Table Scraps: Supplemental Materials

1. Detailed Process Description

This section is a detailed description of our process that provides more detailed methodological descriptions than what is present in the manuscript. Citations refer to the manuscript bibliography.

1.1 Phase One: Qualitative Coding Study Overview

In the first phase we address Q1: *What are the wrangling practices of journalists?*

We performed qualitative coding of the programming scripts and computational notebooks supporting published articles authored by data-literate journalists with programming skills. To do this, we systematically searched GitHub and ObservableHQ to identify an initial pool of more than 1,000 code repositories related to journalism. From this initial pool we manually inspected each repository to identify those containing data analysis, resulting in a set of 225 repositories.

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

Studies on provenance in e-science make the distinction between whether data provenance records are data or process oriented [26, 37, 38]. We also make the distinction between data and process in the qualitative coding portion of phase 1. 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.

1.1.1 Observation of Technical Artifacts

We employ the term technical observation study from the software engineering literature to denote a data collection strategy where the authors observe user-generated technical artifacts, such as source code [29].

This approach is similar to indirect observation, as both methods involve mediated, post hoc analysis of the user group. Data collection in indirect observation is instrumented through researcher-developed tools, such as keystroke logs or transcriptions of audio and video recordings [36]. However, data in technical observation is an artifact of the phenomenon itself.

Wrangling is particularly 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 of data wrangling 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 as a qualitative methodological approach 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. In this paper, we limit our claims to focus on how data wrangling occurs within this user group, with limited conjecture into why journalists are performing these actions. While code comments and the state of a table before and after a transformation provide some sense of the user's motivation, qualitative research methods such as interviews and direct observation would be better suited to untangling this question in greater depth [36]. 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 forms, and may omit false starts and dead ends. Moreover, our repo collection process is designed to filter out all instances of unsuccessful wrangling. Our study may thus paint a more simplistic view of wrangling than the reality.

1.1.2 Repository Selection

We compiled an initial pool of 1,301 public code repositories on GitHub[1] and ObservableHQ[2] 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.

On GitHub, we identified more than 1,000 journalistic repos through two avenues. We conducted a programmatic search through the platform's Search API, parameterized by topic, owner, and programming language. We satisfy the relevance to journalism criteria as those repos where the author has added the journalism or data journalism topic tag. Also, we use the Twitter bot @NewsNerdRepos that has been monitoring new repos on the platform published by journalists, and consider that any GitHub user or organization monitored by this account[3] satisfies the journalism-relevance criterion. We satisfy the wrangling language criteria 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. Although some previous work discusses wrangling via programming scripts written in Perl [18], currently R and Python are much more popular contemporary scripting languages among data journalists. 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 thus manually inspected the contents of each repo to exclude irrelevant ones, yielding 225 repos.

[1] www.github.com
[2] www.observablehq.com
[3] www.github.com/silva-shih/open-journalism

ObservableHQ is a computational notebook environment similar to Jupyter, where Observable notebooks are created using the front-end web development tools of HTML, CSS, and JavaScript. We included a total of 36 repos from three journalists at major metropolitan daily newspapers who were panel members of a NICAR 2019 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 pool of coded repos. After similar manual filtering, we retained 34 repos from this set.

We include this curated corpus of 225 journalistic repos containing data analysis in supplemental materials. In addition to the targeted languages of Python, R, and R Markdown, this corpus also serendipitously includes wrangling in Bash using csvkit.

1.1.3 Qualitative Coding

After selecting a repo to annotate from the curated pool of repos we exported all repo material to PDF, including scripts, Jupyter notebooks, and markdown files, and articles. Exporting all materials to one file format gave us a unified platform for applying codes that is programmatically accessible. Next, we open-coded these materials through PDF comments using Adobe Acrobat DC; only the sections related to data wrangling were coded. We maintained the codeset as a separate YAML file containing all unique open codes, updating the YAML file with the code name and description when the coder decided that the observed phenomenon warranted the development of a new open code. We performed axial coding in batches, splitting and consolidating code groups as needed, and we used YAML's nested syntax to designated axial code groups in the codeset.

All repos were coded by a single researcher, the first author. We repeated this process until saturation, where we deemed the codeset to adequately describe the phenomena we encountered; we reached this point at 50 repos. As part of deciding when we had reached saturation, we also continually checked the number of unique codes in the codeset against the number of repos coded, as shown in Supp. Figure S1.

Throughout the coding process, we opportunistically applied new open codes to previously coded notebooks when the coder spontaneously recalled applicable segments. After coding 25 notebooks, the coder systematically re-coded the previous 25 with the codeset at that state. After saturation at 50 repos, the coder systematically re-coded all repos according to the final codeset. In the subsequent phase, we modified the internal structure of our axial codes to better align with existing literature. Open codes remained effectively unchanged, except for modifying names and descriptions for clarity.

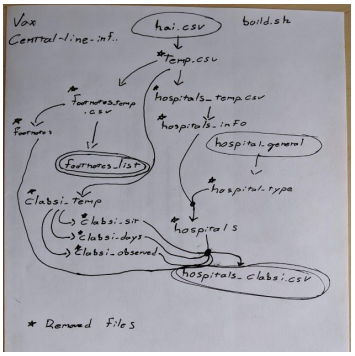
From the curated corpus, we iteratively chose one repo at a time to annotate through a manual selection procedure using two inclusion criteria. First, we checked that the repo was connected to a published article, so that our taxonomy is based only on repos where journalists were successful in wrangling raw data for their analysis or visualization needs. Second, we aimed for a mix of diversity of journalistic experience by considering both individual users and the news organizations they are affiliated with. For individuals, we measured their level of activity within GitHub using its API to obtain a count of git commits they made across all repos, see Supp. Fig. S4. We also considered the individual's prominence

within the computational journalism community, a subjective and qualitative assessment using our domain knowledge, see Supp. Fig. S5.. For organizations, we similarly considered both the quantitative measure of how many repos existed within the pool (see Supp. Fig S2), and our qualitative assessment of the organization's prominence by how many commits its members make (see Supp. Fig S3). We strove to deliberately select a mix of repos, to ensure coverage of both well-known data journalists and organizations who publish code and data to GitHub frequently, as well as lesser-known journalists and news outlets. The final corpus of 50 coded notebooks contains work from 33 journalists at 26 news organizations. Supp. Figures S2 through S5 show the breakdown of this final corpus by journalist and news organization.

