Ways with data: Understanding coding as writing

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Chris Lindgren, Ph.D.

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Dr. Christina Haas

May, 2017

Acknowledgements

I am amazed by how much love and labor makes for a completed dissertation. I am indebted to my advisor and mentor Christina Haas. Thank you, Professor Haas, for pushing my limits, being such a good listener, and helping me craft this research agenda. I couldn't have gotten this far this quickly without your keen eye for method, generosity with your time (and personal library!), and tenacity for sharpening disciplinary arguments.

Thank you to my amazing cohort and friends: Kira Dreher and Rachel Tofteland-Trampe. Very few days pass without me appreciating how I had the privilege of sharpening my ideas and writing with such brilliant and kind people. I know this project (and my job materials!) improved after every cup of iced coffee and some banter at the Riverside Cafe.

Thank you to my dissertation committee: Pat Bruch, Richard Graff, Laura Gurak, and Sashank Varma. Your questions and suggestions about my work throughout my time at UMN set this project on such a great disciplinary path.

A special thanks to my friend and colleague Justin Schell. I can't thank you enough for helping me expand my horizons at such a huge institution that is UMN. You ushered me across invaluable cross-disciplinary terrain and helped me meet amazing people doing amazing work that is connected to my own.

Thank you to my family; especially, my partner, Laura Lindgren. Your support was immeasurable, and your my model for how to be a better person: both professionally and personally.

Thank you also to my 2 children, Ava and Gavin. Whenever things got rough, I could always count on both of you to help lift me up again.

Finally, thank you to “Ray” – the heart of this case-study. I had such a wonderful time working with you on this project. I will always appreciate your willingness to participate in this scholarly adventure.

Dedication

I dedicate this dissertation to all of the people who set out to understand coding as more than just some utility. For people who do the life-long work to qualify it as language that impacts people.

Coding as writing with data

```
var myDissertationClaim = new Map();

function writing() {
    return 'writing';
}

myDissertationClaim.set('coding', 'is');
myDissertationClaim.set(writing(), 'with data');

console.log([...myDissertationClaim]);
// Prints out [['coding', 'is'], ['writing', 'with data']]

for (let [key, value] of myDissertationClaim) {
    console.log(`${key}: ${value}`);
    // Prints out:
    // coding: is
    // writing: with data
}
```

Abstract

In this dissertation, I report findings from an exploratory case-study about Ray, a web developer, who works on a data-driven news team that finds and tells compelling stories with large sets of data. I implicate this case of Ray's coding on a data team in a writing studies epistemology, which is guided by the following question: *What might be learned about coding, if writing researchers explore the consequences of making language material and computational in a digital medium*? I begin this study by outlining a theory of materiality of writing through 6 propositions, which serve as a lens to review literature and theories about coding that articulate the characteristics of code as written communication. From there, I describe my grounded-theory approach to this exploratory case and the battery of ethnographic methods used to collect observational data of Ray's coding over the course of approximately 6 months. Next, I present findings from my grounded analysis across 2 chapters. The first findings chapter cultivates a thick description of Ray, his coding, and how his coding is embedded within a broader objective to find stories in and through aggregate information, which I call *aggregate narratives.* In the second findings chapter, I conduct a more granular analysis of Ray's coding of goal-oriented slices of data from the original data set source—a coding practice that produces what I term *provisional texts.* Findings indicate how Ray's coding of the provisional texts, and the texts themselves, provide active epistemic functions to create aggregate narratives. Finally, I conclude by synthesizing findings with the theoretical propositions about the materialities of writing discussed in the first chapter. Overall, Ray's coding and its materialities show how coding is a dynamic, situated cultural practice, which invites future inquiries into and across domains of coding practices.

Contents

List of Tables

List of Figures

Chapter 1

Introduction: Understanding code and data as texts and coding as writing

> [W]riting is always embedded in activity. It is always enacted toward goals; it is always shaped by circumstance, by agency, by history. Big-W Writing does not exist; it is always small-w writing because in human culture writing is not done for its own sake. ... [T]here is no place, it seems to me, for a notion that writing can be pristine, something in and of itself.
>
> –Christina Haas, Burnett & Haas, 2007, p. 34

1.1 Code as text, machine, both, or more?

Since deciding to become a writing researcher, I have often revisited the passage above from Haas' interview with Burnett (2007). I keep returning to the concept of writing as having no "place" to call its own – no space to neatly define "itself and of itself." Haas' insights about writing reminds me just how enmeshed writing is amongst the countless activities that makeup our everyday lives: that people make up the fabric of writing, which makes it a dynamic cultural and rhetorical practice unbound by any unifying theory. I find these ideas about the complexity of written language both fascinating and frustrating.

Such a position is also frustrating, because my own research agenda sets out to investigate what it is about our experiences with writing that constantly reproduces myths that writing is or can be ‘pristine’: a window into one’s mind and thoughts, or how it can be made plain, transparent, and self-evident. Writing and rhetoric scholars constantly create approaches and methods to re-articulate how these myths persist, developing new ways about how to respond to these questions about the nature of texts and written language.[1]

In this dissertation, I explore my fascination and frustration about writing by building on prior research that seeks to understand the consequences of making language material: studies on the materialities of writing (Haas, 1996; Wickman, 2010; Witte, 1992). Put generally, researchers in this field study how people are mediated by the material means of producing and using texts in coordination with they ways texts organize aspects and inscriptions about the material world. Haas refers to this link between the material and the symbolic as writing’s “central fact” and its “central puzzle” (p. 3).

Herein, I engage writing’s “central fact” and “central puzzle,” which I see spill over into conceptions and perceptions about computer coding. In the remainder of this chapter, I review ways coding has been taken to be understood more directly as writing. Such a review of coding-as-writing carves out space for my dissertation to more closely examine the question: *What can be learned about coding by studying it as writing?*[2] My dissertation puts this question within the frame of studies of the materialities of

[1] One needs to look no further than studies of literacy and writing to see how writing indeed has no place of its own, and instead people make it meaningful within the places of their own literacy (*cf.* Banks, 2006; Barton, 1991; Cushman, 2011; Grabill, 2001; Haas, 1996; Heath, 1982; Reynolds, 2007; Richardson, 2003; Scribner & Cole, 1981)

[2] As an important aside, my longer research goals include flipping this question around: *What can be learned about writing by studying coding?*

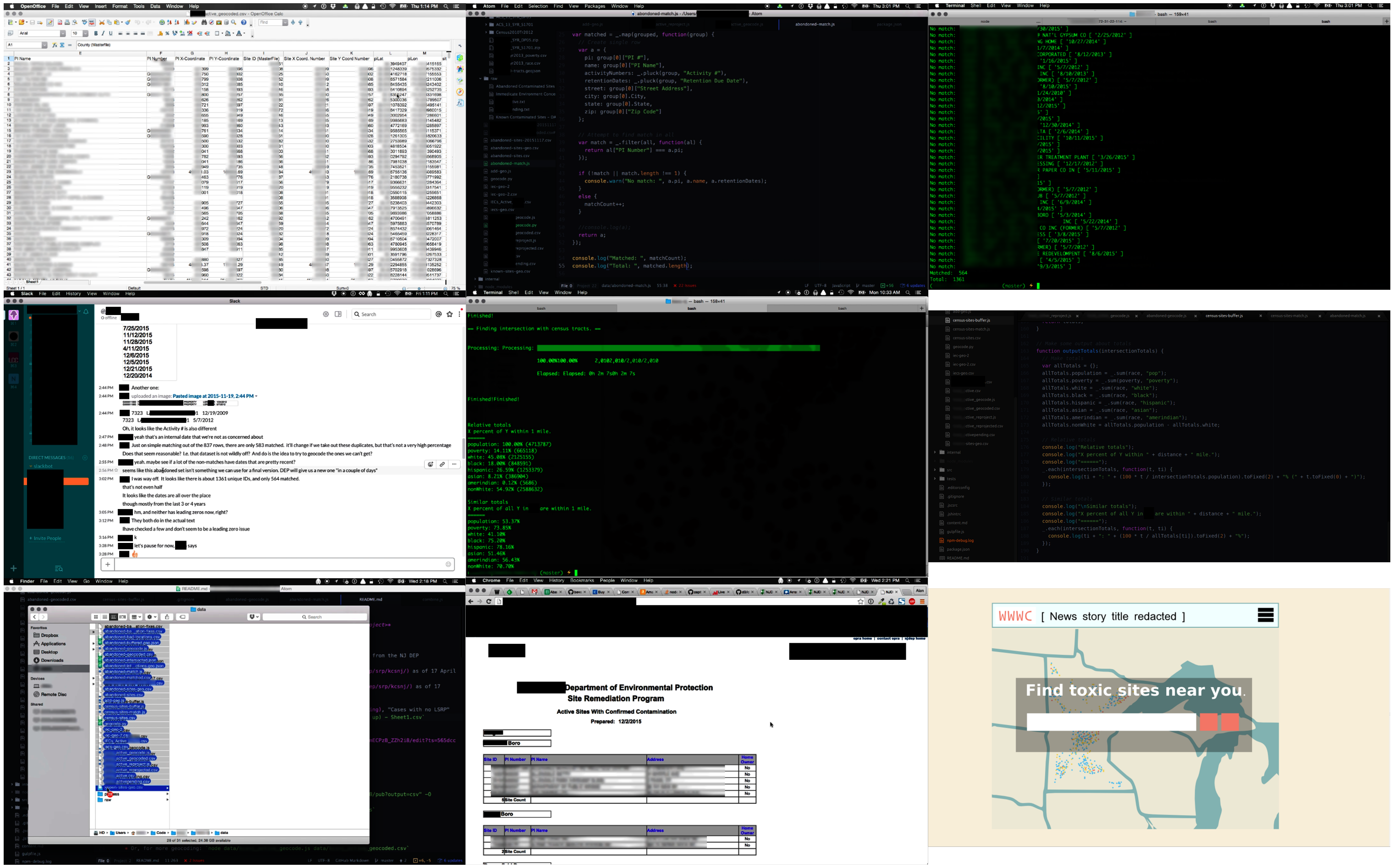
writing, which asks, *What might be learned about coding, if writing researchers explore the consequences of making language material and computational in a digital medium*?

In the remainder of this chapter, I introduce this subject of "What is code?" through a review of some of the prevailing discourse and scholarship that explores the broader questions about whether code is text, machine, both, or something more slippery and interesting. I engage with professionals and scholars across multiple domains, who discuss what code represents, what and how it performs in the world, and how it is or is not a direct referent of will, intent, thought, or deed. Before I do so, I provide a quick preview into the case-study that serves as the basis for my response to this exploratory research question.

1.2 A Case of Reading and Writing Code in a Newsroom

Within the first few weeks of observing Ray,[3] a web developer who works with reporters and editors in a newsroom, I saw how numerous types of texts and tools permeated his coding. Over the course of my 5 months observing Ray, I watched him code on numerous kinds of projects—each with their own exigencies and objectives. (Refer to Figure 1.1 below for a composite image of various types of texts and technologies represented throughout these aforementioned situations.)

