Cognitive Dimensions of Between-Table Context Support in Direct Manipulation Wrangling Interfaces

Steve Kasica

Dec. 13, 2019

Abstract

Despite many commercially available tools that support or are designed explicitly for data wrangling, there exists no systematic evaluation of the strengths and weaknesses of these tools. This analysis project for CPSC 547 Information Visualization evaluates two popular and actively-developed wrangling tools, OpenRefine and Dataprep. By reproducing the wrangling processes conducted by journalists originally using idiosyncratic scripts written in Python and R on real-world data identified in prior work, this project is able to compare and contrast the usability of both applications in the context of real-world data. This usability analysis is based on the cognitive dimensions of notation framework, a user-interface independent set of tools designed explicitly to facilitate such comparisons. In the end, this report finds that OpenRefine and Dataprep share much of the same core functionality; although, Dataprep's use of visualization leads to less *error-proneness* in the overall process and higher quality data its conclusion.

1. Introduction

This analysis project aims to reproduce the wrangling process from two data journalism projects where the journalists wrangled their data with scripts and computational notebooks written in different programming languages. This small but active group of data journalist are proficient in many of the computational tools and statistics techniques of data science; however, they constitute a minority within the population of all journalists who are increasingly looking to enhancing their reporting and tell stories with data. GUI-based, direct-manipulation wrangling interfaces that do not require the user to write any computer code thus have the potential to make data available to more journalists.

Data preparation and wrangling is a well-known, acknowledged step in data journalism. The conference on Computer Assisted Reporting (CAR) holds workshops and tutorials for professional journalists to sharpen their data wrangling skills in R, Python, and OpenRefine. University journalism departments offer courses on data journalism and visualization also incorporate a module on data cleaning, preparation, or wrangling in the syllabus.

Journalists are an interesting sub-group to study in the context of data wrangling because this user group is exposed to a variety of data types and domains. One data journalist may deal with both structured and unstructured data from domains as diverse as civics, biology, climatology, and social sciences. Also, journalists often publish their analysis code and data on public code repositories, such as GitHub. This represents a rich data source on wrangling that was utilized in prior work that this project builds upon.

2. Domain Background

This analysis project focuses on applications that leverage visualization in the domain of data wrangling.

2.1 What is data wrangling?

Data wrangling, also known as data munging, is not as much an individual task as a process of iterative exploration and transformation that enables analysis [6]. This process includes many well-known, overlapping data tasks such as: cleaning, reshaping, integrating, integrity inspection, transforming, restructuring, and tidying. While other disciplines of computer science have developed fully automated approaches to many of these same tasks, wrangling

differentiates itself by its unguided, exploratory nature. Hence, wrangling is especially applicable to journalist who obtain datasets through leaks or freedom of information requests without a clear picture of its potential applications or existing data quality issues.

Wrangling is often implemented in single-use scripts of sequential computer code or through manual table transformations. Wrangling is often done in GUI-based applications, such as Microsoft Excel, or as scripts written in a computer programming language. These wrangling scripts are often written in programming languages such as Python, Perl, or R. A script usually is only applicable to one wrangling processes with a few individual datasets. Hence, the initial cost of programming a script cannot be amortized across different datasets even though many of the lower level table transformation may be the same. Wrangling in an application is often tedious, when doing specific wrangling tasks in a general purposes spreadsheet application [6].

In 2019, there exists many commercial and open-source tools capable of wrangling data. Generally speaking, these tools can be divided into two categories: tools intended specifically for data wrangling and general-purpose data tools with wrangling features. This analysis only compares OpenRefine and Google Cloud Dataprep, which are two applications specifically for wrangling. These are the only tools under consideration in this project because they were recommend for advanced data cleaning in the course Data Journalism and Visualization with Free Tools [15]. This massive online open course (MOOC) is organized by the Knight Center for Journalism in the Americas and the Google News Initiative. While this course also addressed some data wrangling tasks in Google Sheets, these tasks were mainly trivial compared to the kinds of issues addressed with OpenRefine and Dataprep.

2.2 Prior Work

This analysis project builds off of previous research I conducted over a four-month period in the summer of 2019. I analyzed the workflows of data journalists in the wild with a particular eye towards how this user group wrangles data. In this artifact-mediated indirect observational study of data wrangling in data journalism analyses, I performed thematic analysis on 50 collections of computational notebooks and programming scripts from 33 journalists at 26 news organizations. This iterative process of open and axial coding resulted in a hierarchial taxonomy of data wrangling actions and observations that includes 131 codes.

2.3 Previously Identified Workflows

From this prior work, I utilize two artifacts in this analysis project: the original raw datasets used by professional journalists and the record of table transformations applied to these datasets. Collectively, these are referred to as workflows. This analysis replicates two data wrangling workflows from professional journalists. based on cleaning real data on enrollment figures for long-term managed care plans in New York State at *The New York Times* and the other on water usage statistics following a years-long drought by *The Los Angeles Times*.

The workflow Long-term Managed Care (LMC) follows a tutorial taught by Sarah Cohen, then an assistant editor for computer-assisted reporting at *The New York Times* and adjunct professor at Columbia University. This data has also been used to teach advanced data cleaning to journalists as part of a data journalism class at Columbia in 2015 and at the Computer Assisted Reporting (CAR) conference in 2016. This workflow wrangles a single table of Medicaid long-term managed care reports from New York State, and presumably comes from an actual wrangling activity conducted at *The New York Times*. Cohen mentions the purpose of this activity is to quickly compare companies on growth and size for further investigation using traditional reporting methods.

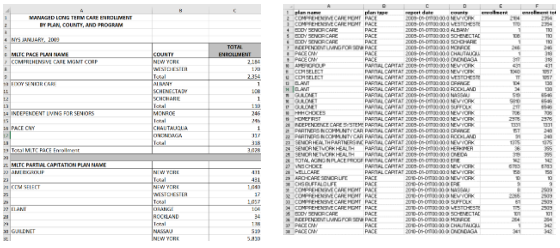

Figure 1: Raw data (left) and its final, wrangled output (right) in the Long-term Managed Care (LMC) workflow.

It is just a coincidence that the Long-term Managed Care (LMC) workflow comes from a news organization on the eastern coast of North American and the second workflow comes from a news organization on the western coast. In Oct. 2016, the

Los Angeles Times published an investigation on county water usage in California after the state government rescinded a mandate restricting water usage. California is a state in the U.S. that suffered from years-long drought peaking between 2013 and 2015. Reporters Matt Stevens and Ryan Menezes further investigated one county that stood out from the rest of the data. This article is an example of the most common genre of data journalism article seen in prior work: articles that compare multiple entities along a common performance metric. Often, the stories in this kind of data are the outliers, as was the case with Stevens and Menezes's reporting.