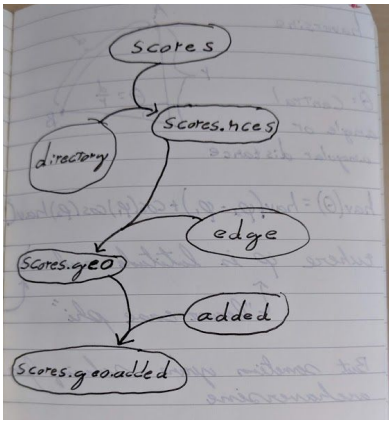
We created Jupyter Notebooks to support the repo selection process with summary plots for saturation and diversity indicators, and for quality assurance between the codes in the annotated PDFs and the YAML-file codeset. At the conclusion of this stage, the open and axial codes were exported as the initial taxonomies.

1.1.4 Flow Diagrams

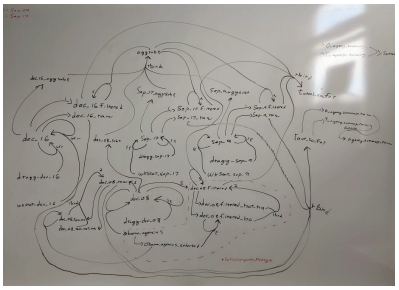
We quickly noticed that journalists frequently employ multiple tables. To facilitate our own understanding of their activity, we sketched table-based data flow diagrams of how raw data moves through the wrangling context when tables were used in complex ways. Figure 2 shows an example. These diagrams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



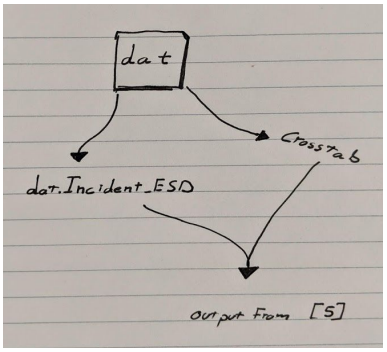
"Central Line Infection Data," *Vox,* July 8, 2015



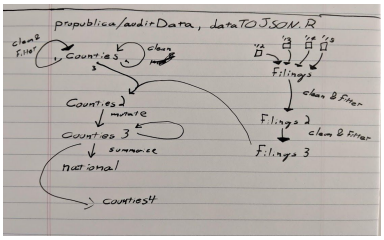
"Maryland Voter Registration Analysis," *Baltimore Sun*, Oct. 15, 2018



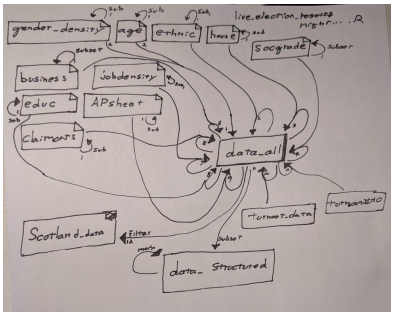
"How the Trump Era is Changing the Federal Bureaucracy," *Washington Post*



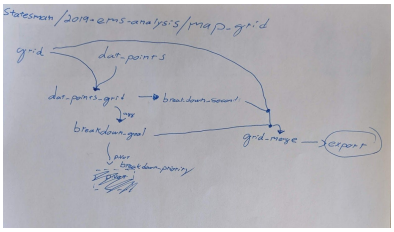
"Analysis of Austin-Travis County EMS Call Data", *Austin American-Statesman*



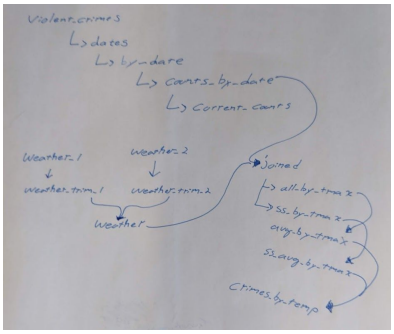
"Audit Rates by County," *Propublica*



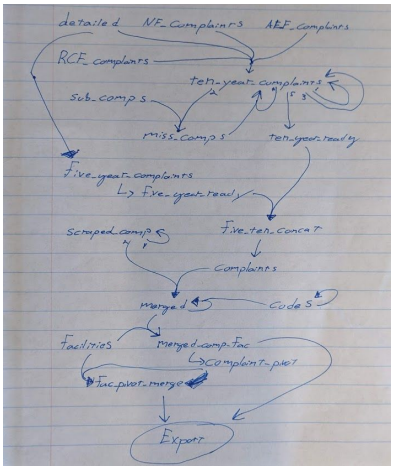
"How the Conservatives Won," *The Times* and *Sunday Times*



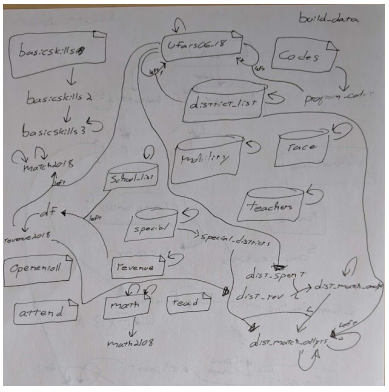
"Analysis of Austin-Travis County EMS Call Data", *Austin American-Statesman*



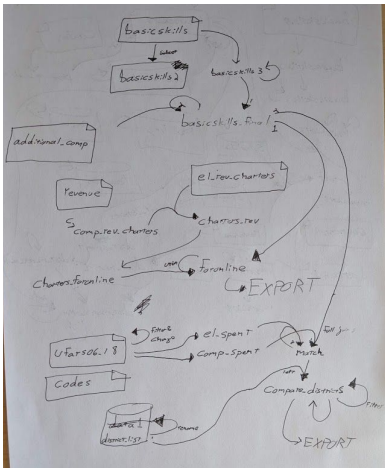
"Crimes and Heat Analysis," *St. Louis Public Radio*