[3] All names, places, and artifacts have been changed, altered, or scrubbed for identifying information for confidentiality purposes in this IRB-approved study.



```
var matched = _.map(grouped, function(group) {
  // Create single row
  var a = {
    pi: group[0]["PI #"],
    name: group[0]["PI Name"],
    activityNumbers: _.pluck(group, "Activity #"),
    retentionDates: _.pluck(group, "Retention Due Date"),
    street: group[0]["Street Address"],
    city: group[0].City,
    state: group[0].State,
    zip: group[0]["Zip Code"]
  };

  // Attempt to find match in all
  var match = _.filter(all, function(al) {
    return al["PI Number"] === a.pi;
  });

  if (!match || match.length !== 1) {
    console.warn("No match: ", a.pi, a.name, a.retentionDates);
  }
  else {
    matchCount++;
  }

  //console.log(a);
  return a;
});

console.log("Matched: ", matchCount);
console.log("Total: ", matched.length);
```

```
No match:
No match:
No match:
Matched:  564
Total:  1361
```

7/25/2015
11/12/2015
11/28/2015
4/11/2015
12/6/2015
12/5/2015
12/21/2015
12/20/2014

2:44 PM [redacted] Another one:
2:44 PM [redacted] uploaded an image: Pasted image at 2015-11-19, 2:44 PM
2:44 PM [redacted] 7323 L[redacted]1 12/19/2009
7323 L[redacted]1 5/7/2012
Oh, it looks like the Activity # is also different
2:47 PM [redacted] yeah that's an internal date that we're not as concerned about
2:48 PM [redacted] Just on simple matching out of the 837 rows, there are only 583 matched. it'll change if we take out these duplicates, but that's not a very high percentage
Does that seem reasonable? I.e. that dataset is not wildly off? And do is the idea to try to geocode the ones we can't get?
2:55 PM [redacted] yeah. maybe see if a lot of the non-matches have dates that are pretty recent?
2:56 PM seems like this abandoned set isn't something we can use for a final version. DEP will give us a new one "in a couple of days"
3:02 PM [redacted] I was way off. It looks like there is about 1361 unique IDs, and only 564 matched.
that's not even half
It looks like the dates are all over the place
though mostly from the last 3 or 4 years
3:05 PM [redacted] hm, and neither has leading zeros now, right?
3:12 PM [redacted] They both do in the actual text
Ihave checked a few and don't seem to be a leading zero issue
3:16 PM [redacted] k
3:28 PM [redacted] let's pause for now, [redacted] says
3:28 PM [redacted] 👍

```
Finished!

== Finding intersection with census tracts. ==

Processing: Processing:

                        100.00%100.00%      2,0102,010/2,010/2,010

                        Elapsed: Elapsed: 0h 2m 7s0h 2m 7s


Finished!Finished!


Relative totals
X percent of Y within 1 mile.
======
population: 100.00% (4713787)
poverty: 14.11% (665118)
white: 45.08% (2125155)
black: 18.00% (848591)
hispanic: 26.59% (1253379)
asian: 8.21% (386904)
amerindian: 0.12% (5686)
nonWhite: 54.92% (2588632)

Similar totals
X percent of all Y in    are within 1 mile.
======
population: 53.37%
poverty: 73.85%
white: 41.10%
black: 75.20%
hispanic: 78.16%
asian: 51.46%
amerindian: 56.43%
nonWhite: 70.70%
```

```
}

// Make some output about totals
function outputTotals(intersectionTotals) {
  // Make totals
  var allTotals = {};
  allTotals.population = _.sum(race, "pop");
  allTotals.poverty = _.sum(poverty, "poverty");
  allTotals.white = _.sum(race, "white");
  allTotals.black = _.sum(race, "black");
  allTotals.hispanic = _.sum(race, "hispanic");
  allTotals.asian = _.sum(race, "asian");
  allTotals.amerindian = _.sum(race, "amerindian");
  allTotals.nonWhite = allTotals.population - allTotals.white;

  // Relative totals
  console.log("Relative totals");
  console.log("X percent of Y within " + distance + " mile.");
  console.log("======");
  _.each(intersectionTotals, function(t, ti) {
    console.log(ti + ": " + (100 * t / intersectionTotals.population).toFixed(2) + "% (" + t.toFixed(0) + ")");
  });

  // Similar totals
  console.log("\nSimilar totals");
  console.log("X percent of all Y in [redacted] are within " + distance + " mile.");
  console.log("======");
  _.each(intersectionTotals, function(t, ti) {
    console.log(ti + ": " + (100 * t / allTotals[ti]).toFixed(2) + "%");
  });
}
```

opra home | contact opra | njdep home

[redacted]Department of Environmental Protection
Site Remediation Program
Active Sites With Confirmed Contamination
Prepared: 12/2/2015

[redacted]Boro

Site ID	PI Number	PI Name	Address	Home Owner
				No
				No
				No
				No
				No
5	Site Count			

[redacted]Boro

Site ID	PI Number	PI Name	Address	Home Owner
				No
				No
2	Site Count			

Figure 1.1: Composite image that includes multiple screen captures of moments during Ray's coding activity. All images have been scrubbed for private identifiers and the bottom-far-right image is a facsimile of one initial screen from an interactive map.

Sometimes a reporter came to Ray with a table of computed values, asking him to verify the data and subsequently develop it into an interactive visualization to embed in a news report. Other times, reporters had only the beginnings of story and multiple large sets of government data. In such a case, Ray and his coding work served as a resource to help reporters develop some data-driven dimension to their developing news narrative. Other times yet, I observed Ray code to collect or import data into databases; databases that collected and produced troves of data sets on dedicated networked servers. These archives dealt with myriad concerns and issues, such as state-level data on recidivism or political campaign finances, or city-level data that tracks public transportation headway times or indexes city government employee salary information.

During other occasions, I observed how Ray's coding developed particular tools: A mapping tool for reporters to make embeddable maps in their stories or a texting campaign application that surveys people about their experience with information on the Web. Even in these more traditional technical tooling projects, I gathered a sense that Ray's coding was closely integrated with the editorial desire to collect, create, and use the increasing amount of publicly available data. Over time I began to see how his coding played an integral role in developing structured data as a kind of text to support his coding work in dynamic ways.

My study of Ray's coding as writing illuminates how coding does not singularly get its power from the computer and its processing speed, nor from any particular programming languages and their ability to express and perform computational tasks. Instead, much of the power of coding and its textual nature can be seen as Ray's ability to put these properties of coding into dynamic play with their numerous ways with data within the context of his particular objectives. *This case-study of Ray's coding provides a robust picture about his creative figuring of data through acts of writing code; namely, how structured data mediated his coding to make sense of the coding project at-hand.* In what follows, I review how 1) people have been developing related ideas about code as texts through different discursive lenses, and 2) how programming languages have developed into more naturalized and abstract writing systems. From there, I begin to integrate such positions about coding and writing into to my research problem: the struggle to understand coding and its materialities, or as Haas put it: "always embedded in activity . . . always enacted toward goals, . . . [and] is always shaped by circumstance, by agency, by history."

1.3 Studies on the materialities of writing

In response to the research problems above, my study brings together a particular lineage of writing theory (Haas, 1996; Scribner & Cole, 1981; Wickman, 2010; Witte, 1992) to bear on Ray's coding activity. This lineage accounts for the core of what I deem integral studies on the materialities of writing, and they introduce my research in a few ways. First, my research findings contribute directly to this lineage and their general propositions about writing and its materialities. By reviewing this scholarship, I can better refine what I mean by writing within the frame of my research question: *What can be about coding, if writing researchers explore the consequences of making language material and computational in a digital medium?* Second, by identifying their cumulative general propositions, I can make my contributions to writing theory more salient later in the dissertation. Third, and perhaps a synthesis of the previous two reasons, this scholarship helps me develop an epistemological frame to interpret other scholarship about code-as-writing reviewed in this chapter. The materialities of writing provide a discursive lens by which to understand how existing knowledge about the textual dimensions of code relate to my study.

To open up perceptions about what counts as texts and textual work, I begin with Witte's (1992) definition of texts, which he defines texts simply as "organized sets of symbols or signs" (p. 137). Witte, and the others discussed in this section, illuminated how texts exceed their linguistic signs, since people take up a mediated process to engage the material nature of the text. This mediated process involves a person's reconciliation of their social-historical experiences with writing and the situated purposes of the text. In the remainder of this chapter, I discuss how this baseline definition of texts frames my study Ray's engagement with artifacts such as code and data sets, technologies such as databases and code libraries, and programming languages such as JavaScript as writing.

Witte's definition of texts helps me to explain more about this mediated process to use and produce texts. In one case, he describes how, for a person to recognize and use a mundane grocery list, they must have a history of experiencing, using, or creating such lists. In other words, meaning is not inherent in the list itself. Instead, meaning is constructed through a relationship between a person, recurrent objectives with the text, and the list's particular semiotics and arrangement that signal potential variance of meaning. An individual's social-material experiences, then, provide them a means to recognize the fact that it is a grocery list with certain imbued functions, features, and social dimensions—however tacit such meaning-making may be or become over time. Witte also notes how grocery lists inscribe material and social dimensions of everyday activities through the creative use of semiotics. Specifically, by semiotics, he considers how a person may write the grocery items in an order conducive to the spatial layout of the store for ease of use *in situ*. Furthermore, the items represent material items to be purchased and future actions to cook desired dishes, which also represent decisions made in relation to the ensemble of people related to the task of cooking for the week or maybe a special event. Such grocery lists may or may not chronicle such details, or even details about what future dishes are desired. Yet, perhaps, this list works in tandem with another kept-up list of weekly meals within the home for others to use and interpret based on their own situated desires and past experiences cooking (and eating). In sum, writing's material-semiotic dimensions shape how individuals develop socialized ways of writing and using texts in addition to simply getting things done. Overall, Witte's insights about texts demonstrate how writing integrates far more than the linguistic sign as somehow meaningful on its own.

How might Witte's exposition about the nature of how people use and make meaning with texts relate to questions about the materialities of coding? Writing research examines the consequences of different material configurations and semiotic dimension of texts: how changing the structure and content of the text with available will change how a person uses it and makes sense of it within their given context. More specific to my case-study of Ray's coding, Witte's insights about simple lists has implications for how coding can be understood as writing, since Ray's coding with and of data sets produces and uses complex, structured lists of "organized values or signs." Understanding code, data sets, and even databases as texts opens up an inquiry into how inscriptions of everyday material, social, and political life become represented for wide varieties of use in a digital medium.

My aim to understand coding as writing also includes conducting an inquiry that pinpoints the consequences of different material and digital technologies on Ray's coding practice to do this data work. It is not enough to simply claim that sets of data and code operate as texts in Ray's workplace, because writing itself is a slippery object of study to understand. In scholarly and commonplace understandings of writing, it is and can be perceived as both noun (object) and/or verb (activity) (Bracewell & Witte, 2003; Haas, 1996). Consequently, my claims about coding as writing requires evidence garnered from observed activity and not the texts alone. As a result of such a consequence, my research collects evidence that traces how Ray's coding activity mediates and is mediated by material and digital technologies, which provide him with a wide bandwidth of semiotic modes to express and interpret ideas when coding to support the reporting work in the newsroom.