3 Data and Task Abstraction

This analysis project derives domain-specific and abstract tasks and data from prior work performing qualitative analysis on records of how professional journalists wrangle their data "in the wild." Section 5 on methods and tool elaborates on this prior work.

3.1 Raw data wrangled by journalists

The data used in this project is the same raw data used by journalists. This data was collected from repositories made publically available in conjunction with published articles. The raw data itself is checked into the repository instead of providing instructions on how to obtain it from its original source. This posterity measure ensures that this raw data will remain available for years to come.

These two workflows were selected because the data they wrangle balances each other well. The *New York Times* workflow deals with mostly categorical data that exists in a pivot table in its raw form. The workflow from *The Los Angeles Times* deals mostly with quantitative data and more quantitative variables derived from those in the raw data. While journalists occasionally work with network and tree data [11], this analysis project only considers simple flat tables because it was he most common abstract data type used in prior work.

The raw data used by *The New York Times* workflow comes compiled from multiple Excel documents obtains by reporters. This dataset was selected for this project because it contains mostly categorical data. This raw table data consists of five attributes and 3,782 items.

- **Plan name** (categorical): The name of the healthcare plan. This attribute constitutes the table key.
- **Report Date** (date): The month and year of the enrollment report.
- **Plan type** (categorical): The type of long-term managed care plan in the report.
- **County name** (categorical): the name of the county in New York State.
- **Enrollment** (quantitative): the total number of people enrolled in a plan per county.

The raw data used by the *LA Times* workflow comes directly from California's State Water Resources Control Board. This state government entity periodically publishes district-level water usage statistics to their website. The *LA Times* includes an Excel file in their repos published to the organization's account on GitHub. We know the raw data's source because it is listed in a section in the published, online article called "How we did it."

The raw version of this water usage table data straight from the California government has 10,936 items and 32 attributes. The data dictionary constructed from the raw data below is a subset of all data variables.

	Supplier Name	Stage Invoked	Mandatory Restrictions	Reporting Month	REPORTED Total Monthly Potable Water Production Reporting Month	REPORTED Total Monthly Potable Water Production 2013	REPORTED Monthly CII 2014/2015 (Subset of REPORTED Total Monthly Potable Water Production Reporting Month)	REPORTED Monthly Ag Use Reporting Month (This value is removed from REPORTED Total Monthly Potable Water Production Reporting Month by Water Board staff to obtain CALCULATED Total Monthly Potable Water Production Reporting Month Gallons)
0	East Bay Municipal Utilities District	0	No	2016-08-15	6007.50	7172.30	1141	NaN
1	East Bay Municipal Utilities District	0	No	2016-07-15	6056.60	7452.20	994	NaN
2	East Bay Municipal Utilities District	0	Yes	2016-06-15	5675.90	6927.50	839	NaN
3	East Bay Municipal Utilities District	4	Yes	2016-05-15	4959.30	6716.50	955	NaN

Figure 2: The raw data used by the *Los Angeles Times* comes straight from California's State Water Resources Control Board. The structure of this data is more receptive to computational methods and thus requires less reshaping than the data in the workflow from *The New York Times* on long-term managed care enrollment numbers. The final, wrangled form of this data is included in Figure 3.

- **Supplier Name** (Categorical): The name of the municipal utility district, such as Easy Bay Municipal Utilities District. This attribute constitutes the table key.
- **Mandatory Restrictions** (Categorical, expressed as Yes/No categories): Whether the district was subject to mandatory water restriction during the reporting month.
- **Reporting Month** (Date) The day, month, and year of the report.
- **REPORTED Total Monthly Water Production Reporting Month** (Quantitative): potable water production during the reporting month
- **REPORTED Total Monthly Potable Water Production 2013** (Quantitative): the water production for the observation month in 2013.
- **Total Population Served** (Quantitative): the population served by the utility district.
- **Supplier has Agricultural Water Use Exclusion Certification** (Categorical, expressed as Yes/No categories): Whether the utility district can subtract water delivered for commercial agriculture from their total potable water production total.
- **% Residential Use** (quantitative): The percentage of potable water that's intended for residential use.

Both tables consist of categorical and quantitative data. The attributes "Supplier has Agricultural Water Use Exclusion Certification" and "Mandatory Restrictions" from the *Los Angeles Times* workflow are classified as categorical as opposed to Boolean, even through the only two levels in this variable were "Yes" and "No," which naturally correspond to True and False. Both table did not have attributes that could be considered ordinal data.

3.2 Wrangling tasks by journalists

I derive tasks in this project from the action codes applied to each workflow from prior work. These were referred to as actions, as opposed to tasks. Tasks imply intention, but because this indirect observation study did not include interviews with journalists, we cannot make claims about intentions. This prior work gives an auditable, reproducable record of the wrangling sequences applied to the data from its raw form to its final formats. This data provides a strong signal of what wrangling tasks journalists perform and how they accomplish them. Why these journalists did what they did and how they did it is still an open question.

Part of the task abstraction contribution for this project involves deriving tasks from these sequences of actions. I substitute the original authors intention with my own judgement from my experience as a journalist and data wrangler familiar with Python and R. Actions from prior work and the tasks derived in this project share a many-to-one relationship, one tasks is comprised of many actions. Thus, the process of deriving tasks is simply segmenting consecutive actions into semantically meaningful chunks. For each task, I also recorded a snapshot of the intermediate table representation as a benchmark for the wrangler, myself, to achieve. Table 1 details each derived tasks for both workflows but not in the order they occur in the workflows.

In reproducing each workflow in OpenRefine and Dataprep, I only consulted the task sequence, which does not list the underlying actions. The task sequence for each workflow is provided in Supplementary Materials. It would be trivial to reproduce the exact sequence of actions in each application. More can be learned about the strengths and weaknesses of each application by only specifying the desired state of the wrangled data at the end of each benchmark.

The workflows I reproduced have been closely read at least three times, first to analyze the workflow in prior work and twice for each application. First, at least five days passed between the same workflow using the two different wrangling applications. Second, the application-workflow order was also varied to further counter balance the experiment design.

	supplier_name	total_water_production_16	total_water_production_15	total_water_production_13	savings_16	savings_15
0	Adelanto City of	434024228.54	387316100.00	393342171.40	0.10	-0.02
1	Alameda County Water District	3937000000.00	3374000000.00	5273000000.00	-0.25	-0.36
2	Alco Water Service	344299000.00	350899000.00	447983000.00	-0.23	-0.22
3	Alhambra City of	775637185.75	777996350.08	1060724599.23	-0.27	-0.27
4	Amador Water Agency	350910000.00	287480000.00	431220000.00	-0.19	-0.33
5	American Canyon, City of	279580524.37	282187335.78	387763198.13	-0.28	-0.27
6	Anaheim City of	5449865116.58	4767206377.01	6479881477.32	-0.16	-0.26

Figure 3: A subset of the wrangled data using in the workflow from *The Los Angeles Times*. The high-level wrangling objective for this data is to aggregate the key attribute and derive a performance metric from quantitative attributes in the original data.