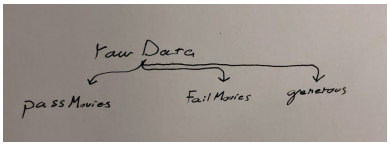
"Long-term Care Complaints Data and Analysis," *The Oregonian*



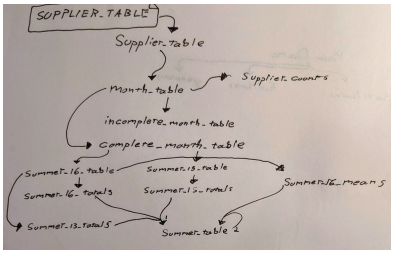
"Education Achievement Gap Analysis," *Star Tribune*



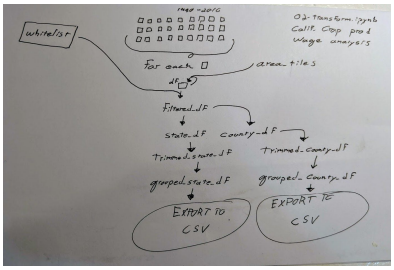
"Education Achievement Gap Analysis," *Star Tribune*



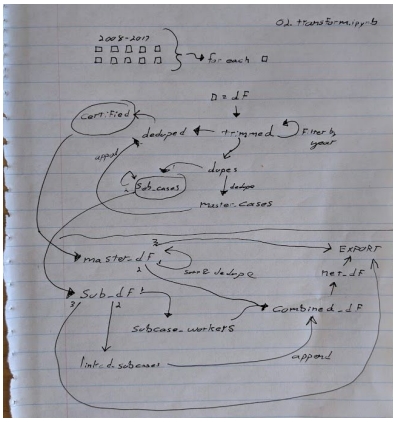
"Bechdel," *FiveThirtyEight*



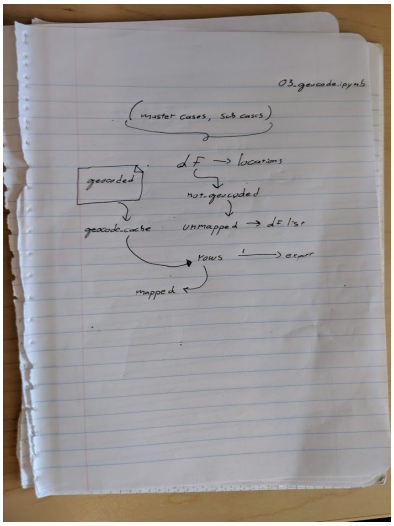
"Conservation-Consumption Score" analysis," *Los Angeles Times*



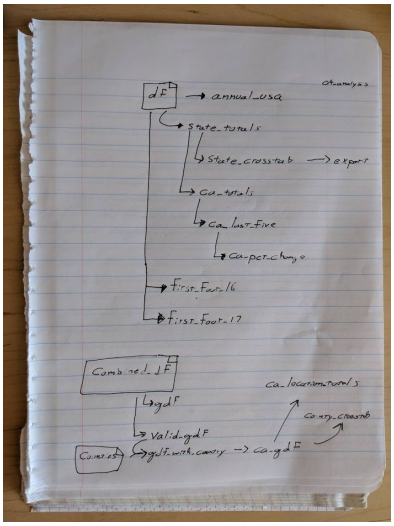
"California crop production wages analysis," *Los Angeles Times*



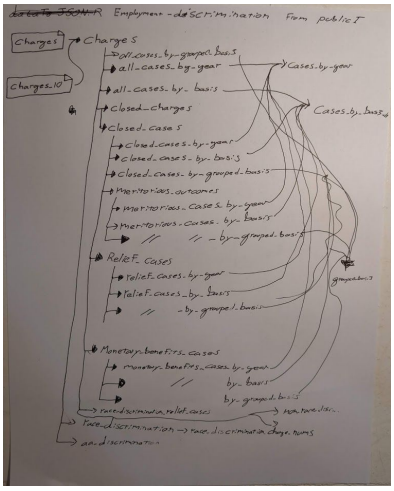
"California H-2A Visas Analysis," *Los Angeles Times*



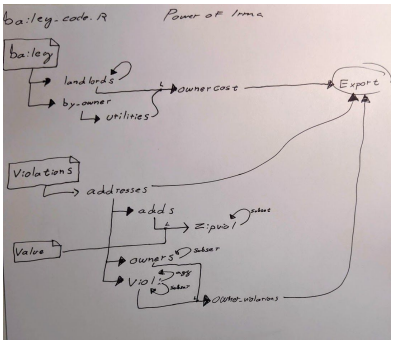
"California H-2A Visas Analysis," *Los Angeles Times*



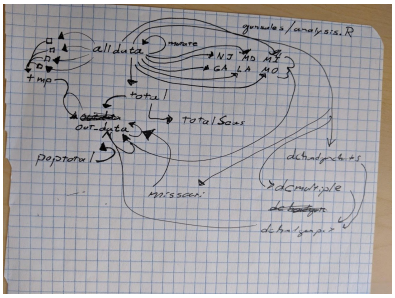
"California H-2A Visas Analysis," *Los Angeles Times*



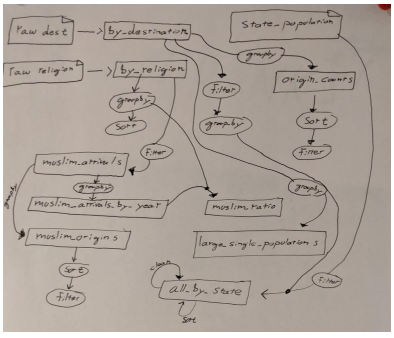
"Injustice at Work," *Center for Public Integrity*



