to do more with less. Visual analytics researchers need to target journalists as an often-overlooked group who specifically needs better tools. When journalists are lumped in with non-technical users, two problems emerge. First, we ignore the growing number of computational journalists who are quite technical and use data in their daily work. Second, tools built for the general user may not be as effective for journalists if they do not match the domain need.

While we thoroughly searched for relevant repos, our data collection method is still inherently biased towards completed projects that yielded newsworthy findings. Thus, we do not claim to offer complete coverage of wrangling activities in journalism. In future work, we plan to explore instances of unsuccessful wrangling through an interview study.

Although excluding failures is a limitation, our corpus of coded notebooks still contains telling instances of unsuccessful wrangling. Success in data wrangling does not solely depend on coercing data into an acceptable state of utility. Expending a reasonable amount of time and effort in relation to the complexity of the task is also an essential criterion for success. Both critical incidents illustrate varying degrees of failure in data wrangling despite producing data for further analysis. Our second critical incident illustrates how data in unacceptable state resulted in real editorial consequences. Journalists in our first example were able to wrangle raw data into a useful state. However, constructing seven intermediate data products is an unreasonable amount of effort for tidying a table. A wrangling process that comprises up to 80-90% of an analyst's time [7, 25] should not be considered successful. Usability metrics, such as time to completion, ought to be an essential measure of success when evaluating interactive wrangling applications, but these have only been included in a few previous system evaluations [18].

We observe that the raw data in many of our repos is significantly better structured than previous work examples, so some amount of wrangling may have been performed prior to the journalists' actions. Our observational data may reflect a *last-mile* problem in data wrangling, where raw data comes in a structured format, but requires further tweaking to match the user's mental model. How journalists operate on data in the last mile may correlate with different types of data journalism story. The *Analysis (P.A)* branch of our Process Taxonomy shares some salient similarities with taxonomies of data journalism stories [48], including *Compare Groups (P.A.b)*, *Show Trend Over Time (P.A.d)*, and *Identify Extreme Values (P.A.c)*. For example, *Comparing Groups* often involves several critical operations in the Actions branch: *Generating Unique Identifiers (A.D.c)* for each group, *Formulating a Performance Metric (A.D.e)* by which to fairly compare groups, and *Summarizing (A.T.c)* this derived data. An interactive wrangling application designed with awareness of common, high-level analysis goals could further expedite the wrangling process for journalists.

9 Conclusion

Our paper answers three questions. First, how do journalists wrangle their data? By employing a technical observation study, we produce two richly detailed descriptive taxonomies of data wrangling in computational journalism to answer this question. Although we find that journalists perform many familiar wrangling tasks that are well supported by previous wrangling frameworks, they need far more support wrangling involving multiple tables than previous work acknowledged. Second, do journalist behaviors and needs match up with existing literature on data wrangling? We find that previous process frameworks for wrangling do not incorporate the concept of a table as a first-class object in the environment. Third, how can we re-characterize wrangling to match the needs of journalists? We present a concise framework that describes required operations to support multi-table wrangling, with actionable generative power that could support future interactive tools.

Acknowledgments

We thank Jürgen Bernard, Anamaria Crisan, Madison Elliott, Zipeng Liu, Joanna McGrenere, Francis Nguyen, Michael Oppermann, and Ben Shneiderman for their formative feedback. This work was supported by NSERC CREATE 386138851 and Discovery RGPIN-2014-06309.

REFERENCES

[1] C. W. Anderson. *Apostles of Certainty: Data Journalism and the Politics of Doubt*. Oxford University Press, 2018.

[2] M. Beaudouin-Lafon. Designing Interaction, not Interfaces. In *Proceedings of the working conference on Advanced visual interfaces - AVI '04*, page 15, Gallipoli, Italy, 2004. ACM Press.

[3] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. Swoosh: A Generic Approach to Entity Resolution. *The Very Large Data Bases Journal (VLDB)*, 18(1):255–276, Jan. 2009.

[4] A. Bigelow, C. Nobre, M. Meyer, and A. Lex. Origraph: Interactive Network Wrangling. In *Proc. IEEE Symp. Visual Analytics Science and Technology (VAST)*, pages 81–92, 2019.

[5] S. Chinoy, I. Lee, B. Welsh, and A. Williams. First Observable Notebook: Prototyping with Polish, Mar. 2019. https://www.ire.org/events-and-training/event/3433/4109.