The task sequence derived from *The Los Angeles Times* workflow has a two salient data wrangling tasks. First, one of the first acts of wrangling was to remove all variables from this dataset but five variables of three data types: water supplier name

(categorical), the month and year of the reading (date), and total water production in gallons (quantiative), total water production in gallons for 2013 (quantitive), and the percentage of total water production that was used in residential zones (quantitative). Second, the month variable, in the sense of variables in Tidy Data [16], exists in two table columns. Water production values for 2013 have their own column, while production-month values for the remains years are properly separated into two columns. This data quality error, a structurally-spliced variable, is a difficult issues to address with wrangling.

The Long-term Managed Care workflow concerns converting a dataset intended for presentation into a dataset intended for computation. This task sequence highlights two important data quality issues addressed by wrangling and one common wrangling task. First, the raw data pivots upon plan name and county to create a hierarchial encoding for total enrollments numbers along the vertical position. Second, the data also includes total numbers for each plan names and for each county within a plan name as rows. Although not a data quality issues, this workflow illustrates an Aggregate Join (T9), adding the total enrollment within a plan name as a separate column at the farthest right column of the final-output table.

Task	Description	LMC	CCS
T1	Extract value in column	✔	
T2*	Reshape table		✔
T3	Remove observations	✔	✔
T4*	Aggregate Join	✔	
T5	Deduplication		✔
T6*	Resolve entity names	✔	
T7	Derive variables		✔
T8	Aggregate observations	✔	
T9	Remove columns		✔
T10	Trim the Fat		✔

Table 1: Tasks with asterisks denote tasks that prior work observed being performed in both a within- and between-table context. LMC refers to the workflow Long-term Managed Care, and CCS refers to the workflow California Conservation Scores.

4. Related Work

This analysis project is related to other work performing usability analysis using the cognitive dimensions of notation framework.

4.1 Cognitive Dimensions

In response to a lack of user interface design methodologies grounded in the design activities of user interface designers in the 1990s, Blackwell and Green describe a cognitive dimensions of notation framework [1]. Rather that positioning it as an analytic method, cognitive dimensions of notation are a framework of interface-independent discussion tools for evaluating the cognitively-relevant features in user interfaces and non-interactive notation.

Related work on usability analysis using this framework mostly concern visual programming languages. Although this framework is supposed to extend to interactive devices, usability-analysis papers incorporating cognitive dimensions often deal with non-interactive notation, especially visual programming environments. Green and Petre, 1996 [2] evaluate two commercially-available data flow languages, Prograph[1] and LabVIEW[2]. Today there is still active support for the Prograph language, and LabVIEW is still receiving active support from National Instruments. This project is different from related work by focusing on two wrangling applications that fall within the category of direct-manipulation interfaces.

4.2 Evaluation of wrangling applications

Related work in evaluating wrangling applications is often done in the context of evaluating novel wrangling tools or techniques by the designer/paper authors. To the best of my knowledge, there does not exist a systematic evaluation of existing wrangling applications by a third-party.

To validate the Wrangler, Kandel et al. performed a controlled user study comparing Excel to their wrangling application in three tasks. Wrangler [7] is a mixed-initiative user interface that drives an underlying declarative transformation language evaluated. In a user study to validate the usability of the interface, researchers compared Wrangler to Excel in three wrangling tasks: extracting text from a column (T1), fill missing values, and table reshaping (T3). While this project includes the same tasks, this

[1] https://en.wikipedia.org/wiki/Prograph

[2] https://www.ni.com/en-ca/shop/labview.html

usability study took a more quantitive approach, measuring time to completion and performing ANOVA on the results of a post-study questionnaire. This project takes a strictly qualitative approach to comparing wrangling applications.

5 Methods & Tools

This analysis project conducts a usability analysis based on the cognitive dimensions of notation framework to evaluate two tools used by journalists for data wrangling. This section includes an overview of data wrangling tools with a more detailed description of the two tools evaluated in this project: OpenRefine and Google Cloud Dataprep.

All of these tools constitute direct-manipulation interfaces. Hutchins et al. [4] define direct-manipulation interfaces as systems where the user has the sense of performing operations directly upon the objects instead of through an abstraction computational medium. All of these applications incorporate a spreadsheet metaphor of the underlying data structure into their interfaces to give the user the impression they are directly manipulating the data; however, the actual organizational structure of the data on a user's computer does not necessarily match the structure on the screen. Example of wrangling applications that are not direct-manipulation interfaces include scripts, computational notebooks, and other environments where the user is wrangling via a programming language.

5.1 Overview of data wrangling tools

Within the category of direct-manipulation interfaces for wrangling, we can divide all existing productions into two categories. First, there are general purpose data tools with wrangling features. Microsoft Excel[3] is the general spreadsheet software by which all data tools are invariable compared against. In the user study conducted to provide an initial evaluation of Wrangler, Excel was the baseline application [7]. It includes features to pivot one's data, which structurally transforms the underlying data into a cross-tabulated format. Google Sheets[4] is a free, online, and cloud-based spreadsheet application in the same product category as Excel. It includes features to deduplicate table rows that contain identical values for all columns. Deduplication (T5) is a common, wrangling task.

The second category of direct-manipulation interfaces for wrangling are applications designed specifically for wrangling. First, Trifacta Wrangling is an interactive data cleaning application that can be run on the desktop or in the cloud. It is the latest commercial evolution of research on interactive data cleaning/wrangling systems by researchers at Stanford and University of California Berkeley in the early 201**0s** [7], [10]**.** For nearly all intents and purposes relevant to the user, Trifacta Wrangler is Google Cloud Dataprep is an instance Trifacta Wrangler running on the Google Cloud platform. Second, Tableau Prep is a desktop wrangling application that includes a three-panel view of the data: a high-level provenance graph of table transformation, a profiling panel of dataset variables, and a traditional spreadsheet/table view of the data being wrangled. Finally, Workbench is a recent open-sourced, cloud-based data cleaning platform.

5.2 OpenRefine

OpenRefine [5], also known as Refine, is one of the oldest applications for wrangling data. The open-source project has gone through previous names as it has changed hands between various supporting organizations. It was known initially developed and known as Freebase Gridworks when it was under the development of Metaweb Technologies, Inc in May 2010. It was renamed to Google Refine when Google acquired Metaweb in July of the same year. In October 2012, Google ceased active support for the project and it became known as OpenRefine [12].