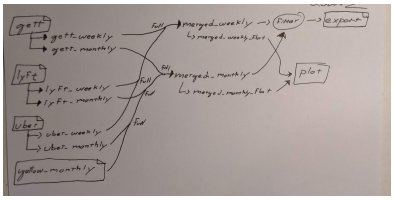
"Power of Irma," *WUFT*



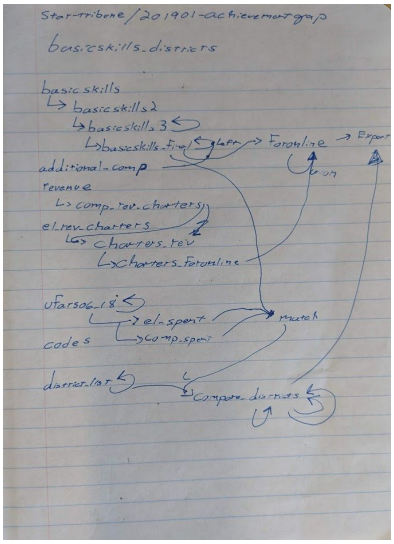
"Analysis of NICS gun purchase background checks," *New York Times*



US Refugee Data and Analysis, *BuzzFeed News*



"Analysis of rideshare trips taken in New York City," *Quartz*



"Education achievement gap analysis," *Star Tribune*

1.2 Phase Two: Literature Review

In the second phase, we address Q2: *Which practices align with or diverge from existing characterizations?*

Our literature search began with seed papers from two research domains: the computer science literature on data wrangling, and the computational journalism literature on challenges in journalistic data analysis. We identified these seed papers from the combination of background knowledge of the authors and targeted searching. We added to this set by following citations, for example from wrangling papers to papers on data integration, mashups, and network wrangling.

This search began concurrently with the phase one technical observation study. After the completion of the first phase, we more specifically searched for papers that address the role of multiple tables and data sources in the wrangling process. After creating the descriptive taxonomy, we reviewed the group of seed papers that we deemed relevant with two goals in mind: noting similarities and differences. First, we harmonized the names of elements in our bottom-up taxonomy with the usage in this previous work, for elements that aligned with previous frameworks, to create a final version of the taxonomy. Second, we verified our initial thoughts about which observed actions that we documented in our taxonomy did not match well with previous wrangling conceptual frameworks.

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 performing data wrangling 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 the operations supported by interactive wrangling applications, and we indeed

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.

1.3 Phase Three: Reflective Synthesis

In the final phase, we address Q3: *How to re-characterize wrangling to better match the observed practices?*

The key finding from the two previous phases was the discrepancy between the complex use of multiple tables by the journalists and the within-table emphasis of previous frameworks for data wrangling. We note that many programming languages and packages do support the concept of a table as a first-class object containing heterogeneous data, for example as objects called data.frames and DataFrame in R and the Pandas Python package, respectively. However, the idea of a table as an object to perform transforms on is largely absent from GUI-based wrangling tools. Most interactive wrangling applications such as Wrangler, Trifacta, OpenRefine, and Workbench support only what we call a single-table wrangling context: the interface is designed around a concept of a single matrix or spreadsheet where the table constitutes the environment, and rows and columns are the only objects being wrangled.

Although the bottom-up taxonomy provides a very thorough characterization of journalist behaviour, we found it did not suffice to translate this insight about the need for multi-table operations that could be supported through a GUI into actionable form. We saw the need to create a much more concise framework for multi-table operations that would be possible to operationalize effectively, with enough generative power to support the design of future interactive software. We thus synthesized a design space for multi-table data wrangling through reflecting on both our observational findings and previous wrangling frameworks.

We check the descriptive power of our framework by checking its coverage against our entire bottom-up taxonomy, as shown in Figure S4 of Supplemental Material. It fully covers all of the table-related actions that we observed, allowing the precise characterization of actions that cannot adequately be described with previous schemes. This coverage check also shows the extent to which the framework and the taxonomy are cross-cutting to each other, with very different internal structures.

2 Establishing Saturation

Part of our process of establishing saturation in our codeset is to explicitly monitor the number of unique codes with respect to the number of repos included in our technical observation study. Approaching 50 repos we notice the size of the codeset leveling off, as shown in Figure S1, showing that few new codes were being added. Moreover, we also informally monitor the number of axial code changes, which also converge. At this point, we determine the codeset has reached saturation, adequately describing data wrangling actions and processes in this domain.

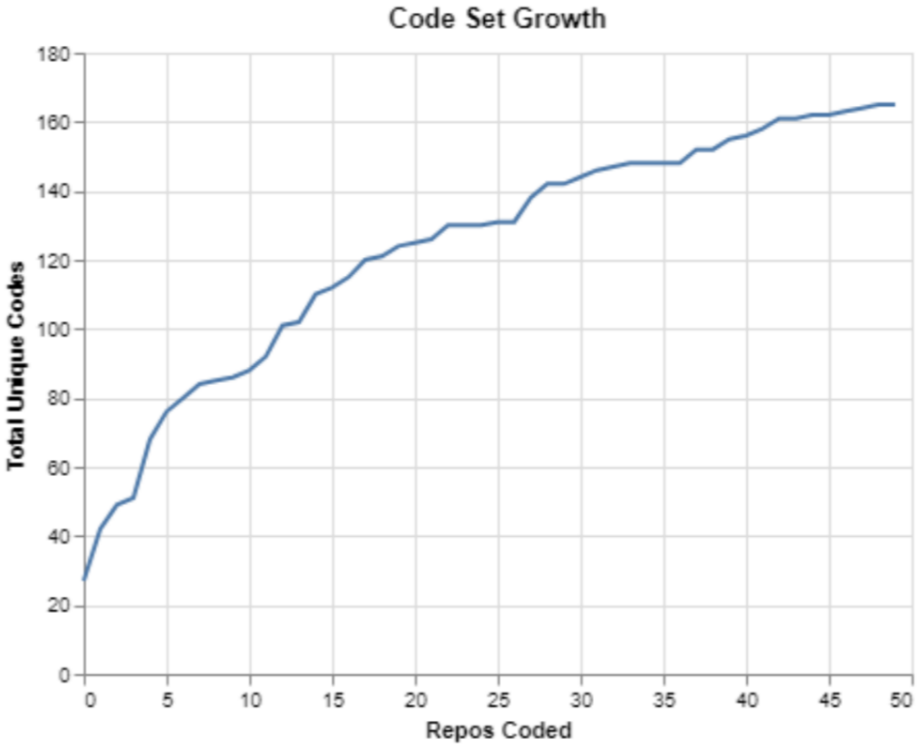


Fig. S1: Codeset growth per repo coded. We show which codes were introduced to the code set as each notebook is included in the analysis. After 23 notebooks, some computational notebooks did not add any new codes. By 50 notebooks, code set growth was so minimal that we declared our code set converged.