Haas' (1996, 1999) research about writers and their technologies helps me to build on Witte's work to explore the consequences behind how humans develop myriad ways to make language material and now, as my dissertation shows, computational in a digital medium. She brought theories of mediation (*cf.* Vygotsky, 1934/2012; Bijker, 1995) to bear upon disciplines theorizing and teaching writing to cope with the implications grounding the claim that "[w]riting is language made material"(p. 3). My dissertation about coding as writing builds on her call to confront how even the most mundane, individual acts of coding are always, all-at-once mediated by mediational means (tools) and semiotic sign systems; that texts and tools provide writers durable and shared cultural experiences; and that, because writing is both material and symbolic, it binds habits of body, mind, and modes of production (*cf.* findings from Scribner & Cole, 1981, pp. 235-237).

For example, Haas, Takayoshi, Carr, Hudson, and Pollock (2011) studied young adults and their text-messaging practices, recognizing how texting is a distinct language form with its own normalized set of features and standards. They observed how people, their technologies, and the situational constraints of their text-messaging contexts provided them with new opportunities for them to write in – or inscribe – paralinguistic cues into English language use. These cues helped writers make meaning more precise, playful, and clearly marked with particular socially-adopted patterns. Haas (1996) places these dynamic dimensions of writing under the broader conceptual umbrella of the studies of writing's many materialities. My case-study explores how Ray takes up the JavaScript programming language within the context of his professional life, describing the forms, modalities, and other factors shape his coding.

Wickman (2010, 2015) also draws upon Witte to build substantive theories about the production and use of lab notebooks in a liquid-crystals physics lab. Specifically, Wickman's (2010) studied how scientists' lab activities worked in coordination with their production and use of lab notebooks to "'discipline'" (p. 265) their interpretations of material inscriptions created in the lab. Interestingly, Wickman's findings connect to my own study of coding as writing, since Ray's coding includes developing coding projects within dedicated folders to produce a corpus of diverse texts that 1) document Ray and his other coworkers' coding activity to process and analyze data, and 2) produce particular data visualizations from the output of those activities. This bounded nature of Ray's coding projects in such folders in addition to his epistemic work to make sense of code, large data sets, and other technical information pertinent to the reporting project provide a space for my dissertation to contribute to his theoretical work. Specifically, my study takes up his definition of the following terms that help me develop supporting language for my study: *text*, *inscriptions*, *semiotic modes*, *semiotic resources*.

Wickman (2010) also draws his definition of *texts* from Witte (1992), noting how it accounts for the diverse nature of texts that include both linguistic signs and non-linguistic symbols (Wickman, p. 288, fn. 2). Wickman takes up Latour's (1979/1986; 1988) definition of *inscriptions*, using it "to encompass representations that are produced through a scientific instrument or technique in the laboratory" (p. 288, fn. 4). In my case-study, Ray sometimes produces inscriptions through his coding work, taking values that represent social-material experiences and combining multiple data sets or values derived from diverse sources. Ray sometimes takes per instance inscribed values made available from some public-facing database interface and organizes them with other values and information to programmatically create data sets as texts. Other times, the inscriptions have already been assembled into data sets and Ray must write code that contextualizes the data to meet the needs of the reporting work. More exigencies and situations exist and will be reported in my findings, but I briefly mention these instances to paint the context of Ray's coding activity with some initial broad strokes in proximity to these definitional terms.

I use Wickman's definition of *semiotic modes*, which he describes as "culturally recognizable channels through which meaning is realized textually, for example, linguistic script, schematics, photographic images, illustrations, and so on" (p. 288, fn. 4). In Ray's case, "culturally recognizable channels" include particular computational modes made available through his use of the JavaScript/Node.js programming language and recurrent code libraries and digital formats. These are just some of the more technical semiotic modes that Ray puts to use when reading and writing of code to process, analyze, visualize, and archive data.

Wickman (2009, 2010, 2015) never quite defines *semiotic resources* explicitly. However, in all uses of the term, he suggest that people coordinate their use of a variety of technologies, artifacts, and inscriptions to produce writing. Semiotic resources differ from modes, then, in that multiple individuals can collectively experience the same semiotic resources, since they are tangible media. However, people in and through their interpretive work with such media draw upon and alter their sign systems, i.e., semiotic modes, which have the potential to become culturally-shared grooves to represent and share meaning.

Overall, these terms help refine calls to study the materialities of writing. Again, think of materiality as how people blend the material and conceptual during an activity. Studies of materiality try to understand how people make sense, question, or muddle up their understanding of the world and indeed the world itself through their socializing and material experiences. The materiality of writing, then, is to understand that texts – writing and reading them – involve far more than passively knowing the meaning of linguistic signs on a screen or page. Writing and its materialities exceed the text itself, because people create and use nonverbal, material-semiotic resources with written language. *I pick up this particular theory of writing to explain how code cannot be reduced to its linguistic sign, its technological referents, or its computational performance.* Coding is no more 'pristine' than writing, since computational dimensions of code must be understood as interconnected to other contextualized semiotic modes. Accordingly, my dissertation's aim is to provide a substantive theory about the materialities that support and alter Ray's workplace coding activity; namely, the *semiotic resources* that Ray employs through acts of writing and reading code.

The theory of writing that I develop throughout the remainder of my dissertation is how coding is not a pre-determined system of written signs to simply instruct computers to do things in the world. Instead, I position coding as a constructive act of representing the social-material world in a digital medium through a wide array of semiotic resources. I will provide evidence showing how the meaning of Ray's code is not static, but developed through a mediated process. Accordingly, I designed this case-study (Dyson & Genishi, 2005; Flyvbjerg, 2006; Yin, 2014) with a Writing Studies methodology (Bracewell & Witte, 2003; Farkas & Haas, 2012) to begin cultivating knowledge about the range of semiotic resources that Ray uses to contextualize data and computational operations through acts of reading and writing code. I place findings from my research in a new arrangement of ideas to shore up a richer sense about why coding, like writing, exceeds the linguistic sign. And, in code's sense, it exceeds the computer's performance too. The act to read and write, whether code or any other text, means a person has taken the time to develop particular cultural practices, and now find themselves responding to yet another call to read and write to put that practice in action. Consequently, coding is subject to similar laws of materiality as writing, because code is making language material. My dissertation explores the consequences of code made material in addition to how code also makes language computational in a digital medium.

In the remainder of this chapter, I survey literature from across fields that ally with similar problems linked to understandings about the written materialities of code. As a way to synthesize these approaches, I maintain a broader question to ask across sections: "What is code?" In so doing, I hope to illuminate the different approaches to understanding code, as well as highlight the research problem that my dissertation addresses; namely, my dissertation develops a substantive theory that begins to explain why writing code has also become a dynamic, expressive use of language: a form of writing that no singular discipline can understand in total, but requires a research agenda that addresses its multiplicities. Understanding coding as a writing means exploring how it shares the following general propositions that the above studies of writing and its materialities have garnered over the years. The following table also includes some of the major literature reviewed hereafter, as a way to introduce how such scholarship integrates with my own research agenda. My dissertation serves as a way to start making these general connections more explicit: for example, how writing code exceeds its linguistic signs, and how code and coding is something more than its end-product, its tools, or its symbol systems.

1. Writing represents and mediates reality.

2. Writing is an individual-social activity.

3. *Mediational means* shape writing activity.

4. Writing exceeds the linguistic sign and is multimodal: writers draw from a range of semiotic resources and organize them as texts.

5. Writing is epistemic. Meaning-making is an active process.

6. Writers imbue texts with values, appeals, and knowledge about the world, i.e. writers develop a sense of the social.

Such general propositions must always be coupled with more substantive, domain-specific properties to better craft explanatory theories of coding and writing, more generally. In what follows, I review ideas indicative of major fields, who have engaged the relationship between people and code in their own distinguishable ways. In software engineering (D. Knuth, 1968/1992; C. V. Lopes, 2014b), I survey circulating ideas with regards to code as writing endeavor: how code has human audiences, expressiveness in its style, and yet this discourse predominantly adheres to more "pristine" notions of how to render code plain. In Human-Computer Interaction (Buse & Weimer, 2010; A. J. Ko, LaToza, & Burnett, 2015), I survey similar issues as the former about making the texts the developers write more plain for its audiences. This field of research largely assumes the act to write code as being stable and generalizable to large-scale, software engineering domains of coding work. Consequently, any claims about what code is, exactly, assumes a particular set of contexts and objectives. In Rhetorical Genre Studies (Brock, 2016; Brock & Kelly, 2015), I review initial efforts by rhetoricians to analyze code in the similar vein as traditionally written texts. Such scholarship begins to generate claims surrounding the value-laden nature of code, and how people can express their ideas in and through code. Finally, in Literacy Studies (Barton, 1991; Scribner & Cole, 1981; Vee, 2017), I review theories of literacy and coding that refine the problem of understanding coding and its materialities; notably, how code not be reduced to its machinic performance or its linguistic sign as a plain rendering of reality. Instead code, akin to traditional writing, exceeds the machine and text. After this review of literature, I provide a structure of the dissertation.[4]

1.4 Code as more than computational techniques: Software engineers and coding as writing

Many professional programmers, whether they call themselves software engineers or developers, often share their thoughts about the relationships between coding and writing. Computer scientist and engineer Knuth's (1992) "literate programming" movement remains the most widely known. He called for other programmers to recognize how code is meant to be read by people, so writing code should be regarded as an artistic and human-minded act (p. ix). He attempted to cultivate a new "attitude"(p. 99) for programmers: instead of seeing their main task as writing computer instructions for a computer, he desired for more programmers to focus on how their code communicates ideas to an audience about what their instructing computers to do. One major way that he tried to implement this attitude was through his work developing the WEB programming system.

[4] I must note how I am aware that for over the last 30 years short bursts of scholarship have called for writing and rhetoric to code, to teach code, and/or to study code (Burns, 1979; Carter, 2016; Cummings, 2006; Haefner, 1999; Hart-Davidson, 2012; Leblanc, 1993; McGee & Ericsson, 2002; Sorapure, 2006). I acknowledge their propositions and efforts persuaded me to take on this project. However, I found that I could not build directly on their work, since such scholarship never provided their own method or traced out their methodology to help me design and carry out a study about the materialities of coding as writing.

Knuth's WEB program authoring system aimed at becoming what some in software development contexts refer to as language agnostic; that is, a coding technology and tools that was not overly-specific to one programming language over another.[5] In Knuth's case, he desired to help programmers arrange code in a way likened to an academic essay. Knuth argued that reordering code to mirror the structure of an academic essay provided programmers potentially more readable code. He claimed that this readability stems from a more natural order "that is best for human understanding" (p. 99).