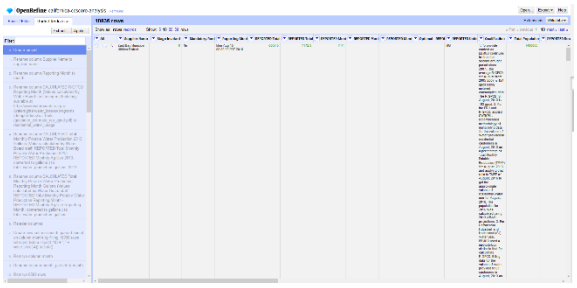

Figure 4: The OpenRefine interface loaded with raw data from the California Conservation Score workflow. Like all wrangling applications, the interface is organized around a table view of the data; however, more sophsticated visualizations are incorporated into other parts of the interface.

The model for applying wrangling operations in OpenRefine largely fit into an iterative subset-modify cycle. Users begin by selecting all or a subset of the

[3] https://products.office.com/en-ca/excel

[4] https://www.google.com/sheets/about/

data and transforming the selected portion. The predominate organizational principle in OpenRefine is a distinction between rows and records. A row in OpenRefine corresponds to a row in a table; however, a record refers to multiple, sequential rows with an index key that exists within the table.

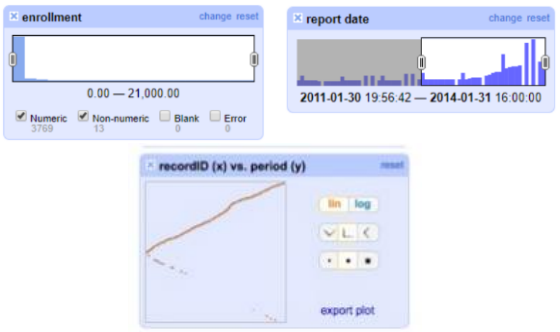

Figure 5: OpenRefine incorporates dynamic queries to filter the dataset through a feature it calls Facets/Faceting. The Numeric Facet (top left) shows a histogram of quantitative column values, and the Timeline Facet (top right) provides the same function for date column values. Both of these features filter the data through interval selection. The Scatterplot Facet (bottom) supports filtering based on two variables. This image comes from the OpenRefine Wikipedia, as my installed version of the application was never able to display the plotted data.

OpenRefine incorporates visualization into its interface through its Faceting feature. "Faceting" in open refine refers to interactive visualizations to support dynamic queries coordinated with the table representation of the data. These filter parameters can be combined with any other filter parameter to further refer the filters applied to the data. Figure 5 details the types of visualizations supported in Faceting. Textual facets display the unique values in a column sorted alphabetically with counts of the unique occurences on these items, and this type of facet does not incorporate visualization. Users can select any combinations of values to filter the dataset. Numeric Facets are essentially histograms enabled with interval selection to support dynamic queries. The column must be entirely comprised of quantitative data for this facet to work. The Timeline Facet provides a similar view of the distribution for column comprised of data objects. While the previous Facets only visualized one column of data, the Scatterplot Facet visualizes two columns of quantitative data. However, this feature never worked properly in the course of this analysis. The OpenRefine Wiki also does not offer any support for troubleshooting this issue.

5.3 Google Cloud Dataprep

Although Google Cloud Dataprep is branded as its own application, it is actually an instance of Trifacta Wrangler running on the Google Cloud Platform. Trifacta Wrangler/Dataprep is also the commercial descendent of the original Wrangler interface by Kandel et al. [7]. Thus, that particular application has been excluded from this analysis. I assume that the original authors would encourage journalists to use Dataprep or Trifacta Wrangler instead of the original Wrangler application for actual data wrangling work. In this analysis, I will refer to the shared interface as Dataprep even if it is identical to Trifacta Wrangler and extremely similar to the original Wrangler.

Like OpenRefine, Dataprep is a GUI application for wrangling data structured around a spreadsheet/table view of the data.

The only substantial difference between the Trifacta Wrangler and Dataprep seems to be the underlying computer architecture. Dataprep's architecture uses services from Google, such as their Cloud Storage product for the underlying raw data. But these architectural differences between these two products have a subtle but trifling impact on the usability of the user interface, which is the primary scope of this project. For example, exporting the final output of the Long-term Managed Care on Google Cloud's shared infrastructure took an average of 5.5 minutes (over eight trials) even though the output file is only 267 KB. But system level concerns about speed or data scalability are outside the scope of this project.

Figure 6: The Google Cloud Dataprep interface loaded with raw data from the Long-term Managed Care workflow. Dataprep incorporates more visualization into its interface than OpenRefine and the final wrangling output contained fewer errors.

Dataprep possess one unique feature that distinguish it from other wrangling applications. It suggests possible transformation for the user to apply to the dataset based on previous data transformations using a proprietary recommendation algorithm. Similarly, it will suggest transformations based on sections of text

high-lighted by the user. Interacting directly with a table is a novel interaction technique for specifying table transformations. The status quo for creating transformation specification is by either navigating through menu-items and toolbar buttons or specifying transformation in the underlying the transformational language.

While OpenRefine leverages visualization in a way that primarily supports filtering and secondarily supports exploration. Dataprep incorporates visualization for visual data profiling through two idioms: color stripe, figure 8, and visualizing column distributions, figure 9. Both of these idioms are highly relevant to cognitive dimensions of the interface while performing wrangling tasks and are discussed in depth in the analysis section.

6. Analysis

This section discusses how cognitive dimensions apply in the process of data wrangling by considering the most salient dimensions each on in turn, situating it in the context of other domains for illustrative purposes, and comparing and contrasting OpenRefine and Dataprep along these dimensions. The ultimate aim of this section is not to provide an exhaustive examination of cognitive dimensions in wrangling but to "coax out" convergent design features by compare the two applications with discussion grounded in this cognitive dimensions framework. Thus, dimensions that illustrate significant differences and similarities between the two interfaces comprise the majority of the discussion in this section.

6.1 Error-Proneness

According to Blackwell and Green [1], *error-proneness* in a notation aims to capture areas where a design features lead to systematically occuring errors, especially those where the notation does not offer protection from committing them. This class of errors excludes simple mistakes and slip-ups, and usability analyses drawing on the cognitive dimensions framework note that the distinction between the two is not clear [2]. While Blackwell and Green do not elaborate on methods for differentiating between the mistakes and serious errors, usability studies that use quantiative methods to identify statistically significant occurrence of errors may be one approach.

One way to elucidate *error-proneness* in wrangling interfaces is to consider mismatches across the Gulf of Execution, which Hutchins et al. [4] describe as the distance between the thoughts and goals of the user and the commands specified to the system. When this distance is zero, then system appears to behave entirely as expected, and errors constitute occasions when the system appears to act in ways other than what the user intended. These type of errors naturally break into two categories: false positives and false negatives. Table 2 breaks down these two errors in the context of removing observations from a dataset.