2.1 Newly Introduced Codes by Repo

Below we explicitly list which repos defined new open codes in our codeset. Repos are ordered by when they were coded in our technical observation study. Note that the naming conventions here reflect our internal project structure for organizing repos. There is not a one-to-one correspondence between the repo names on GitHub or in our spreadsheet of journalists repos. For example, *fivethirtyeight, The Times and Sunday Times, Vox, TrendCT,* and *Neue Zürcher Zeitung* all keep data journalism projects in one repo, and we have separated individual projects in this one repo into separate directories.

1. **buzzfeednews/2019-04-democratic-candidate-codonors:** create child table, trim by categorical value, repetitive code, count unique values, figure a rate, create annotations, outer join, compare groups, deduplicate, create soft key, group by variable, create a frequency table, trim by quantitative threshold, gather, load, format values, export, canonicalize variable names, remove variables, peek at data, govt data portal, union datasets, sort, change var type, self join dataset, aggregate, standardize categorical variables, construct a subroutine, align variables

2. **la-times/california-ccscore-analysis:** describe statistically, show trend over time, remove incomplete data, visualize data, calculate spread, count number of rows, standardize variable, trim by date range, inspect data schema, identify extreme values, cross tabulate, divide & conquer, calculate change over time, trim fat

3. **la-times/california-crop-production-wages-analysis**: adjust for inflation, construct data manually, construct data pipeline, wrangle data for graphics, combine periodic data, trim by geographic area, inner join, lookup table values

4. **la-times/census-hard-to-map-analysis**: parse variable, tolerate dirty data

5. **TheOregonian/long-term-care-db**: generate high-level summary, generate dataset identification, scrape web for data, create lookup table, refine table, fill in na values after an outer join, count the data, combine categorical values, edit values, replace na values, use non-public, provided data

6. **baltimore-sun-data/2018-voter-registration**: impute missing data, calculate a statistic, aggregate join, join aggregate, extract data from pdf, assign ranks

7. **nytimes/heat-index**: generate data computationally, cartesian product, examine relationship, compute index number

8. **buzzfeednews/2016-11-bellwether-counties**: rolling window calculation, get extreme values, create a unique key, remove non-data rows, spread table, use academic data

9. **stlpublicradio/2018-05-31-crime-and-heat-analysis**: split, compute, and merge, merge seemingly disparate datasets

10. **buzzfeednews/2016-09-shy-trumpers**: use another news orgs data

11. **nytimes/the-cube-root-law**: domain-specific performance metric, use public data

12. **buzzfeednews/2016-04-republican-donor-movements**: explore dynamic network flow

13. **la-times/california-h2a-visas-analysis**: consolidate variables, temporary joining column, preserve existing values, select rows with missing values, resolve entities, api request, schema drift

14. **wuft/endangered-species-act-louisiana**: scale values

15. **wuft/power_of_irma**: variable replacement, set data confidence threshold, use data from colleague, fix incorrect calculation, create togglable operations, use previously cleaned data, interpret statistical/ml model

16. **time/wikipedia-rankings**: collect raw data, explain variance

17. **time/babyname_politics**: resort after merge, data loss from aggregation

18. **buzzfeednews/2015-11-refugees-in-the-united-states**: test for equality, make an incorrect conclusion, lossy join

19. **publicI/employment-discrimination**: replace variable levels

20. **fivethirtyeight/bechdel**: data type shyness

21. **fivethirtyeight/bob-ross**:

22. **quartz/nyc-trips**: full join

23. **quartz/work-from-home**: concat parallel datasets, create a flag, copy table schema, data too large for repo, split and compute

24. **fivethirtyeight/buster-posey-mvp**:

25. **vox/verge-uber-launch-dates**:

26. **vox/vox-central-line-infections**: geolocate dataset records, report rows with column number discrepancies

27. **nytimes/prison-admissions**:

28. **baltimore-sun-data/school-star-ratings-2018**: remove duplicate variables

29. **bbc/electric-car-charging-points**: perform network analysis

30. **bbc/internal-migration-london**:

31. **bbc/midwife-led-units**: freedom of information data

32. **fivethirtyeight/librarians**:

33. **fivethirtyeight/infrastructure-jobs**:

34. **washington_post/federal_employees_trump_2017**:

35. **statesman/2019-ems-analysis**:

36. **propublica/auditData**:
37. **trendct/lending-club**:
38. **buffalonews/new-york-schools-assessment**:
39. **polygraph/skatemusic**:
40. **correctiv/awb-notebook**: test for null values, silently dropping values after groupby
41. **star-tribune/201901-hospitalquality**:
42. **times/general-election-2015-classification-tree**: wrangle data for model, check for nas
43. **star-tribune/201901-achievementgap**: bin values, query database
44. **npr/school-choice**: transpose
45. **nzz/1805-regionen-im-fokus-des-US-praesidenten**:
46. **la-times/swana-population-map**:
47. **la-times/california-buildings-in-severe-fire-hazard-zones**: search for clusters
48. **buzzfeed/us-weather-history**: validate data quality with domain-specific rules
49. **nytimes/gunsales**: adjust for season
50. **statesman/demolitions**:

3 Incorporating diversity

In order to prevent this code set from being biased by one individual or organization's data wrangling behavior, we deliberately sought out notebooks from a variety of news organizations and data journalists. This analysis comes from, but is not limited to, news organizations that constitute "major players" in data journalism.