[6] S. Cohen, J. T. Hamilton, and F. Turner. Computational Journalism. *Communications of the ACM*, 54(10):66, Oct. 2011.

[7] T. Dasu and T. Johnson. *Exploratory Data Mining and Data Cleaning*. John Wiley & Sons, 2003.

[8] L. Given, editor. *The SAGE Encyclopedia of Qualitative Research Methods*. SAGE Publications, 2008.

[9] E. Grieco. U.S. Newsroom Employment has Dropped a Quarter Since 2008, with Greatest Decline at Newspapers, July 2019. https://www.pewresearch.org/fact-tank/2019/07/09/u-s-newsroom-employment-has-dropped-by-a-quarter-since-2008.

[10] C. Groskopf. The Quartz Guide to Bad Data, Nov. 2015. https://github.com/Quartz/bad-data-guide.

[11] C. Groskopf. Agate, Mar. 2018. https://github.com/wireservice/agate.

[12] C. Groskopf, J. Germuska, A. Bycoffe, and T. Mehlinger. csvkit, Feb. 2012. https://github.com/wireservice/csvkit.

[13] J. Hamilton. *Democracy's Detectives: The Economics of Investigative Journalism*. Harvard University Press, 2018.

[14] J. Heer and A. Perer. Orion: A System for Modeling, Transformation and Visualization of Multidimensional Heterogeneous Networks. In *Proc. IEEE Visual Analytics Science and Technology (VAST)*, pages 49–59, Oct. 2011.

[15] D. Huynh. OpenRefine, Oct. 2012. https://openrefine.org.

[16] D. F. Huynh, R. C. Miller, and D. R. Karger. Potluck: Data Mash-Up Tool for Casual Users. *Journal of Web Semantics*, 6(4):274–282, Nov. 2008.

[17] S. Kandel et al. Research Directions in Data Wrangling: Visualizations and Transformations for Usable and Credible Data. *Information Visualization*, 10(4):271–288, Oct. 2011.

[18] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: Interactive Visual Specification of Data Transformation Scripts. In *Proc. SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pages 3363–3372, May 2011.

[19] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. Enterprise Data Analysis and Visualization: An Interview Study. *IEEE Trans. Visualization and Computer Graphics (TVCG)*, 18(12):2917–2926, Dec. 2012.

[20] H. Lam and T. Munzner. *A Guide to Visual Multi-Level Interface Design from Synthesis of Empirical Study Evidence*. Morgan Claypool Synthesis Lectures on Visualization, 2010.

[21] J. Lin et al. End-User Programming of Mashups with Vegemite. In *Proc. Conf. on Intelligent User Interfaces (IUI)*, pages 97–106, Feb. 2009.

[22] J. Liu, N. Boukhelifa, and J. R. Eagan. Understanding the Role of Alternatives in Data Analysis Practices. *IEEE Trans. Visualization and Computer Graphics (TVCG)*, 26(1):66–76, 2020.

[23] Z. Liu, S. B. Navathe, and J. T. Stasko. Network-Based Visual Analysis of Tabular Data. In *Proc. IEEE Symp. Visual Analytics Science and Technology (VAST)*, pages 41–50, Oct. 2011.

[24] M. Meyer and J. Dykes. Criteria for Rigor in Visualization Design Study. *IEEE Trans. Visualization and Computer Graphics (TVCG)*, 26(1):87–97, 2020.

[25] M. Muller et al. How Data Science Workers Work with Data: Discovery, Capture, Curation, Design, Creation. In *Proc. Conf. on Human Factors in Computing Systems (CHI)*, pages 1–15, May 2019.

[26] G. I. J. Network. Nils Mulvad - Excel is 90% of data journalism - YouTube. https://www.youtube.com/watch?v=aahUKhuB9Bw.

[27] D. Nguyen. Dollars for Docs: Using Google Refine to Clean Messy Data, Oct. 2010. https://www.propublica.org/nerds/using-google-refine-for-data-cleaning.

[28] D. A. Norman. *The Design of Everyday Things*. Doubleday/Currency, 1990.

[29] E. D. Ragan, A. Endert, J. Sanyal, and J. Chen. Characterizing Provenance in Visualization and Data Analysis: An Organizational Framework of Provenance Types and Purposes. *IEEE Trans. Visualization and Computer Graphics (TVCG)*, 22(1):31–40, Jan. 2016.

[30] P. Ralph. Toward Methodological Guidelines for Process Theories and Taxonomies in Software Engineering. *IEEE Trans. on Software Engineering*, 45(7):712–735, 2019.