This last point is an important one: how Knuth adheres to a natural arrangement of texts as an essay. It may be no surprise to researchers of technology, mediation, and writing that not everyone agrees with Knuth nor prefers the WEB concept or implementation (*cf.* Cordes & Brown, 1991). Recall how Knuth named this movement *literate programming*. He admits that he named it out of "a bit of malice" (p. 100), since he begrudgingly adopted a structured programming methodology during the 1970s. The details about structured programming are not important in order to emphasize the following point: how Knuth's cultural norms are intertwined with technological ways people read and write. This stands for code or any other text, as I will discuss in more detail in the next chapter.[6]

Knuth pulls no punches about his own desire for this essayist vision. He goes so far to tell his broader community that he means to "get even" (p. 100) and notes how "[b]y coining the phrase 'literate programming,' I am imposing a moral commitment on everyone who hears the term; surely nobody wants to admit writing an *illiterate* program" (emphasis original). In this way, Knuth pokes about the materiality of coding in relationship to traditionally print-based modes of writing. Yet, Knuth never fully

[5] Please note how much human coding work it takes for something to become considered under the notion of *language agnostic*. I do not have the appropriate space to discuss this matter here, but Knuth's coding environment would require an interpreter engine that painstakingly integrates various languages and their grammars. In short, agnostic simply means outsourcing human coding efforts to a more select few, who do the coding work to enable Knuth's writing environment.

[6] *cf.* Haas' (1996) exposition on the consequences of making spoken language material and Wickman's (2010) notion of "culturally recognizable channels" (p. 288, fn. 4)).

confronts how or why he naturalizes essay writing as *the* materiality of writing that he deems more readable and worth proposing to coding communities. In effect, Knuth developed *a way* of reading and writing code, rather than what he hoped would cultivate a unifying, language-agnostic one.

Knuth elaborates on a few qualities of his literate coding practice. He states that the developer-as-essayist is mainly “concerned with exposition and excellence of style” (p. 99). He elaborates on what he means be these 2 qualities:

> Such an author, with thesaurus in hand, chooses the names of variables carefully and explains what each variable means. [They] strive for a program that is comprehensible because its concepts have been introduced in an order that is best for human understanding, using a mixture of formal and informal methods that nicely reinforce each other. (p. 99)

Again, Knuth draws from the values, tools, and skills of an essayist. He imagines coding as a very specific type of language art, wherein communicating intent and concept through a prosaic style of writing code match the computing culture’s predominant focus on engineering and mathematics.[7] He means well to bring written communication to the forefront of minds, but collapses the human experience of writing to academic essays and consequently totalizes one particular materiality of writing and reading as best for anyone writing and reading code. He proposes very particular print-based literacy properties commonly found in his circles of academia, rather than integrate his WEB programming system with research that explains how and why people develop

[7] I think it important to note how Knuth’s fix on the clarity of code as literary and artistic expression could also be considered too thin. For instance, computer scientists and new media artists Mateas and Montfort (2005) make the case about how obfuscation, not Knuth’s nor others’ bent on clarity (p. 145), appeal to many others across programming communities. According to Mateas and Montfort, weird, obfuscatory languages and programming practices invite others to be fully captivated by esoteric code—akin to some forms of poetry. In keeping with this appeal, I learned more about such code aesthetics through an interview with new media artist Daniel Temkin (Lindgren & Temkin, 2015) about esoteric languages, their appeal, and Temkin’s own ongoing project, `esotric.codes`. In short, communities of programmers enjoy and choose to dwell in the unknown and esoteric states that languages often deliver, which constitutes another dimension of code and its literary expressiveness (*cf.* Mark Marino, 2014).

affinities for some materialities and aversions toward others.[8]

Knuth's essayist vision of programming has been picked up by many others across the developing history of programming. For instance, a long-standing movement exists to tranpose Strunk and White's (2000) rigid prescriptions of prose in *Elements of Style* to professional programming practices. Kernighan and Plauger (1978) developed their book, *Elements of Programming Style*, out of the same vein. White published Strunk's rules in a militaristic manner, as a campaign against the so-called paucities of contemporary (at the time) language use. In its preface, White characterizes the delivery book in this way, stating that "The reader will soon discover that these rules and principles are in the form of sharp commands, Sergeant Strunk snapping orders to his platoon" (p. viii). The book contains 11 rules of usage, 22 principles of composition, and 21 suggestions for a more clear, concise style. Kernighan and Plauger's *Elements* are of the same uniform. They take example code in procedural languages of the time (Fortran and PL/I) and develop lessons at the end of each section, some lessons of which one might confuse as Strunk and White's own, such as "Say what you mean, simply and directly" or "Write clearly – don't sacrifice clarity for efficiency." Other lessons echo Sergeant Strunk's tone, when snapping orders such as, "Avoid too many temporary variables," "Don't comment bad code – rewrite it," or "Use statement labels that mean something." These efforts to map particularly prescriptive purviews of writing onto coding persist today.[9]

[8] I make this claim not expecting Knuth or others in the computing sciences to have already developed a research agenda that would arrive at a writing-research purview. Their discipline is driven by research-and-design values and the myriad implications about what research looks like and produces under a science and engineering college. I frame it this way to begin carving out how my research can be distinguished from such movements within Computer Science and Engineering.

[9] Refer to well-known software developer, Atwood (2008), and his blog about programming: *Coding Horror*. In particular, Atwood wrote a post concerning some of these very matters: "Coding: It's just writing." In the comments of the post, programmers discuss their own satisfaction felt, when reading and adhering to Strunk and White's writing orders. Many others share their posts (Batchelder, 2002; Devlin, 2008; Koehls, 2008; Wheat, 2006), which follow the same or similar suit as Kernighan and Plauger, developing their own orders and principles as seemingly derived straight from this 20th century style manual.

These aforementioned perceptions and prescriptions of coding as writing is how they all adhere to a window-pane understanding of language and its materialities. They cradle their general rules and the enforcement of such rules within some grains of truth: maxims that they have constructed through their practice of software development. Yet, these principles belie the people, their histories, dynamic situations, and language use brought to bear upon their knowledge, skills, and tools coordinated during every coding task. Overall, a response to the "What is code?" question within this domain often yields ideas about making code plain by practicing tacit notions of simplicity and following particular guidelines as passed down across generations of programmers. Coding as writing, then, becomes aware of audiences beyond the computer, but submits to the notion that writing code across contexts and situations warrants particular prescriptive patterns born out of anecdotal experience.

If the question of understanding coding as writing were left here, their remains a kind of Schrödinger's effect to either ignore or pursue further. Just as Haas notes how writing does not dwell and develop in a place of its own – that writing is not "pristine" – people who write code understand too well how code can easily become muddled, while still performing its computational work. It seems that code cannot be placed in its own tidy space either, so the question remains: *What might be learned about coding, if writing researchers explore the consequences of making language material and computational in a digital medium*?

Another computer scientist and engineer, C. V. Lopes (2014b), helps me to answer this question with some more nuance. In her book about programming style, Lopes approaches the textual and stylistic nature of coding differently than Knuth and the aforementioned practitioners. Instead of code styles as prescriptive patterns, she conceives of style as born out of constraints. Rather than cultivate Knuth's 'literate' attitude and essayist vision for reading and writing code, Lopes wrote an Oulipan-inspired[10] book

[10] As Lopes herself reports, *Oulipo* refers to a group of largely writers and mathematicians who imposed particular constraints on the form of their literary works. Queneau's (1947/1981) widely-known *Exercises in Style*, for instance, wrote the same story plot in 99 different ways.

on *Exercises of Programming Style.*

Throughout the book, Lopes provides historical overviews about how many languages share conceptual lineages. C. Lopes (2014a) explains how she came up with this project on programming style, while teaching a graduate-level Computer Science course that focused on analyzing programming languages for their computational techniques. Her goals in the course and the book included illuminating the computational techniques embedded in a language and distributed across languages as a way to develop a historically-minded expressiveness of programming.

What Lopes refers to as techniques, I consider a gesture toward the materialities of coding as writing, where languages offer particular modalities for writers to use. Her book taps into the materialities of coding by her writing of a simple term-frequency algorithm 33 different ways, using 33 different sets of computational and technological constraints with the Python programming language. Throughout the book, she describes the code-styles associated conceptual and language histories, exigencies, and developments of such a computational technique. She emphasizes how these techniques are now embedded within numerous languages to express and use in a multitude of ways, and her book often contains exercises for people to write the same code style in a different language to convey this point.

In effect, Lopes explores and links historically recurrent semiotic resources and their potential contextual expressions. Through her writing exercises at the end of each chapter, she also implicitly suggests how such computational modes must be applied differently in different languages. For example, Lopes writes the term frequency algorithm in a style that she calls "Go Forth," as named after the Forth programming language. "This curious little language [Forth]," Lopes writes, "has at its heart the concept of a stack" (p. 18). (See Figure 1.2 for Lopes' visual representation.) She discusses how computer scientist Charles Moore implemented one of the first concepts of a stack data structure as a personal project derived from his own programming-related

Queneau's book served as a direct inspiration for Lopes' book.

problems, while working within the Smithsonian Astrophysical Laboratory in the 1950s (p. 20). Such a conceptual strategy to manipulate data remains integral to modern programming languages, which she shows through her example code.

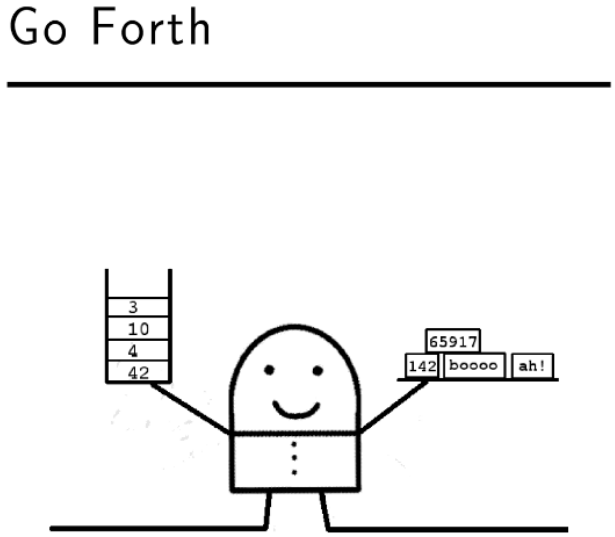


Figure 1.2: Lopes (2014) includes an animated representation of the style (p. 16) at the beginning of every chapter in similar fashion as Queneau (1947/1981).

An exposition into Lopes' code is not important for the broader aim of this chapter: To learn how others across professions and disciplines have linked coding and writing. What is important is how Lopes' book highlights a few of the external and historical elements that influence how a person approaches the task to write a particular computational task. Through an artistic approach to code styles, she begins to show how people who write code do so as contributions within a socializing history. Such insights about the deep histories of computational techniques within and across programming languages run parallel with theories of spoken and written utterances. Bakhtin (1986) might say that a coder is never the first to write or read code, but they always do so within the broader history of programming languages (p. 69). Witte (1992) would concur and note how the coder's code and context in which they write and read exists in a broader intertextual relationship; that is, varying types of code, "relate not only to one another but also to a culturally enacted stream of discourse that allows people to construct particular meanings through particular sorts of texts" (p. 253).