3.1 Prolificness of News Organizations

Some news organizations are more engaged in data journalism than others. In order for the result of our technical observation study to be representative of the practices of a variety of organizations, we deliberately selected notebooks for inclusion in our technical observation study by news organizations across the spectrum of prolificness in this genre of journalism.

We ranked these organizations by two metrics based on our pool of journalistic code repositories containing data analysis:

- The count of individual code repositories
- The number of commits by journalists working for different news organizations

3.1.1 By Number of Repos

Most news organizations, including *BuzzFeed News*, *Los Angeles Times*, and the *Austin American-Statesman*, create one repository per analysis workflow. We include at least one repository from the top 19 news organizations by the number of unique repositories in our pool journalistic code repositories containing data analysis. We also deliberately select repositories from news organizations that only have one repository in this pool.

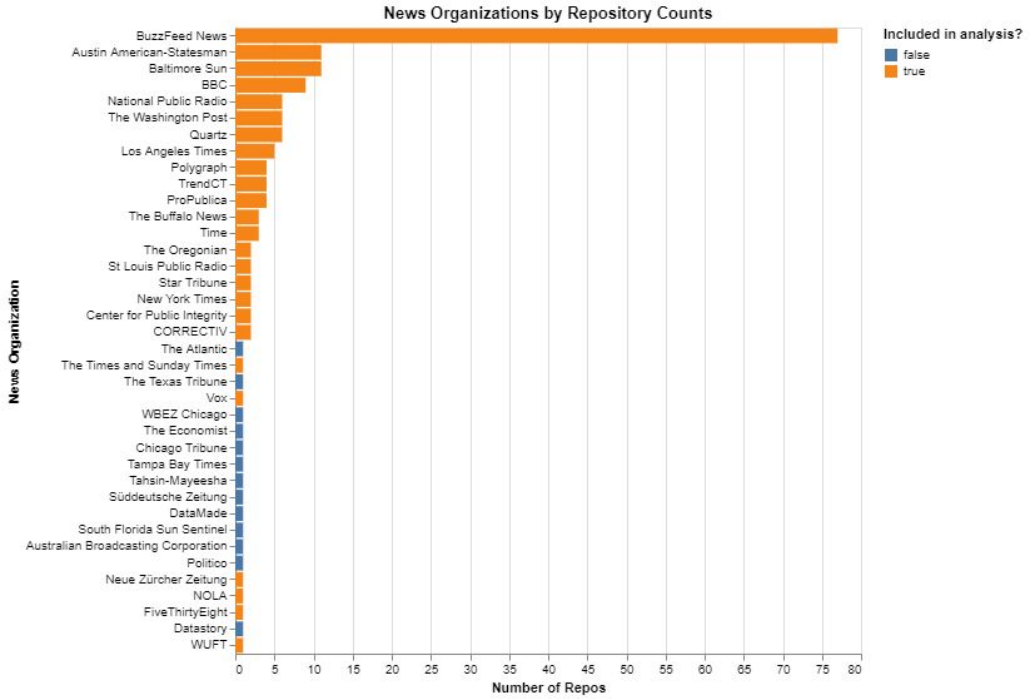


Fig. S2: News organizations ranked by number of commits across all associated repositories. This bar chart shows the sum of all commits between multiple users with membership in various news organizations on GitHub in our curated pool of journalistic, data-analysis repositories. The chart is color-coded by whether at least one repository from that news organization was included in our technical observation study. Orange values indicate the news organization was included and blue indicates otherwise.

3.1.2 By Commits

However, one limitation of ranking news organizations by the number of repositories is that some organizations, such as FiveThirtyEight, keep computational notebooks for multiple data journalism articles in one master code repository. A commit in Git can be thought of as a unit of change. Thus, the more a repository has changed over time, the more commits. If a news organization is only using one repository for all their data journalism work, then it should have lots of commits.

When ranking news organizations by commit counts, our qualitative analysis includes the top 18 news organizations by commit count in addition to news organizations with only a few commits in our pool of journalistic code repositories containing data analysis.

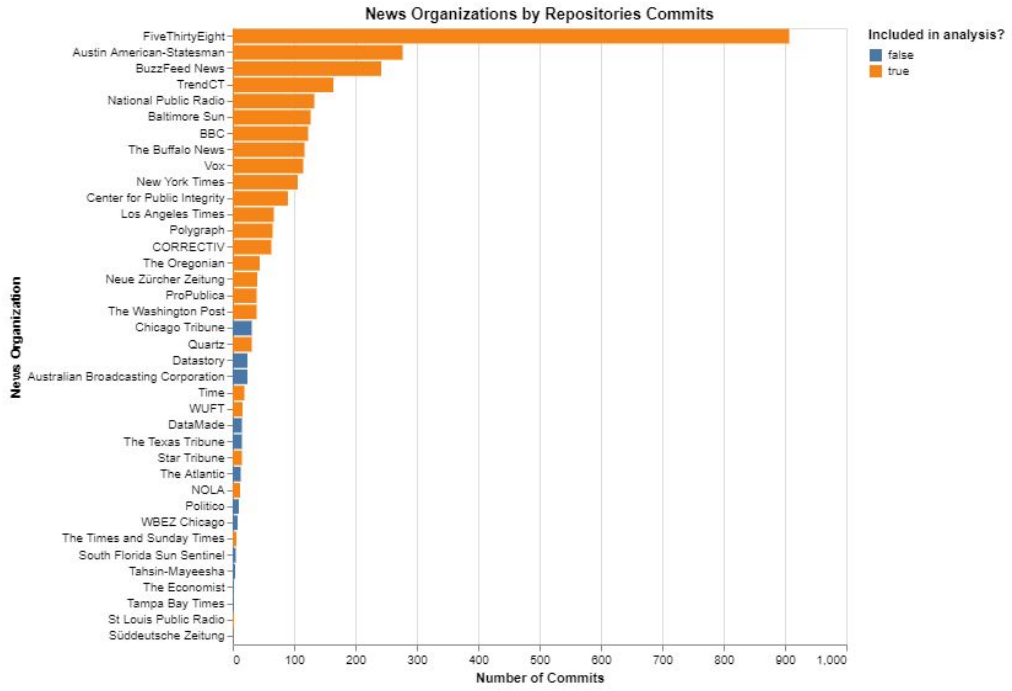


Fig. S3: News organizations ranked by number of commits, which we use as a proxy measure for prominence in the data journalism community. This bar chart shows the number of commits per news organization in our pool of journalistic, data-analysis repositories. The chart is color-coded by whether at least one repository from that news organization was included in our technical observation study. Orange values indicate the news organization was included and blue indicates otherwise.