	Observation was deleted	**Observation was retained**
Observation should be removed	Success	False negative
Observation should be retained	False positive	Success

Table 2: Errors in the wrangle process can often surface from mismatches between what the user intends to accomplish and the specification of the notation.

Because each workflow includes the raw data and the final, wrangled output, there exists a "ground truth" wrangling results to compare the end product of wrangling with both direct-manipulation applications. Errors in the output reproduced by the applications then constitute discrepancies between my output and that produced by the journalists. Supplementary Materials contains one example script of this diff produced in OpenRefine and Dataprep. While it is possible for our reproduced wrangling to find data quality issues undiscovered in the original workflow, this was the case for neither of the two workflows considered in this analysis; however, prior work did discover errors in other workflows.

One source of *error-proneness* in wrangling applications occurs in a common *secondary notation*, regular expressions. Errors that result from regular expressions inevitably result from an imprecise expression manually entered by the user. These errors can occur in any domain, not just data wrangling. But, both Dataprep and OpenRefine incorporate design features to mitigate these errors since the specification of regular expressions is central to accomplishing many common wrangling tasks, including T1 and T3.

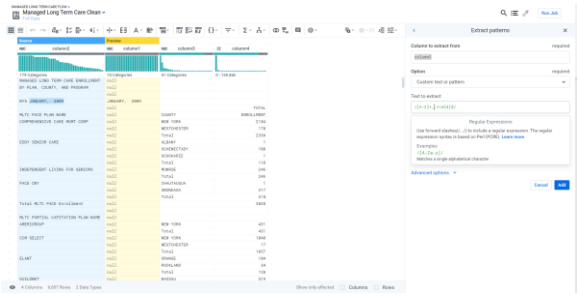

Figure 7: This figure illustrates extracting column values in the Long-term Managed Care workflow using Dataprep. With thousands of total columns, this process can be extremely error prone. It is up to the interface to guard against this kind of error even though it revolves around the use of secondary notation.

Error-prone behavior in regard to these two tasks involves not checking all the variables or observations that match the regular expression and all those that do not match. This sort of exhaustive search is the only way to ensure that the distance between what the user intended and what was specified in the notation is zero. The time complexity of this operation is linear to the size of the input. While such complexity is desirable for computational processes, this process is strictly a human activity become prohibitively expensive when the dataset contains hundreds of observation. Design features that increase visibility can aid in mitigating these errors and section 6.4 elaborates on them further.

6.2 Secondary Notation

The dimension of *secondary notation* refers to supplementary information separate from the official syntax [1]. In the usability analysis of programming languages *secondary notation* refers to comments and indentation. These features non-essential components intended to assist the user in completing a task, and wrangling applications also contain many of these supplementary features.

If we are to interpret the data order in rows and columns as "official syntax" in the context of user interface evaluation, *secondary notation* in wrangling applications incorporate three forms of *secondary notation* into their interfaces: regular expressions, menus, programming languages, and visualization. Regular expressions are a concise specification for a search pattern in textual data often used in wrangling for extracting components of values in categorical data (T1) and specifying which rows to remove based on column values (T3). OpenRefine also allows the user to write column-extraction or row-matching specification in Python or General Refine Expression Language (GREL). Finally, both OpenRefine and Dataprep incorporate visualizations beyond a large table display of the data being wrangled into their interface.

Within the category of *secondary* notation visualization constitutes *redundant recoding*, a channel of information that is "separate and easier channel for information that is already present in the official syntax" [3]. In my analysis the presence of visualization had a strong effect on *error-proneness* when it came to accomplishing two tasks: extract value from column (T1) and remove observations (T3).

Although regular expressions, menus, and programming languages are often used in wrangling interfaces to match table rows to filter and for extracting values from columns, the presence of visualization in Dataprep had the greatest impact on reducing *error-proneness*. The final analysis using Dataprep caught errors missed in OpenRefine because they were flagged during wrangling with the data-profiling visualizations in Dataprep. This is largely due to the color stripe present at the top of each column, as illustrated in figures 7 and 8.

Figure 8: Color stripes such as the one picture above adorn the top of each column in Dataprep.

In terms of what-how-why analysis, this color stripe idiom visualizes the proportion of data quality categories in an individual column with stacked line marks to encode proportion and color to encode category. There are three data quality types: valid, which signals the column values match the column data type specified by the user; mismatched, such as having alphabet characters in a column of quantitative variables; and missing values, often denoted as NULL, NA, or left as an empty string. These are especially useful for finding data quality issues.

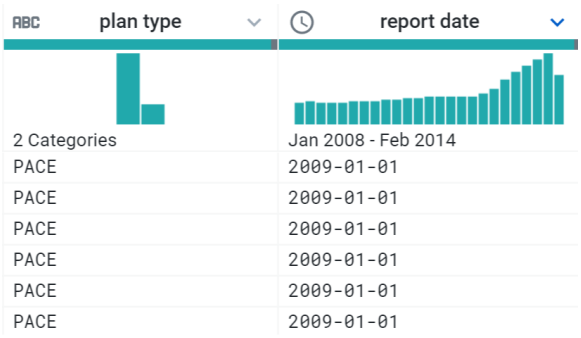

Figure 9: Beneath the color stripe, shown in figure 8, the bar charts and histogram charts provide another useful *secondary notation* for identifying data quality issues in Dataprep.

Another useful visualization that didn't directly lead to finding bad rows but gave me a sense of confidence in the data were bar charts and histograms of the column distribution, as show in figure 9. In terms of what-why-how analysis, both of these idioms visualize the distribution of values within a column using line marks along the vertical position and position along the horizontal column to identify which values deviate from "normal." In histograms, the keys are binned ranges within the underlying data and in bar charts the keys are the unique levels of the categorical data. Both visualizations hope to signal deviations in the data from "normal," and require domain expertise to distinguish valid and invalid values.

Histograms may be able to detect a nefarious data quality issues that this analysis project did not address because it was absent from the two workflows. A common data quality issue involves values within the same column on different scales of magnitude. For example, a table value may encode one million dollars as "1,000,000" or it may denote it as "1" with it being implied that the values are in the millions. A bimodal histogram may signal that this type of error exists in the underlying data, and domain expertise is necessary to confirm that such values are actual errors.

6.3 Provisionality

Blackwell and Green define *provisionality* as a one's commit actions made or a notation system's ability to support speculative operations or "what-if" games [1]. Although not a notation, the shades of *provisionality* can best be explained by different types of writing instruments. The marks made by pencils and dry-erase markers have a high degree of *provisionality* because they can be easily erased, but marks made with permanent markers, pens, and tattoo guns have low *provisionality* because they cannot be easily erased.