[31] V. Raman and J. M. Hellerstein. Potter's Wheel: An Interactive Data Cleaning System. In *Proc. International Conference on Very Large Data Bases (VLDB)*, pages 381–390, 2001.

[32] J. Reback, W. McKinney, and et al. pandas-dev/pandas: Pandas 1.0.3, 2020. https://doi.org/10.5281/zenodo.3509134.

[33] S. Rogers. Data journalism at the Guardian: What is it and how do we do it?, July 2011. https://www.theguardian.com/news/datablog/2011/jul/28/data-journalism.

[34] S. Rogers. Data Journalism Broken Down: What We Do to the Data Before You See It, Jan. 2013. https://www.theguardian.com/news/datablog/2011/apr/07/data-journalism-workflow.

[35] S. Rogers, J. Schwabish, and D. Bowers. Data Journalism in 2017: The Current State and Challenges Facing the Field Today, Sept. 2017. https://newslab.withgoogle.com/assets/docs/data-journalism-in-2017.pdf.

[36] M. Schudson. *Discovering the News: A Social History of American Newspapers*. Basic Books, 1978.

[37] H. Sharp, J. Preece, and Y. Rogers. *Interaction Design: Beyond Human-Computer Interaction*. Wiley, May 2019.

[38] Y. L. Simmhan et al. Performance Evaluation of the Karma Provenance Framework for Scientific Workflows. In *Proc. International Provenance and Annotation Workshop (IPAW)*, volume 4145 of *Lecture Notes in Computer Science*, pages 222–236. Springer, 2006.

[39] Y. L. Simmhan, B. Plale, and D. Gannon. A Survey of Data Provenance in E-Science. *ACM SIGMOD Record*, 34(3):31–36, Sept. 2005.

[40] J. M. Smith, P. A. Bernstein, U. Dayal, N. Goodman, T. Landers, K. W. T. Lin, and E. Wong. Multibase: Integrating Heterogeneous Distributed Database Systems. In *Proceedings of the May 4-7, 1981, national computer conference*, AFIPS '81, pages 487–499, Chicago, Illinois, May 1981. Association for Computing Machinery.

[41] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65, Jan. 2002.

[42] J. Stray. Network Analysis in Journalism: Practices and Possibilities. In *Proc. Knowledge Discovery in Databases (KDD)*, Aug. 2017.

[43] J. Stray et al. Workbench, 2018. https://workbenchdata.com.

[44] S. Sunne. Diving into Data Journalism: Strategies for your newsroom, Mar. 2016. https://www.americanpressinstitute.org/publications/reports/strategy-studies/data-journalism.

[45] Tableau Software. Tableau Prep Builder, 2019. https://www.tableau.com/products/prep.

[46] Trifacta. Trifacta Wrangler, 2012. https://www.trifacta.com/products/wrangler-editions/#wrangler.

[47] R. Tuchinda, P. Szekely, and C. A. Knoblock. Building Mashups by Example. In *Proc. Intelligent User Interfaces (IUI)*, pages 139–149, Jan. 2008.

[48] A. Veglis and C. Bratsas. Towards A Taxonomy of Data Journalism. *Journal of Media Critiques*, 3(11):109–121, Sept. 2017. http://mediacritiques.net/index.php/jmc/article/view/158.

[49] B. Welsh. What is a Data Desk? PyData LA Keynote, Nov. 2018. https://www.youtube.com/watch?v=_73obGWno1.

[50] H. Wickham. The Split-Apply-Combine Strategy for Data Analysis. *Journal of Statistical Software*, 40(1):1–29, Apr. 2011.

[51] H. Wickham. Tidy Data. *Journal of Statistical Software*, 59(1):1–23, Sept. 2014.

[52] H. Wickham, R. François, L. Henry, and K. Müller. dplyr: A Grammar of Data Manipulation, June 2019. https://dplyr.tidyverse.org.

[53] H. Wickham and L. Henry. tidyr: Tidy Messy Data, Sept. 2019. https://tidyr.tidyverse.org.

[54] J. Wong and J. I. Hong. Making Mashups with Marmite: Towards End-User Programming for the Web. In *Proc. Conf. on Human Factors in Computing Systems (CHI)*, pages 1435–1444, Apr. 2007.

[55] Workbench. Data Journalism Made Easier, Faster, and More Collaborative, July 2018. https://link.medium.com/V8t8ejsyz8.