In this dissertation, I will illuminate the relationships between historically embedded computational modalities of programming languages and the other contextual factors that mediate Ray's production and use of code. I do so by analyzing how Ray takes up computational modalities *in situ* in combination with other situated objectives and tools. In this way, such embodied histories can be theorized in relationship to the diverse ecologies of coding practices, rather than as generic, decontextualized techniques. My dissertation draws a rich picture about how written language is linked to the computational modalities that Lopes begins to unite. More specifically, my research examines how Ray reads, writes, and uses multi-dimensional data sets in complex ways through his coding. I position my ethnographic case-study – informed by the writing theory discussed so far – as a means to produce a substantive theory that connects coding's computational constraints to other social dimensions brought about by other artifacts that mediate Ray's coding in a newsroom.

1.5 Code and coding as quantifiable units of analysis

Controlled experimentation of computer programming has been developing as various fields of study since the 1970s (Basili & Reiter, 1979; R. Brooks, 1977, 1980; Koenemann-Belliveau, 1991; D. Knuth, 1972; Soloway, 1986). Since then, quantitative studies have become increasingly prevalent across numerous disciplines in an effort to increase developer efficiency and productivity, as well as code quality. Researchers from the cognitive sciences, social sciences, and computing have been promoting and solidifying quasi-experimental designs (A. J. Ko & Chilana, 2011), converging under fields of study, such as Computer-Supported Cooperative Work and Social Computing (CSCWSC) and Human-Computer Interaction (HCI). In what follows, I review some of the research that addresses the “What is code?” question.

These domains identify pragmatic units of analyses of coding work to conduct quasi-experimental research. Studies typically generate findings about general features of programming languages or the effect of a new or existing tool or methodology. For example, some research may ruminate around a particular problem, such as “code clones” across a large codebase. Code clones are a consequence of the over re-use of a code pattern that can function across multiple situations within a program. These clones become an issue, when they become scattered about the codebase, which often correlates with increasingly error-prone code, since one code change may trigger errors across the entire program. Research around this named and prevalent issue in software development domains have tested tools that can identify across a codebase (Roy, Cordy, & Koschke, 2009); others have examined how such clones are produced (Kim, Bergman, Lau, & Notkin, 2004); others yet have studied how particular programming methodologies may specifically mitigate the production of such clones Nickell & Smith, 2003). Overall, such research represents the general orientation of this field toward problem-solution inquiries and evaluations for software development domains.

My claim about the overall problem-solution orientation of these studies can be backed up by a meta-study of over 1,700 software engineering articles (A. J. Ko et al., 2015). Some of the top researchers in CSCWSC and HCI discuss how their field indeed focuses on evaluating and creating faster software development tools and methodologies. These research agendas are excellent for measuring effects on general dependent variables, but the nature of some of the dependent variables that they study largely remain unexamined. For example, Buse and Weimer (2010) recently developed a metric to measure code *readability* in the same vein as the plain language models such as Flesch-Kincaid, Gunning-Fog Index, and the SMOG Index. They aim to locate which factors contribute to human notions of code’s readability, so they conducted a study with 120 university students enrolled in a variety of computing courses (p. 4). Participants were asked to annotate 100 short snippets of code in the Java language, resulting in 12,000 total readability judgments to analyze. Through a series of statistical tests, they took the average scores of the significant results of participant agreement, as a way to derive features for their modeling scheme. From this model of averages, they identified 19 common factors to develop their readability model, such as some of the following properties: line length, indentation, and the number of properties such as keywords, numbers, identifiers, operators, branches, assignments, loops, etc. Buse and Weimer note how their study does not attempt to "'simulate' the reading process," since they intentionally remove complexities of reading that may arise from a situation and context. However they neglect to review, define, or understand what such a reading process may or may not include, or how programming languages have increasingly become more abstract and expressive.[11] By omitting this part of the puzzle of reading code, they leave much to be accounted for about what and how a person reads code, which calls into question about how useful tools such as these become and what assumptions about languages to which they adhere.

Overall, they set out to produce a generalizable tool that adheres to the notion that decontextualized measures of languages features can make code more readable; hence, aligning with notions of rendering language plain. However, how is readability achieved, if reading itself is a constructive activity (*cf.* Haas & Flower, 1988; Haswell et al., 1999), or programming languages themselves incorporate myriad computational techniques to be used in particular ways? My dissertation may not address this particular issue of readability directly, but writing-related research suggests that reading processes are quite complex and the impact and design of any model and tool to address code clarity may need to take a step back and integrate the materialities of written language.[12]

[11] In section 1.7, I will review how literacy scholar Vee (2017) has discussed this matter of programming as an increasingly abstract writing system.

[12] Interestingly, both of my cohort colleagues and their research take up similar issues about readability, or plainness. Kira Dreher's dissertation research on the plain language application to the city charter challenges notions of plainness, finding that the expert users of the charter rely on networks of texts to interpret it. Some report how they outright ignore the new plain-language version, when it comes to their needs. Rachel Tofteland-Trampe asks usability experts to scale back their notions of user-interface naturalness by researching an under-represented group of

In a response to similar issues of the field, prominent computing researcher A. Ko (2016) describes the problems and challenges that computing educators and researchers face, when trying to measure how people learn how to code. Consequently, he pivoted his research agenda away from programming methodologies and tools to the programmer and their coding task, so he can help computing educators develop richer measures of assessment. He argues that one of the most important goals for any computing education field involves developing more fine-tuned methods to measure what a person knows and what they can do with that knowledge when performing a coding task. He claims that diverse coding tasks must be studied,

> ... helping us [researchers] observe ability across a range of skills, each event finely tuned to reveal the practice that lurks beneath someone's cognition. Only then will we have any clue how to support and develop skills in computing and know that we're doing it successfully.

CSCW, HCI, and Computer Science Education all elide efforts to understand what and how skills develop and are used by people *in situ*. However, can coding skills be assessed uniformly across domains of their practice? Are coding skills generic patterns to assemble in a particular way, or are skills more than how features of a language should be written for people and machines? Similar questions haunt writing teachers and administrators. My own textbook for a technical communication course isolates typified patterns of communicative moves that make up genres, generic patterns, of professional writing. Depending on how I were to teach with this textbook, this isolation of the local features of texts can be quite useful and important, but it can also misconstrue writing

novice users. According to her findings, naturalness and plainness of interfaces seem to be constructed over time through what Haas (1996) referred to as cultural myths of *transparency*. Transparency, I think myself and others mentioned here would all agree, is important and needed, since it helps people share their ideas and complete their objectives. However, researchers and professionals must continue to learn, test, reflect, and integrate our assumptions about how such transparency falls under perhaps cultural *familiarity* and practice, rather than some monolithic notion of natural representations and their uses. These issues build up in a well of research that will likely never run dry, nor should it.

activity as formulaic. Neither writing nor coding involve simple procedures to fashion local features and moves.

When answering the "What is code?" question, this group of research, whether in HCI or Computer Science Education, tends to elide context and its specific connection to reading and writing activities. Overall, context largely becomes assumed as larger-scale software development. Ray writes code on a news team dedicated to working with large sets of data, which only sometimes includes the more systematic tooling of an application, but never really engages the recurrent development of a particular software application or tool. Context illuminates the logical tension between producing generalizable, capital "C" Coding by only focusing on software engineering domains of practice. Consider another stark contrast of context from software development domains: What if Ko's call for clearer assessments of programming and learning are placed into the Visual Arts? How do these standards and foci change or not?[13]

In this dissertation, I argue that exploring questions about coding as a writing practice will help educators begin to understand how the coding that they research and teach encompasses a particular domain. My case of Ray's coding in the newsroom places his context, experiences, and situations at the forefront of my study. Such an inquiry will begin to provide substantive theories about how one should avoid conceptions of coding as "pristine." In sum, these fields of research have provided an important picture of the generic features of reading and writing code through fine-grained units of analysis surrounding problem-solution inquiries, each of which attempt to measure the effect of some variable. However, by taking coding and its contexts as assumed, coding as a writing phenomena remains conceptually pristine, i.e., unchallenged. There still remains a need for substantive inquiries into how people shore up multiple forms of semiotic resources to produce texts. My research begins to put these semiotic resources into the context of this very technical work, as an additional research approach to understand

[13] For example, consider the Array[ ] listserv (`http://arrayproject.com/content/discussion`), which is a community of visual artists, who sometimes engage in month-long topics surrounding the teaching of code within their courses.

how writing and reading code applies more than its general features; namely, just as there is no capital "W" Writing, there is no capital "C" Coding.

In the next two sections, I review research on coding that begins to see coding as social, material, and symbolic action. Specifically, I first review how rhetoricians (Brock, 2016; Brock & Kelly, 2015) perceive code as objects of study. Akin to natural language texts, they argue that code incorporates peoples' histories, philosophies, and values. From there, I review moves within literacy studies (Vee, 2017) to understand the links between writing systems in natural languages with that of computers and coding. Both approach the "What is code?" question differently, and I will note those differences, and how my research builds out from both lines in distinct ways.

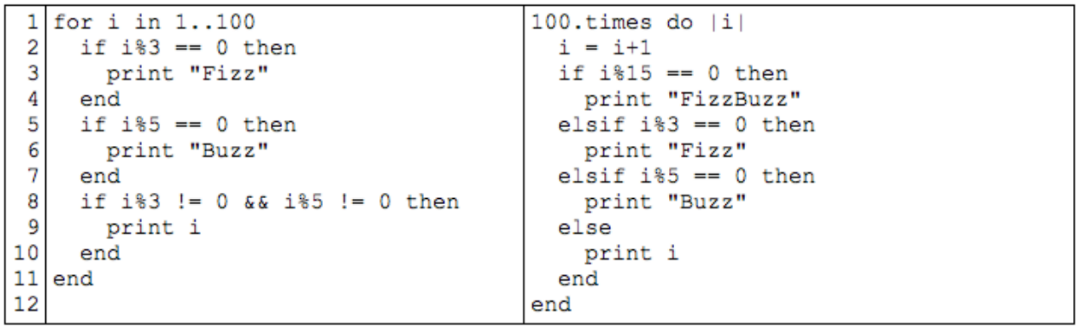
1.6 Rhetorical genres of code: Code as social action

Rhetoricians in the field of Rhetorical Genre Studies (RGS) offer writing and rhetoric scholars approaches to studying the social-rhetorical action embedded in and enacted through source code. Brock and Kelly (2015) draw from Carolyn Miller's (1984) work on rhetorical genre as social action to examine how computer code includes "situated and typified rhetorical actions and activities undertaken in response to recurrent situations or exigences" (p. 2). They argue that their method and methodology can "trace how genres in code produce and reproduce certain norms, values, and social actions through technologies" (p. 1) through the analysis of source code texts.

According to Brock and Kelly, a rhetorical analysis of source code texts can pinpoint the human values driving code-writing (in)decision during development cycles. In their study, they analyzed a corpus of open-source code within the Drupal Content Management System project. Specifically, they cross-examined code changes for a CAPTCHA, sign-in verification module between the Drupal 5.0 and 7.0 releases. They highlighted how the Drupal contributers' changes in their technical approaches represents a slow change in the community's programming philosophy: that is, the uptake of a new functional modularity approach to its overall design. They link these value changes to particular code changes. For example, they cite a major change in the code, when contributors abandoned their use of a more general `variable_get()` function, which called "data from a pre-existing server authentication" and switched "to a session-specific `fsockopen()` verification check from Drupal 5 to 6" (p. 2). Brock and Kelly argue that such major code changes represent changes in the social values of the broader Drupal community contributing to this module.