In wrangling applications, *provisionality* occurs in a system in two ways: previews and undo. First, a system can preview to the user the results of a transformation. Second, the system can provide easy ways to recover from actions committed, much like a pencil with an eraser. Previewing is an important feature for operations that address both rows and columns. Both OpenRefine and Dataprep support preview and an undo features. The fact that both of these interfaces have converged upon these features signals that they are important in wrangling applications.

Related work in data wrangling that evolved into Dataprep supports previewing. Wrangler [7] supports provisionality in both tasks through juxtaposing the table before and after the transformation with color linking and transparent overlays to preview the results.

The value extraction task (T1) always incorporates *secondary notation* as specification of the extraction method. Hence, column transformation previews also enable *provisionality* of this other notation. This feature is essential when specifying extraction with Regular Expressions in both wrangling applications because it guards against the *error-proneness* of this notation.

Provisionality can greatly assist in wrangling tasks involving table transformations, especially when extracting values from a column (T1). Both OpenRefine and Dataprep generate a provisional column filled with the output of the extraction method. Every change in the extraction specification updates this preview column, and committing the operation is essentially making this column a concrete variable of the data.

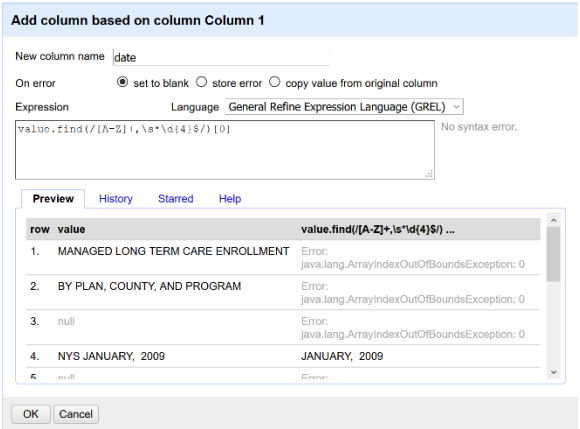

Figure 10: Preview features of the output from column extraction (T1) in OpenRefine reflect with a high degree of provisionality. In

this example of wrangling in the LMC workflow, the application is extracting the date of a report from a column with mixed variables: date, plan name, and plan type. The same feature is illustrated in Dataprep in Figure 7.

While both OpenRefine and Dataprep support previewing, they implement this feature differently, as figures 7 and 10 illustrate. OpenRefine generates a preview modal dialogue window. Dataprep situates a column within the table display of the data, color codes the column that is the source in blue and the preview column in yellow and highlights the match in the source column. The ability to preview the effects of an action is one way to increase *provisionality* and recovering from a committed action is another method.

Both OpenRefine and Dataprep also support recovering from an actual with an undo feature situtated within a list detailing the sequences of table transformations applied to the raw data. Dataprep supports this application a little better than OpenRefine. While both applications enable the user to edit a previous table operation, delete a transformation, and reorder transformations. Dataprep supports these operations within the recipe panel. OpenRefine supports this through directly editing the JSON file containing, which is a *secondary notation* of the system. Dataprep allows the user to temporarily disable individual transformations but not delete them from the history. OpenRefine does not support this since JSON does not support comments. A user may copy the specification for the table transformation in another application, such as Notepad. But this strategy still constitutes deletion in the wrangling application.

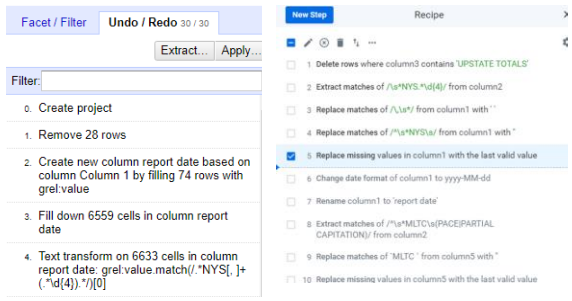

Figure 11: Both applications record an auditable history of the table transformations applied to the raw data. The left figure shows history in OpenRefine and the right figure show history, also known as recipes, in Dataprep. This features increase the provisionality of the interface by not forcing the user to commit to previously committed table transformations.

Both applications could further increase the *provisionality* through this feature by supporting branching table transformation sequences. The wrangling provenance record in current use from both applications is strictly linear. Longer branches of what-if wrangling exploration could be reported by supporting a bifurcate actions in this history that allows the user to operate down a separate branch. Such a feature is similar to the speculative development avenue that are currently supported in software development projects using source code management tools such as SVM or Git. This additional feature would come with the tradeoff of increasing the number of *secondary dimensions*, and wrangling users would have to learn another subcomponent in a system that already uses many subcomponents.

6.4 Visibility

The cognitive dimension of *visibility*, also called *visibility and juxtaposability,* refers to how well system components can actually be seen by the user [1]. Before describing *visibility* in data wrangling applications, it is illustrative to briefly describe how usability analysis of notation in other domains interpret this dimension. In the domain of programming environments, Green and Petre describe a system with maximum *visibility* as one where every part of code is simultaneously visable [3]. For computer programs that are small enough, *visibility* is not an issue; however, as the size of the software project increases, *visibility* decreases, obviously.

As with programming environments, *visibility* in wrangling environments has the same inverse relationship with the environmental input. Where the dimensions of the tabular datasets are to data wrangling as lines of code are to computer programming. In wrangling, being able to view the sections of the data currently being transformed, and even those section not transformed, is the major issue of *visibility*. One obvious and ubiquitous interaction technique for overcoming *visibility* limitations with large datasets is to utilize scrolling. When the number of columns and rows of a table exceed the "real estate" afforded by the computer screen, then many wrangling applications and those that support wrangling utilize vertical and horizontal scrolling, respectively.

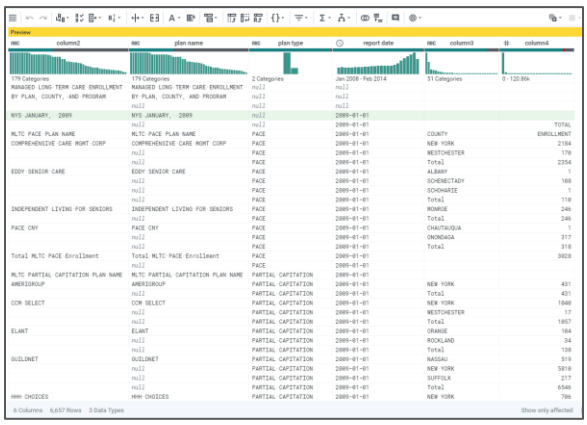

Figure 12: Dataprep uses color highlighting to denote rows of the table that will be removed with the implementation of the pending table transformation. While OpenRefine has the option of viewing either all the removed rows or all the retained rows, Dataprep situates the rows to be removed in the same view as the rows to be retained. However, the user can also specify to view just the rows to be removed to increase the *visibility* of the system.