This analysis includes 25 news organizations out of 37 that had computational notebooks deemed relevant to this analysis (67.57%).

Organization	**Is included?**
Austin American-Statesman	Yes
Australian Broadcasting Corporation	No
BBC	Yes
Baltimore Sun	Yes
BuzzFeed News	Yes
CORRECTIV	Yes
Center for Public Integrity	Yes
Chicago Tribune	No
DataMade	No
Datastory	No
FiveThirtyEight	Yes
Los Angeles Times	Yes
NOLA	Yes
National Public Radio	Yes
Neue Zürcher Zeitung	Yes

New York Times	Yes
Politico	No
Polygraph	Yes
ProPublica	Yes
Quartz	Yes
South Florida Sun Sentinel	No
St Louis Public Radio	Yes
Star Tribune	Yes
Süddeutsche Zeitung	No
Tampa Bay Times	No
The Atlantic	No
The Buffalo News	Yes
The Economist	No
The Oregonian	Yes
The Texas Tribune	No
The Times and Sunday Times	Yes
The Washington Post	Yes
Time	Yes
TrendCT	Yes
Vox	Yes
WBEZ Chicago	No
WUFT	Yes

3.2 Prolificness of Individual Journalists

In addition to taking steps to incorporate comprehensiveness and diversity of news organization into our descriptive taxonomy, we also attempt to add comprehensiveness and diversity in the individual journalists.

We exclude some data journalists with commits from this summary because their commits were insignificant contributions to repos such as comments, README file updates, initial repo setup, and general code clean up.

- Andrei Scheinkman, *FiveThirtyEight*
- Dhrumil Mehta, *FiveThirtyEight*
- Stephen Turner, *FiveThirtyEight*
- Nate Silver, *FiveThirtyEight*
- Dan Nguyen, *The Upshot*
- Derek Willis, *BuzzFeed News*

Note that this summary also excludes journalists who:

- Worked collaboratively and only one of them committed code.
    - Matt Stevens
    - Adam Pearce
- Only were included in the technical observations study via Observable notebooks
    - Sahil Chinoy
- Did not commit their own code. For example, *FiveThirtyEight* code appears to be committed by someone else.
    - Rob Arthur
    - Stefano Ceccon
    - Walt Hickey

3.2.1 By Commits

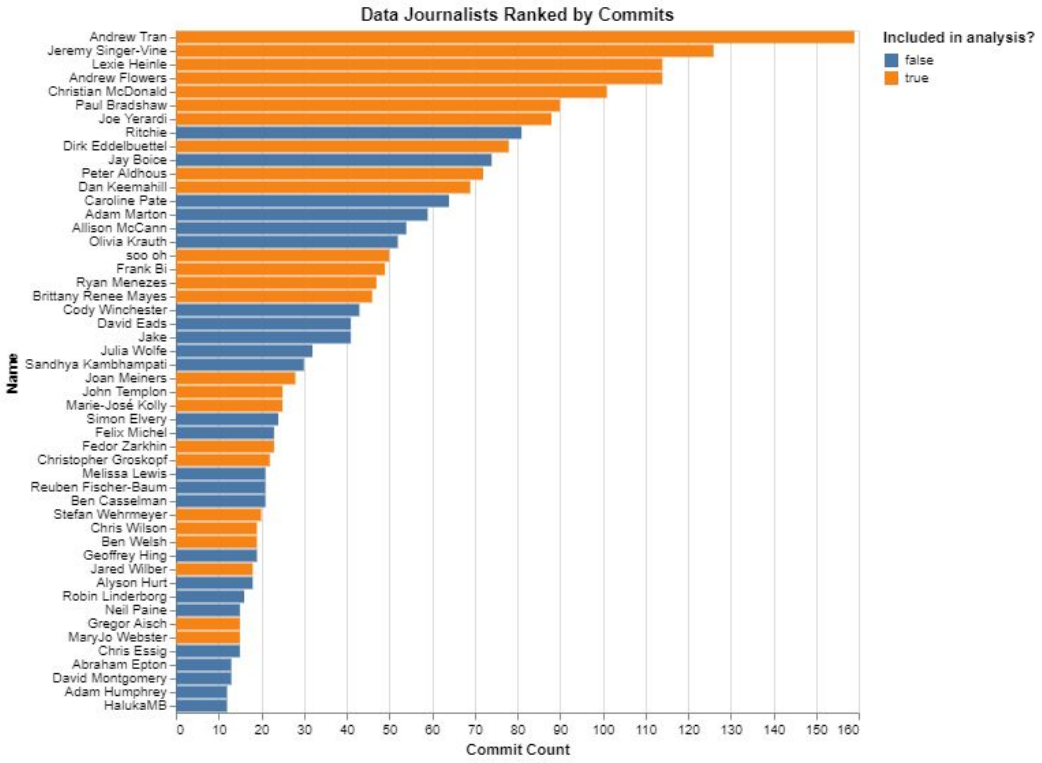


Fig. S4: Data journalists who authored code repositories in our pool of journalistic, data-analysis repos, ranked by number of commits. This chart is color-coded orange to indicate that the individual authored an analysis included in our technical observation study.

3.2.2 By Followers

Our qualitative analysis is based on repositories authored by the top eight data journalists ranked by the number of followers in addition to many GitHub users with less followers.