Brock (2016) builds on this preliminary scholarship by explicating a rhetorical style of code. In what Brock deems a "code genre" (para. 14), he analyzes a widely known coding task called the FizzBuzz test, which is often used within software development, job interview situations. The FizzBuzz test asks a job candidate to write code that prints the numbers 1-100, but with some strings attached. The code must print 'Fizz' for any multiple of 3, 'Buzz' for any multiple of 5, and 'FizzBuzz' for any number that is a multiple of both 3 and 5. Akin to Lopes discussed earlier, Brock surveys the varieties of ways people write this computational task as code within a specific programming language and technique. (See Fig. fig:brock-fizzbuzz-ruby for an example in Ruby programming language.) Brock argues that the expressive range of code offers people decisions about how they see their code in relationship with their professional audience(s) as it relates to the computational modes and conventions used to represent their FizzBuzz code.

This RGS branch of scholarship indeed establishes new approaches to understand how or why changes are made in developing a codebase over time. By building on a particular branch of genre studies, Brock and Kelly set out to show how code-as-texts include an individual's interpretive habits in relationship to social dimensions. Their approach begins to link human values and audience awareness to particular pieces of code within larger, everyday codebases, such as a Drupal module, or in situations, such as a job interview. They also open up a better sense about how code exceeds its computational performance in that it seems that certain values remain in contention



```
for i in 1..100
  if i%3 == 0 then
    print "Fizz"
  end
  if i%5 == 0 then
    print "Buzz"
  end
  if i%3 != 0 && i%5 != 0 then
    print i
  end
end
```

```
100.times do |i|
  i = i+1
  if i%15 == 0 then
    print "FizzBuzz"
  elsif i%3 == 0 then
    print "Fizz"
  elsif i%5 == 0 then
    print "Buzz"
  else
    print i
  end
end
```

Figure 1.3: Screen capture of Kevin Brock's (2016) FizzBuzz code example in the Ruby programming language (p. 7).

over time due to both technical and social considerations. More broadly, they set out to theorize the deliberative dimensions of coding practices.

My study of the written materialities of Ray's coding practice builds on their constructed sense about the social-rhetorical dimensions of code. However, my approach differs in a few ways. The main difference between RGS and materiality approaches to studying writing is their object of study. An RGS approach focuses on texts, their stable-for-now properties (Bazerman, 1988; Berkenkotter, 1995), and how such texts interact with other texts to cultivate a sense about how people develop typified "interpretive habits" (Spinuzzi & Zachry, 2000, p. 173). An RGS approach generates claims by focusing on the texts and their features, rather than the ensemble of people, tools, and knowledge that play integral roles in producing and using texts. As Witte (1992) has noted before, RGS assumes *a priori* rhetorical situations without investigating in finer detail about how such socially-constructed categories of situations become typified.[14] A study of coding and its materialities as a writing provides me with methodological tools

[14] Of course, this subject deserves and needs greater attention. While I believe that genre and materiality theories are compatible – perhaps complimentary in ways – their methodologies differ enough to warrant a devoted space to tease out what each theory of texts and writing brings to the disciplinary table (*cf.* how Witte (1992, 2005) seemed to already be thinking about these cross-talk issues.) For the purposes of this dissertation, I only wish to highlight some of my initial reasoning behind why I take up a materiality-approach to studying coding as writing.

to investigate the dynamics between Ray and his tools and texts across recurrent situations during my 5-month, ethnographic study. Akin to Wickman (2010, 2015), studies of materiality collect and triangulate different data about how writers integrate multiple kinds of semiotic resources to understand their situation and produce writing that fulfill the goals of the project. As a result, my study of Ray's coding takes up materialities of writing as an epistemological and theoretical endeavor over an RGS approach.

1.7 Code as literate social, material, and symbolic action action

> The machine is supplied with a 'tape' (the analogue of paper) running through it, and divided into sections (called 'squares') each capable of bearing a 'symbol'.
>
> –Alan Turing, 1936, p. 17

In the early 20th century, Mathematician Turing (1936/2013) imagined a machine – now known as the Turing Machine – that served as the germ for the existing computing paradigm that we all still experience today. As Turing begins to define in the epigraph above, the Turing Machine writes, reads, and re-writes symbols on an infinite loop of tape. This writing machine employs a header that moves either left or right along the tape, reading and (re)writing symbols in these squares. These acts of writing lists, per se, are contingent on instructions written on a card for the header to follow and execute. Turing's imaginary machine marks one of the most integral conceptualizations in computing, which inspired others to forge deeper material and conceptual connections between computing and writing. Turing conceived of a new method writing to take on complex human problems in new ways and with new 'materials': *data*. In hindsight, he conceived of a new communicative activity in as much as a computational one.

Of course, more historical moments, figures, and developments are needed to establish theoretical grounds for coding and its materialities as the latest form of writing.

Literacy scholar Vee (2017) developed a book-length, historical treatment of this particular subject. In Vee's book, *Coding Literacy*, she argues that both natural languages and programming languages enable people to represent the world as much as mediate what it might become (p. 114). She investigates the significant historical shifts and rhetorical treatments of coding as a mass literacy. For instance, she discusses how coding was originally the "dirty, hands-on work with computers" (p. 12), wherein academics made explicit moves to connect their developing computing disciplines with the sciences.

Vee also notes how the Computer Sciences have grown within a conflicted relationship with coding, since coding's status as a particular kind of engineering vocation has been changing rapidly since the 1950s. She shows how the dynamic developments of coding always seem to resist Computer Science attempts to contain coding within their disciplinary curricula (p. 12-13). By reviewing these comparisons and perceptions of coding and writing through a lens of literacy, she begins to illuminate how code, like other forms of writing, is indeed more than its computational performance, its procedures or functions, or its technologies. Vee argues that coding is far more than the common boundaries built around it in the prevalent discourse of Computer Science and engineering domains. She states, "While the relationship between code language and execution is slippery, code's status as simultaneously description and action means it is both text and machine, a product of writing as well as engineering" (p. 20). My dissertation builds on these initial moves to articulate the problem of reducing code to any particular domain, generalizable skill, or perception of code as synonymous with machine.

As I discussed throughout sections 1.1 and 1.3, writing cannot be bound to one place or singular unifying form and function. The study of the materialities of writing have illuminated how, as Vee notes, "Neither speech nor writing perfectly represents the other . . . Similarly, code allows for a form of representation different from writing" (p. 118). As a result, I do not mean for my study to claim that writing code is like writing with natural languages.[15]

Vee agrees with the other writing researchers in section 1.3 that studying the materialities of writing enables researchers to cultivate findings about how people play a role in how technologies and symbolic systems are taken up across contexts of activity. We also agree that both writing and coding are human acts to organize and enact a particular representation of reality. Akin to Wickman's study of scientists' production of lab notebooks, all forms of writing represent reality in particular ways, as well as mediate it. Writing constitutes the creation of representations of everyday activity as much as it is the actualization of it. Writing makes language material, and coding makes language material and computational in a digital medium. Both forms of written language spatialize ideas for thinking and doing. As Vee puts it, "Textual writing and code represent as well as construct the world." (p. 114), and do so in distinct ways.

In the same ways that other research on the materialities of writing (Haas, 1996; Witte, 1992; Wickman, 2010), Vee begins to pull together strands of questions and claims that Haas in particular used to provoke thoughts about how writing (with natural languages) exceeds the sign. For instance, Haas historicizes her research agenda – *What are the consequences of making language material?* – through multiple theoretical figures; notably the philosophical debate about the nature of writing that spans millenia between Plato and Jacques Derrida. The specifics of Derrida's arguments about writing are particularly relevant to Vee's and my own broader exposition of the consequences of making language digital and computational.

Both Haas and Vee discuss how Derrida (1981 and 1998, respectively) complicated theories of written language by challenging long-standing myths about writing as merely grammatizing thought in order for it to travel great distance through time and space.

[15] Vee's position about the over-reliance by others to compare the particular properties of traditional writing with that of coding could be compared with the naming of different materialities of writing and their early manifestations. For example, take Aristotle (2007) and his use of the word for *prose* in *On Rhetoric.* When prose is translated literally from Greek, it means *in bare words* (3.1.2), since there is no word for prose yet. At this particular moment in history, poetry was the dominant form of writing, so early Greek conceptualizations of prose bore out of an anti-thesis to poetry: a not-verse (*cf.* Graff, 2005).

Instead, Derrida argued that the act to organize language spatially and materially as writing translated information into an object with new modalities that both constrain and generate new empirical boundaries of communication.

For Derrida, the social and the material must be understood as also interlocked with the signs organized as texts. Writing may travel a greater distance, but it does not presuppose, as Derrida argued (1977/1988), that a text carries the whole of an author's signature, i.e., quantifiable code, since texts also produce what he calls *différance*. Texts cut across contexts and ordering of material things through those who read and experience them just as those who write them. In effect, Derrida was attempting to open up theories of meaningfulness that did not adhere to positivist or deterministic ideas of communication as a finite system of rules.[16] Derrida's *différance* theorized a non-quantifiable space and time as forever enmeshed with texts. This theory of relations between the social, material, and written communication poses numerous issues for any claims to what computer code is, how people write or read code, or what code accomplishes beyond the scope of any developer's original intent.

Haas used such theoretical insights to develop her Technology Question. Vee uses it to extend this theory of writing into coding practices. Drawing on Hayles' (2005) scholarship, Vee positions computer coding as exceeding writing in the same vein as writing does in relationship to speech:

> . . . Hayles takes [Derrida's] argument one step further by arguing that code exceeds rather than reduces writing. The way that code represents processes makes us look at the world in a new way . . . Hayles sees a complicated and reciprocal relationship across modes. Even if we aren't writing with the computer, and even if we cannot program, she observes that we are always working in language plus code: code has forever changed writing in the same way that writing has forever changed speech. The existence of multiple

[16] An important note for follow-up: What might be learned by comparing Derrida and Witte (1992), where Witte made a similar case about writing and texts, but through Charles Peirce and Vygotsky.

> modes changes our rhetorical choices and how we think about them. (p. 121–122)

Both myself and Vee adhere to literacy and writing research that understands how coding can never be reduced to machinic performance, a cognitive behavior, or how the situations in which people find themselves using texts as somehow independent from the histories of cultural practices of writing that inextricably shape writers and their writing. As Vee puts it, “Within scenes of writing, the objects, the people, and their complex relations shape literacy” (p. 34).