Visibility is especially important when filtering rows (T3) to give the user confidence that the transformation is operating upon the rows they want and not on the ones they want to be retained. Both applications show the rows that will be removed and the rows that will not be removed. However, Dataprep's implementation of this feature is superior because columns to be removed are situtated within the table along with the rows that will be retained. OpenRefine only allows the user to view one or the other. Thus confirming that the system is removing exactly what you intended it to remove in Dataprep requires less human memory than in OpenRefine, which concurs with the rule of thumb in Visualization Analysis and Design that "Eyes Beat Memory" [9].

Visibility also has a difficult-to-access relationship between visualization as a *secondary notation*. Visualization can effectively "show" users a dimensional subset of their data in a much smaller space. While scrolling through a column of thousands of values is one way to see all the data, most people would probably prefer viewing a histogram of the distribution within the column, depending upon the task.

Visualization for wrangling also addresses a common problem of how to view a large dataset when screen "real estate" is limited. Related work has addressed this issues specifically in queries on large datasets. VisDB [8] visualizes the query specification process in a database, and compresses each database record as one pixel to represent a large table of data on one screen. However, visualization intervention in the wrangling applications considered in this project attempts to visualize the schema of a table instead of each individual data point.

The book Visualization Analysis & Design [9] classifies three design choices for reducing the amount of data shown within one view and thus can also address the common problem in data wrangling. First, filtering can reduce the amount of data presented. Second, aggregation can reduce the size of the data by combining many observations into one. This design choice effectively coarsens the dataset. Finally, embedding describes providing an additional view of the data triggered by the user interacting with the dataset. Elided data, where some data is filtered and others are summarized, is one example of embedded data. The two wrangling applications considered largely utilize aggregation design choices through bar charts and histograms, elaborated upon in section 6.2 on *secondary notation*.

One short coming of filtering through Faceting in OpenRefine is that there is no way to change the granularity of the visualization. For example, the Long-term managed care workflow requires the user to filter the data for just summer months in 2013, 2015, and 2016. When attempting to accomplish this task with the Timeline Facet, the user isn't able to select the specific month. The feature does not support semantic zooming to give the user the control they need to accurately specifying the transformation.

Visualizations of the distribution of values within a column can assist in locating data quality issues that were not included in the two workflows analyzed but were present in other workflows from prior work. Because much public data originates from manual data entry into a form or even a spreadsheet, data quality issues can result from human error. Both bar charts of categorical data and histograms of quantitative data can support the task of finding values that do not conform to the trend of the data in general.

The visibility of the data being operated upon becomes crucial when accomplishing tasks T1 and T3 as it can leads to errors characterized by mismatches between what the user intends and what is specified in the systems notation. Section 6.1 elaborated on this issues and *error-proneness* in wrangling applications, in general. Both applications implement a preview feature that provide an interim representation of the

underlying dataset if the table transformation currently being considered is executed. This feature is a clear example of leveraging *provisionality* in an interface and is further described in section 6.3.

In Dataprep, only 34 table rows can be viewed at one time. Column widths can vary from table to table. LMC is the highest dimensionality dataset in this project with six columns at the conclusion of the wrangling processes, and this table just barely fits in the window allotted. Hence, only a small subset of data is directly visable to the user at any given moment.

In pursuit of T3, remove observations, the visibility of which table rows are removed and which are retained has a significant impact on *error-proneness* of a wrangling application. Dataprep possesses a unique feature that makes it superior to OpenRefine when removing rows in this task. Dataprep allows the user to toggle between viewing table rows that match the filter criteria and will be removed and those that do not match and will be retained. OpenRefine only displays the table rows that will be removed, placing the onus on specifying what is retained on the user's memory, hence increasing the cognitive demand of the task.

6.5 Other Dimensions

There are few cognitive dimensions not worth discussing in much detail within the context of two direct-manipulation interfaces.

By design, direct manipulation interfaces have high degrees of *progressive evaluation* and *closeness of mapping*. The ability for current progress to be checked at any time is *progressive evaluation* [1]. Direct-manipulation interfaces provide a high degree of this dimension by default. At any stage of wrangling in both OpenRefine and Dataprep, the user is able to see their current progress. Wrangling in a programming environment has a lower degree of this dimension. In prior work studying wrangling notebooks from journalists, users frequently inspected the current state of the table after a transformation operation, which was coded as *peek at data*. Likewise, interfaces that facilitate wrangling through interactions with a table representation of the data enjoy a high degree of *closeness of mapping*, the closeness of the representation and the domain [1].

Viscosity is defined by Blackwell and Green [1] as the amount of effort necessary to perform a single change. In chemistry, viscosity is an expression of the resistance to flow of a system under stress, and the cognitive dimensions framework repurposes this term to mean how resistant a system is to changes. When applied to data wrangling, it means the amount of work performed to transform a table into a particular structure. This dimension manifests itself when removing many columns from the dataset, which falls under *Trim the Fat* (T10). Both interfaces converge on a way to select multiple columns for removal at one time. This feature is extremely convenient as the California Conservation Score workflow requires the user to remove more than 30 columns from the raw data. Removing columns individually would constitute an unnecessary amount of effort to perform a single task, which is remove all unnecessary data variables.

7. Discussion and Future Work

In the future, it may prove fruitful to widen the scope of wrangling applications considered Tableau Prep, Workbench, and Microsoft Excel. Although software products from the data visualization company Tableau are not free by default, the company provided journalists with complimentary licenses of Tableau Prep, its data wrangling application, since 2018 along with Tableau Desktop, its flagship visualization and analysis application [14]. Like Dataprep/Wrangler Trifacta, Tableau Prep also uses univariate visualization idioms to profile the underlying data for error detection and provides a table view of the dataset being wrangled. However, this product has a unique network view of wrangling provenance. Workbench is another wrangling product especially relevant to journalism because it was built for journalists in mind. It was initially launched in 2017 as an "integrated data journalism platform that makes it easy to assemble data scraping, cleaning, analysis, and visualization tasks without any coding" [17]. While it appears to follow similar data wrangling conventions to OpenRefine and Dataprep, a more in-depth analysis may shakeout its strengths and weaknesses in this crowded field of data wrangling products. It has been said that 90% of data journalism is done in Excel [2]. This product's ubiquity may be due to its default presence on many newsroom computers [13].

8. Conclusion

In the end, this report that OpenRefine and Dataprep share much of the same core functionality; although Dataprep use of visualization lead to less error-proneness in process and high-quality data at the conclusion of the process. Wrangling itself can be a highly error prone activity. But according to the

cognitive dimensions framework, we cannot simply decrease the *error-proneness* of a system, each dimension is intimately connected to other dimensions. Thus, incorporating visualization as a *secondary notation*, increases the complexity of the system but the user gains the ability to see potential data quality issues that were previously only accessable from manually scrolling through the table. Increasing the *provisionality* of the interface, both in previewing the results of a table transformation and recovering from committed results is another "knob" that the tool builder can tweak to decrease *error-proneness*.