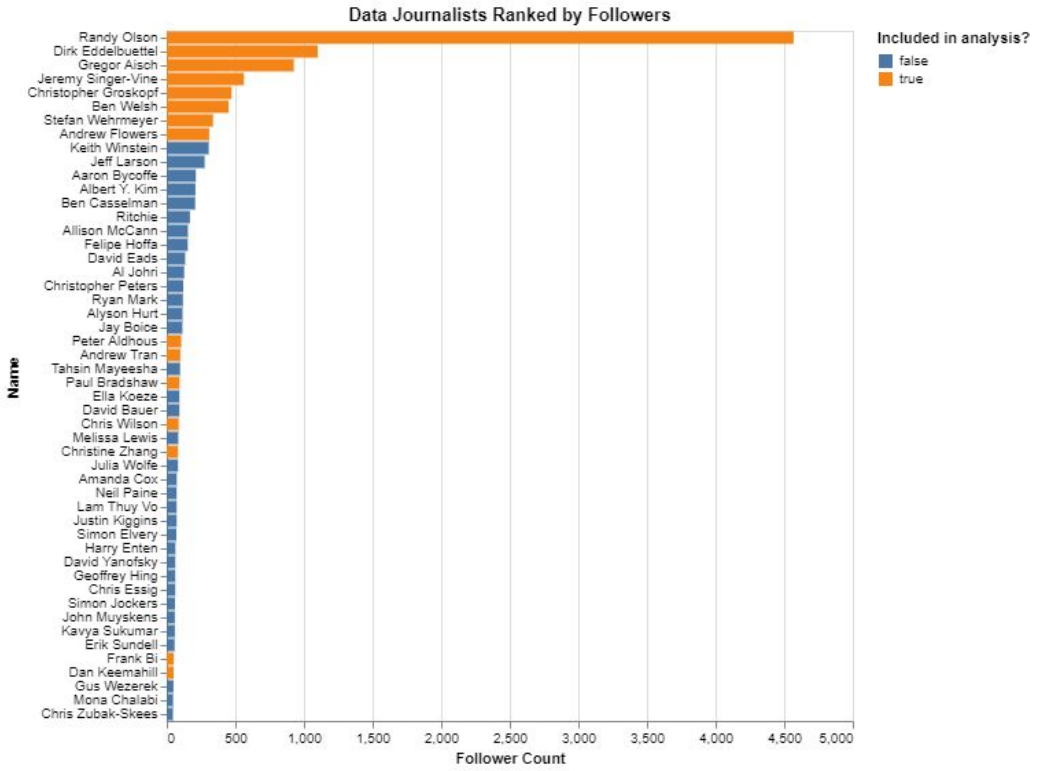


Fig. S5: Data journalists who authored code repositories in our pool of journalistic, data-analysis repos, ranked by the number of followers. We use an individual's popularity as a proxy measure for their prominence within the data journalism community, vetted by our knowledge within the domain. This chart is color-coded orange to indicate that the individual authored an analysis included in our technical observation study.

4. Terminology Harmonization

This document provides further details on how we harmonized, and deliberately did not harmonize, the open codes we developed during qualitative coding with terminology in other work. The terminology harmonization largely happened in Phase 2: Literature Review.

4.1 Salient Examples

Changes to the codeset that reflect alignment with commonly used terminology. Some high-level examples.

- **Fetch**, we initially coded many of the actions involving requesting data from sources external to the environment as *grab* or *extract*, which we consolidate and standardize as *fetch* in the final version of our codeset.

- **Create Soft Key**, we initially coded instances where users created observation keys with the intention of being unique, but without such guarantees as *Create a semi-unique key*, but renamed

it to *Create Soft Key* to reflect the usage in data cleaning literature.

- **Remove Variables, Remove Observations,** we initially coded instances where users removed observations, rows in tidy data, and variables, columns in tidy data, as rows and columns using several verbs: *winnow* and *subset*. We chose to utilize the terminology of tidy data as it gave us a vocabulary for describing data independent of how the dataset is structured in the programming environment. This convention became useful when discussing wrangling work-flows that utilize the table-like data frame object as well as other structures, such as lists of dictionaries, as is the convention in the Python package `agate`, which was developed for journalists.

- Database operations, we consolidated the code of many familiar database operations with terminology from their equivalent SQL specification. For example, *consolidate* became *Union Dataset* and *Intersect* became *Inner Join*.

- **Split, Compute, and Merge** We initially coded many high-level data flow transformation patterns where data is separated, operated upon, and combined, under *split and merge*. While we acknowledge that this step is similar to Wickham's *Split, Apply, Combine* strategy of data analysis. We did not resolve it with this terminology to maintain the distinction that in our observations, users would apply different computations to subsets of the data, while Wickham's work specifies applying the same computation to each subset of the original dataset.

4.2 Version Differences

We summarize changes to the codeset during the nomenclature harmonization pass.

- Harmonize terminology for the entire codeset with Tidy data. Changes columns, rows, and tables with Tidy equivalents: variables, observations, datasets.
- ***Grab → Fetch*** *(rename),* describing data retrieved from some external source.
- ***Create a Semi-Unique Key → Create Soft Key*** *(rename)*, describing a key used with the intention of being unique, but not guaranteed to be unique.
- ***Winnow Columns → Subset Columns*** *(rename)*, describing the removal of columns.
- ***(Fetch, Extract) → Grab*** *(consolidate)***,** describing pulling data out of other formats into a tabular one.
- **Calculate Z-Score → Standardize Values** (*consolidate),* metrics that quantify deviation from some definition of "normal."
- **(Calculate Mean, Calculate Media) → Calculate Central Tendency** *(consolidate)***,** metrics that describe a "typical" value.
- ***Fix Data Errors Manually → Edit Table Values*** *(rename)*, directly editing the table
- ***Consolidate → Union Table*** *(rename),* Combine the rows of two tables
- ***Intersect → Inner Join Tables*** *(rename,* Join to table's columns such that non-matching rows from either table are absent from the resulting table.
- ***Encode Table Summary Data in Row → Aggregate Join*** *(rename)*, combining aggregated data with the low-level data column-wise.

- General restructuring of axial code groups.
- ***Split and Merge → Split, Compute, and Merge*** *(rename)*, following the data analysis pattern.