Literacy scholar David Barton (1991) made a similar call for literacy researchers. He discusses the move by literacy scholars to recognize the importance of such scenes as *domains* and how they shape literacy. Barton defines a *domain* as a socially-constructed institution, such as the home, school, or work, where different types of literacies are learned and practiced. He claimed that literacies “are nurtured by these institutions” (p. 5), and reciprocally these institutions are sustained through their literacy practices. Domains, according to Barton, are the contexts of writing practice that he wants researchers to fold into their unit of analysis. He explains that particular elements of any domain influence how people develop different meanings and forms of literacies as a result of such differences. For Barton, certain literacy practices stabilize over time and carry and shed values contingent on mediating factors (p. 9). Whether researchers choose to complicate traditional educational or professional contexts, or research new domains, both Barton and Vee invoke literacy studies as a means to theorize writing at work across different domains of practice.

In my dissertation, I invoke the foundational work by literacy researchers Scribner and Cole (1981) to develop a framework to understand the importance of context on any form of writing. Scribner and Cole, through their landmark study of the Vai culture, theorized the act of writing within a *practice framework*. Their concept of *practice* illuminated how understanding the nature of writing requires situating it within the particular, recurrent and goal-directed writing activities of the culture practicing it.

Such an investigative and conceptual move served as a response to prevailing paradigms of writing's relationship to cognition and culture, which attempted to extrapolate a thin theory of literacy that universalized writing.[17]

Scribner and Cole's *practice account* of literacy shows how writing is not speech without context and more than a technical, generalizable skill. Their research demonstrated how writing must be understood as dynamic, even within recurrent domains of practice, putting the onus on researchers to produce better methodologies and theories to understand how individuals integrate writing acts within their cultural contexts. Scribner and Cole's *practice account* helps me study how code is more than some collection of decontextualized computational techniques. Their research provides me with a methodological means to construct substantive theories about how Ray utilizes texts, tools, and artifacts to develop his habits of mind in relationship to the semiotic modes of code production. Accordingly, Scribner and Cole's theory of literacy informs the design of my case-study to illuminate the consequences of Ray's particular coding practice by cultivating a richer sense of the recurrent situations within his particular context. In the following chapter, I discuss how a practice account shapes the formation and design of my case-study. Before I do so, I summarize this review of literature and provide a breakdown of the structure of my dissertation.

1.8 A synopsis about the materialities of writing

In this chapter, I reviewed scholarship from across fields to trace resonant approaches and efforts to develop more nuanced knowledge about the "What is code?" question. I also positioned a lineage of writing theory – dedicated to examining everyday, situated writing practices – as a productive epistemology for understanding new dimensions of coding activity as a writing practice. The strength in a materialities approach lies in

[17] By thin theory and universalized writing, I emphasize how Scribner and Cole's research strongly challenged commonly held beliefs that learning how to read and write is directly connected to developing a higher intelligence and ability to think abstractly. This theory is more widely known as the Great Leap theory of literacy (*cf.* Goody & Watt, 1968; Havelock, 1976; Ong, 1958).

its ability to illuminate the substantive consequences connected to the human activity to produce and use language-as-material; that is, moving a step beyond understanding code's properties (code-as-noun) and integrating more studies of how code is produced (code-as-verb). Such a theory of writing's materialities can render new insights about the consequences of making language material *and* computational in a digital medium.

In this chapter, I also aimed at compiling some of the major propositions about the materialities of writing from foundational works in the still developing area of study. Recall how I provided a list of tabulated general propositions about the materialities of writing, which summarizes the theoretical dimensions of writing important for the conclusions of this dissertation about coding as writing. In an effort to explore coding as writing, I use these general propositions about the materialities of writing not as a way to conflate coding with writing with natural languages. (See Table 1.1 below for a tabulated list with the inclusion of important literature reviewed in this chapter.) Instead, numerous forms of writing seem to share these general properties. Accordingly, I examine Ray's coding activity as a writing practice that inscribes particular material and social dimensions of everyday life as digital texts created for his particular contexts of use.

Table 1.1: A list of the general propositions about the materialities of writing and their respective sources.

Propositions	Witte	Haas	Wickman	S&C	Vee*	Brock*	Lopes*
Writing represents and mediates reality	x	x	x	x	x	x	
Writing is an individual-social activity.	x	x	x	x	x	x	
Mediational means shape writing activity.	x	x	x	x			x
Writing exceeds the linguistic sign and is multimodal.	x	x	x	x	x	x	x
Writing is epistemic. Meaning-making is an active process.	x	x	x	x			

* Scholarship contributing to coding-as-writing.

S&C = Scribner and Cole (1981)

Table 1.1: A list of the general propositions about the materialities of writing and their respective sources.

Propositions	Witte	Haas	Wickman	S&C	Vee*	Brock*	Lopes*
Writers imbue texts with values, appeals, and knowledge about the world.	x	x	x	x	x	x	x

* Scholarship contributing to coding-as-writing.

S&C = Scribner and Cole (1981)

1.9 Dissertation Structure

In the chapters outlined below, I develop a substantive theory of coding as writing to understand how Ray's coding work supports the reporting work to develop data-driven news narratives.

I. Chapter 1 – *Toward Understanding the Materialities of Coding as Writing*: This chapter defines my research problem by explaining how my research build on previous studies of the materiality of writing. Traditionally, writing has been theorized as making language material. In this chapter, I draw upon previous studies of the materiality of writing, which explore the consequences of making language material, by also exploring the consequences of languages made computational in a digital medium. I position writing research as a means to open up a space to design and conduct an exploratory case-study of Ray's coding activity as writing.

II. Chapter 2 – *Research Method*, introduces the context of my research setting. From there, I elaborate on how studies of the materiality of writing produce a generative methodology to study coding. Then, I explain the design and conduct of my case-study (Dyson & Genishi, 2005; Yin, 2014) with a modified grounded theory method (Farkas & Haas, 2012; Glaser & Strauss, 1967/2009) to collect, analyze, and construct *chains of evidence* to develop my substantive theory of coding as writing.

III. Chapter 3 – *Findings: Aggregate Narratives*, presents findings from an analysis of the accompanying discourse and texts to Ray's coding mediated by large data sets. My findings suggest that coding on a team dedicated to telling compelling narratives about peoples' lives illuminates how Ray's coding – the production and use of myriad texts and tools – helps Ray develop insight about what data points can be linked to cultural and political narratives.

IV. Chapter 4 – *Findings: Coding Provisional Texts*, presents findings from a closer analysis of Ray's coding work to process and analyze data. Understanding his coding work as writing and structured data as texts opens up conceptualizations about how coding plays a role in Ray's process to contextualize data sets from their origin into the developing project goals.

V. Chapter 5 – *Conclusions, Implications, and the Future of Writing Studies of Coding*, discusses how my theory of coding as writing contributes to existing writing-related scholarship. I raise broader concerns connected to the consequences learned about how the structure and format of particular kinds of data mediated Ray's coding activity in diverse ways. For example, how database and data set design should be studied and understood as a rhetorical and communicative practice; especially in lieu of increasingly available data from larger political or corporate institutions and/or publicly available data across numerous digital platforms. Overall, I argue that examining coding as writing develops rich theories about how coding is another human incorporation of texts to organize, express, and perform diverse functions linked to inscriptions of everyday, social-material life.

Chapter 2

Method and Preliminary Findings

In this chapter, I survey the context of this case and the methodological decisions that I made to explore the following research question: *What can be about coding, if writing researchers explore the consequences of making language material and computational in a digital medium?* I adopted a single, exploratory case-study approach, since I desired to enrich theories about the materialities of writing with what Robert K. Yin (2014) refers to as *unusual* case: a case that instigates disciplinary pause to reflect on particular theoretical norms. From there, I discuss how I managed my grounded theory approach to recursively and iteratively collect and analyze data. Specifically, I discuss the import of this case context, the foundation of my methodology, the design of my method, and how this method enabled me to define units of analysis within the broader case of Ray's coding on a data-driven news team. I outline these moves below:

1. **Section 2.1 – The Case: Its Context and Selection**: I discuss my difficulties finding and selection this case context. From there, I review the qualities and benefits of a single, exploratory case of language use (Dyson & Genishi, 2005)

deemed as *unusual* (Yin, 2014).

2. **Section 2.2 – Methodological Approach to Studying Coding as a Writing**: Discusses how I bounded my study with questions that directed my data collection process during the initial exploratory phase of my observational period (Bracewell & Witte, 2003).

3. **Section 2.3 – A Grounded-Theory Approach to Data Collection and Analysis**: Reviews my modified grounded theory approach (Farkas & Haas, 2012; Glaser & Strauss, 1967/2009; Saldana, 2009) to the collection and analysis of data; discusses how it also led to the construction of 4 embedded units of analysis.

2.1 The Case: Its Context and Selection

This study is a qualitative case-study of the coding practices of Ray: a web developer, who works with editors and reporters on a data-driven news team. Ray telecommutes and worked from home during my observations. He stays in close contact with his team through stand-up morning meetings over Google Hangout video chat and the Slack messaging application throughout the work day. He is considered one of 2 technical people on this team, which also includes an editor, producer, and multiple reporters, where some reporters are officially designated as data-driven newsteam member, while others work within other parts of the broader newsroom. In Table 2.1, I provide a list of Ray's colleagues, who worked with Ray in some capacity during my observational period.[18])

Other team members read and wrote their own code from time to time or contributed to the code within the project in various ways. Ray often spoke about projects in the 2nd person, even if he was the only developer who had worked on the code-side of the project. When asked about why, he remarked that his experiences have shown him how

[18] All people listed have provided consent and names have been changed for confidentiality.

coding projects take many minds and input, so he prefers to acknowledge this pluralized work in such a way. However pluralized the coding work, my case focuses on Ray and his coding contributions on this team.

Table 2.1: Data-news team members and their roles.

Participant	Job Position	Designation
Ray	Web Developer	Data Team
Phil	Web Developer	Data Team
Vince	Editor	Data Team
Jun	Producer	Data Team
Rosa	Reporter	Broader newsroom
Avery	Assistant	Podcast

2.1.1 Coding on a data journalism team

Ray codes within a relatively new domain of journalism: *data journalism*. Reporters and editors have long used numerous forms of data and information to develop their narrative content. However, teams that are being known as conducting data-driven news integrate larger sets of data into their practice—data sets that have become easier to access, process, analyze, and even create—thanks in large part to coding and the Internet. Editors Brian Abelson, John Keefe, Sisi Wei, and Chris Wiggins (2015) recently spoke on a panel about the subject of data journalism. They all agreed that data teams help journalists ask and explore new questions about broader social trends and issues, but also verify and valorize what more traditional informants tell them. They also all concur that large data sets, and combinations of them thereof, inspire and cultivate motivation to consider what stories can be told from the data. Both Wei and Wiggins note how a data set comes with its own set of biases, so, as Wiggins puts it, “You have to interrogate its [the data] biases." Overall, coding has become a highly valued skill to put into action to work with and through data.

Ray's coding has become situated within broader editorial agendas developing through such data journalism teams[19] that use, produce, and contextualize large swathes of data. Journalist Brian Boyer, formerly of National Public Radio (NPR) and their data-journalism team, says, "One of the more exciting opportunities [of data journalism] is to create your own data. . . . We [journalists] can't just rely on the government to hand us data that we like. Often times, we have to go out and make it ourselves, which is hard . . . " (qtd. in Royal & Blasingame, 2014).