This analysis project only considered two direct-manipulation applications that incorporate visualization and incorporating more wrangling applications and more tasks will provide a better understanding of the tradeoffs inherent in designing tools for data wrangling.

Bibliography

[1] A. Blackwell and T. Green, "Notational Systems—The Cognitive Dimensions of Notations Framework," in *HCI Models, Theories, and Frameworks*, Elsevier, 2003, pp. 103–133.

[2] Global Investigative Journalism Network, "Nils Mulvad - Excel is 90% of data journalism - YouTube," *YouTube*. [Online]. Available: https://www.youtube.com/watch?v=aahUKhuB9Bw. [Accessed: 06-Dec-2019].

[3] T. R. G. Green and M. Petre, "Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework," *J. Vis. Lang. Comput.*, vol. 7, no. 2, pp. 131–174, Jun. 1996.

[4] E. L. Hutchins, J. D. Hollan, and D. A. Norman, "Direct Manipulation Interfaces," *Hum.-Comput. Interact.*, vol. 1, no. 4, pp. 331–338, 1985.

[5] D. Huynh, *Open Refine*. 2012.

[6] S. Kandel *et al.*, "Research Directions in Data Wrangling: Visualizations and Transformations for Usable and Credible Data," *Inf. Vis.*, vol. 10, no. 4, pp. 271–288, Oct. 2011.

[7] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer, "Wrangler: Interactive Visual Specification of Data Transformation Scripts," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2011, pp. 3363–3372.

[8] D. A. Keim and H.-P. Kriegel, "VisDB: database exploration using multidimensional visualization," *IEEE Comput. Graph. Appl.*, vol. 14, no. 5, pp. 40–49, Sep. 1994.

[9] T. Munzner, *Visualization Analysis and Design*, 1st ed. A K Peters/CRC Press, 2014.

[10] V. Raman and J. M. Hellerstein, "Potter's Wheel: An Interactive Data Cleaning System," in *Proceedings of the 27th International Conference on Very Large Data Bases*, San Francisco, CA, USA, 2001, pp. 381–390.

[11] J. Stray, "Network Analysis in Journalism: Practices and Possibilities," *Proc KDD*, p. 8, Aug. 2017.

[12] Subscribe, "From Freebase Gridworks to Google Refine and now OpenRefine." .

[13] S. Sunne, "Diving into Data Journalism: Strategies for your newsroom," *American Press Institute*, 09-Mar-2016. .

[14] S. Teal, "Journalists: Now Tableau Prep is free for you," *Tableau Public*, 26-Apr-2018. [Online]. Available: https://public.tableau.com/en-us/s/blog/2018/05/journalists-now-tableau-prep-free-you. [Accessed: 06-Dec-2019].

[15] M. Tulio Pires, "Preparing data," *Data Journalism and Visualization with Free Tools*, 21-Oct-2019. [Online]. Available: https://journalismcourses.org/course/view.php?id=44&section=3.

[16] H. Wickham, "Tidy Data," *J. Stat. Softw.*, vol. 59, no. 1, pp. 1–23, Sep. 2014.

[17] "Data Journalism Made Easier, Faster, and More Collaborative," *Medium*, 07-Jul-2018. [Online]. Available: https://medium.com/@Workbench/data-journalism-made-easier-faster-and-more-collaborative-e33081bf0080. [Accessed: 06-Dec-2019].

Supplementary Materials

California Conservation Score Task Sequence

1. **Format Display**
   Format Tables to display: at most 500 columns and floats have two significant figures.

2. **Trim the fat**
   Import uw_supplier_data100516.xlsx but only keep the columns with supplier name, month, total water production in gallons, total water production in gallons in 2013 and residential water usage. Note: the total records in this table should be 10,936

3. **Filter dataset by data range**

4. **Convert column type**
   Work the month variable into a filterable format. In the analysis in Python by the *Los Angeles Times*, they converted it to a string.

5. **Remove the rows**
   Keep only unique observations from three summer months (June, July, and August) in 2015 and 2016.

6. **Remove duplicates**
   Note: the total records in month table should be 2,425

7. **Remove incomplete data**
   Eliminate any suppliers from the month table dataframe who have fewer or greater than six months of data with those labels.
   Hint: There are 19 incomplete suppliers
   At the end of this process there should be 2,334 records.

8. **Aggregate dataset**
   Make a new table from the month table, it could be called summer table, that groups and sums the total water production for each summer.

   Hint: The between-table context way to do this is by separating into many tables, and create separate tables summing the total water production gallons for 2015, 2016, and 2013, grouped by supplier name.

   Compare the total number of records for each new table. It should be 389 for each. Count the total number of records in summer table. It should be 389 records for each sub table.

9. **Derive percentage change**
   Derive the change in savings between 2015 and 2016 in the summer table as a new variable. In order to calculate this measure, you'll have to derive two intermediate variables, the percent change of summers 2015 and 2016 versus the baseline of summer 2013 as a new variable, and the savings change in summer table. The final product should look like this.

   Optionally, rank cities that have regressed the most towards their 2013 baseline. I would consider this analysis, however.

10. **Calculate mean**
    Calculate mean residential water usage in 2016 for each supplier in summer table. Calculate the average

monthly water usage per person (R-GPCD) in each district for the summer of 2016 as a new table called summer 2016 means.

11. **Aggregate join**
    Aggregate join average monthly water usage per person (R-GPCD) on summer table just for 2016.
    Join those water usage average to our combined table called summer table
    Create a summary statistic to judge how many districts regressed in summer 201

12. **Calculate performance metric**
    Calculate a "Conservation-Consumption Score" that adjusts the savings change by the amount of water usage to surface the high-usage districts that regressed the most. Add this value to summer table.
    Look at the top ten districts by this score

13. **Export the table**

Long-term Managed Care Sequence

1. **Remove Upstate Total rows**
   Remove Upstate Totals

2. **Extract date**
   Extract the report month and year

3. **Extract Plan type**
   Extract the plan type

4. **Extract Plan Name**
   Create a plan name variable

5. **Find and remove all junk rows:**
   Remove the many junk rows with variables in plan name about:

   a. Report dates
   b. Plan names
   c. Notes about the plan
   d. BY PLAN
   e. Statewide Totals
   f. Plan totals

6. **Resolve plan names**
   Resolve plan name entities

7. **Tidy the dataset**
   Transform the dataset into a true Tidy dataset and rename the columns

8. **Calculate plan totals per month as a new variable**
   Calculate total enrollment by plan name, added as new variable. This is done by aggregate join.

9. **Remove rows with missing county**
   These have the provider name WELCARE.