Ray often finds himself engaged in creating, processing, or using sets of data through coding. His coding mediates how he negotiates the meaning of data sets based on their original context(s) and the situated objectives developing within the newsroom. My findings indicate that Ray's coding activity to process, analyze, and curate data, as well as produce visualizations from it, involves richly complex rhetorical and epistemic work to contextualize these lists within the goals of the developing news narrative and/or broader editorial agendas of his data team. In short, his technical role on the team includes the production and use of complex information organized into data sets and structures that operate as texts.

Ray employs his coding knowledge and skills to support the narrative work of reporters and editors on the data team itself and across the entire WWWC organization.[20] During my observational period of about 5 months, Ray worked on 11 projects.[21] Ray's coding work typically supported other editors or reporters and their ongoing narrative work on a topic or issue. He read and wrote code to collect, process, and analyze large

[19] See Seth C. Lewis' (2014) edited special issue in *Digital Journalism* on "Journalism in an Era of Big Data: Cases, Concepts, and Critiques." I must note how my research does not draw on or contribute directly to such journalism scholarship. Instead, I cite this as a means to support the recurrent exigence for this type of data-driven journalism practices.

[20] WWWC is a fictitious name created for confidentiality purposes.

[21] This number captures merely what I observed, and even then there are other types of work Ray fulfilled that was difficult to inscribe. For example, conducting a quick code review for a team member, or helping them think through a particular problem.

data sets to then sometimes create interactive data visualizations. He also developed tools to support journalism work, such as an embeddable, interactive map tool and a data importer tool. Four of the 11 total projects were specifically developed by Ray's team, 3 projects supported reporters from the broader newsroom, and the remaining 4 projects supported other organizational services. Refer to Table 2.2 below for a more comprehensive breakdown of the projects, the project objectives, Ray's role on each project, and the number of days I observed said project.

Table 2.2: Data team projects, their outputs, and Ray's role.

Org./Proj.	**Objectives**	**Ray's Role**	**Days Observed***
Data-driven news			
City payroll	Team analyzed and visualized latest annual payroll data of city	Project lead: Write code to analyze, visualize, and create report	1
Recidivism archive	Process and import FOIA requested state data on recidivism into existing Amazon server	Data processor: Delegated to singularly conduct this importing work.	1
Train headway times	Team produces a set of values related to train headway times based on state DOT data feed.	Technical support: Ray tasked to fix a server issue with the broken feed.	1
Broader newsroom			

Continued on next page

Table 2.2 – continued from previous page

Org./Proj.	Objectives	Ray's Role	Days Observed*
State toxic sites	Help a reporter with their ongoing story about the state's toxic sites	Main developer: Write code to help retrieve, process, analyze, and visualize large data sets	6
Campaign finances	Reporter wanted database of recurrent data. Ray processed and archived state political campaign finance data	Data processor: Write code to archive data	4
City restoration project after natural disaster	Help a reporter with their ongoing story about the affects of a large natural disaster on a state-funded residential property building project	Reporter tabulated computed values; Ray asked to verify its validity and visualize data as a county-level map	3
Natural disaster affect on train times	Reporter inquires about affect of natural disaster on train times	Consultant: Over a phone call, Ray discusses data sets that can and can not help them develop their story	1
Other			

Continued on next page

Table 2.2 – continued from previous page

Org./Proj.	Objectives	Ray's Role	Days Observed*
Health texting application	Podcasting team requests help to create a mobile app for a texting campaign; App is participatory and collects survey-based data	Lead developer on the app's design and development; Also develops data sets for team	8
BINGO	Podcasting team needs to update code for browser-based game	Solo developer, who refactored / revised, the code	2
Mapping tool	Ray develops a mapping tool for reporters to create embeddable maps in their stories	Lead developer on this open-source project	2
Data importing tool	Ray develops an idea to create a data importing tool	Lead developer on this open-source project	2

* Some observation days included coding work on multiple projects.

Each project brought about different exigencies, sets of concerns, teams of people, responsibilities, and consequently an array of different types of coding. In Section 2.3 – Data collection and analysis, I will discuss how I used a grounded theory approach to explore the ways Ray's coding practices were meaningful and made meaningful in the context of the data team. Before I review my data collection, though, I will describe some of the factors that shaped the selection of this case. I hope that details about my process to meet and be invited to observe Ray provide future researchers – both in our doctoral program and interested writing researchers of coding – insight into my research design decisions.

2.1.2 Case selection and rationale

In all honesty, "selection" is a bait-and-switch on my part. I am not so sure *selection* justly captures the process involved in meeting and building a relationship with Ray. Before I met him, I went through a rough 6 months of ebbs and flows. I networked, engaged different communities through events like hackathons and meetups. I followed leads provided by fellow colleagues, friends, and other of whom I met along the way. One lead would fall through, but more would spring up. After some time, I found myself inundated with phone calls, emails, LinkedIn message threads, coffee shop meetings, more hackathons, more programmer community meet-ups, and so on. At one point, my case almost became a software development team, who were creating a particular kind of medical implant device. After that fell through, I almost landed within a team of software developers, who developed special networked applications for trucking companies and their trucker logs. Another yet led me to consider observing a team of university-employed developers, who worked with a particular set of departments. Despite these leads, almost all of them either fell through or left me wanting to explore more options, since I wanted to find a case to help me estrange coding from more traditional conceptions of it.

I describe these situations and tasks, since I think it is important to unravel the notion of what *selection* means and involves; particularly for a graduate student's first large research project. At first blush, a reader of my research may infer that *selection* suggests that I simply stood in front of a buffet of options and decided that Ray would serve as the best fit for my research situation. Instead, I desire to highlight how the arrival and constitution of this case was, has been, and is a reflexive negotiation between my research goals and my experiences with some of the broader Minneapolis coding community.

Numerous other factors shaped the negotiation and reflexive selection of this case. Yet, 2 factors may be of particular import to highlight here: code's status as licensable technology and fearful managers of developer productivity. These 2 factors seemed to encompass my difficulties finding a research site in which to study code as writing. Software domains are quite contentious, since the texts that developers write in professional domains are often protected technologies – even obfuscated. Consequently, many domains with patent-protected hardware and licensed code rejected my request to study their context. Mainly, they wished to abstain from sharing the technological dimensions of their code, which is completely reasonable, but a notable hurdle. Furthermore, I also received various reasons from managers that can be summed up by one particular manager's concern: how my research would "take developers from their chairs." In brief, code was perceived as a valuable commodity and developers' productivity was paramount.

In lieu of these factors and dimensions of studying coding as writing, I suggest that researchers use this knowledge to develop actionable items and clear protocols that state how the code and coding will be collected, stored, and rendered in scholarship. I became heightened to these issues along the way and developed a few different strategies and documents with the intent to express my sincere regard for any potential research participant needs and wishes.

I created 2 types of documents that helped me respond to perceived fears about my research and developer time. One such document, a "Request to Conduct Research" (see Appendix A.1), served me in 2 main ways. First, the creation of the document provided an exigence for me to articulate and amplify particular properties of my research protocols, which also helped me advance my own understanding of it. Secondly, I could use the content in correspondence and conversations, but also send the entire document to more interested parties. I also created a document about my research protocols, partitioning the content to make it more approachable (see Appendix A.2). I desired that this document would help me 1) ward of fear from supervisors nervous about my presence and time with their developers, and 2) eventually serve as a talking tool when I sit down with more invested parties.

Once I met Ray, sat down with him over lunch a few times, he obliged my request to become a part of his professional life. From there, we needed to persuade the legal team at WWWC that my study would define and follow an agreed set of provisions. To do so, he was advised by his supervisors to have me write a formal request letter on official letterhead (see Appendix A.3). My request was eventually approved, and my search for research site that I began at the beginning of the Fall semester finally concluded around the following mid-Spring semester.

Case rationale

As I note above, the selection of this case is a complex topic. The reasons that helped me decide that pursuing it further are just as complex. In what follows, I partitioned this section based on 3 main reasons that influenced the selection of this case of Ray's coding in a newsroom:

1. Collect diachronic and synchronic data of coding,
2. Learn from an *unusual case* of writing, and
3. Gain access to an often difficult domain of practice.

Firstly, the selection (and design) of this case investigates existing propositions about the materialities of writing by putting them into conversation with coding. Existing scholarship (Brock, 2016; Vee, 2017) about the written and rhetorical dimensions of coding use diachronic evidence – past events and artifacts – to develop their claims about coding. Scribner and Cole's (1981) *writing practices* practice framework adheres to the claim that the consequences of any person's reading and writing skills is more fully understood by identifying a person's uptake of particular writing technologies, knowledge, and sign systems within their situated activities (p. 237). My study contributes to this line of scholarship by collecting data that tethers both diachronic and synchronic qualities of Ray's practice (Gutiérrez & Stone, 2000; Witte, 2005).

Secondly, as I mentioned before, my case falls under the category of what Yin refers to as an *unusual* case (p. 51). Yin argues that a single case can be rationalized, if the case tests or confronts some of the tacit knowledge claims about a phenomena. In my discipline, this case diverges from the activities and writing-related phenomena typically studied, and consequently confronts the "theoretical norms" (p. 52) in Writing Studies. This unusual case of studying coding as writing provides writing researchers a rich description of a relatively novel form of writing – programming – and a newer instantiation of it in journalism domains. Findings about Ray's coding can in turn begin to be integrated with existing theories of writing and its materialities.

Another advantage of this *unusual* case is its potential to convey how the minutiae of Ray's everyday coding is full with substantive truths about coding. this case of Ray's coding practices on a data-driven news team will not yield a yardstick by which to measure the appropriate effect of data structures and formats on Ray's ability to make sense of and use data set(s) for analysis and visualization. Instead, I adhere to Dyson and Genishi (2005) and their call to use cases as a means to illuminate "what some phenomenon means as it is socially enacted within a particular case"(p. 10). I aim for my *unusual* case of Ray's coding to provide differently rich data and findings, which can be put into a dyadic relationship with previous theories about the materialities of writing. Such a move will help writing studies of coding push out from the dominant schemes of understanding coding under a pragmatist account: as an engineering or purely technical feat. As Vee (2017) notes, "...a focus on the computer *per se* is shortsighted and misleading as the processes of computing make their way into more and more technologies"(p. 22). Programming languages have become more expressive, as such languages continue to balance machinic performance and human expression of ideas.

Another layer of this case's *unusual* nature relates to Ray's coding domain: a newsroom. Ray has had much experience working in more traditional software development domains. In the newsroom, Ray experiences project objectives, artifacts, and timelines that do not necessarily adhere to traditional software domains. For instance, the editor, Vince, has been attempting to publish some type of data-driven news piece every 2 weeks, while Ray noted in an interview that some of the better projects take about 1 month to produce. To produce a different kind of coding project on such an interval serves as an unusual timeline for professional domains, despite the more recent trends for pushing code sooner on regular update intervals.

2.2 Methodological Approach to Studying Coding as a Writing

This case of Ray's coding presented me with some methodological challenges. No prior writing-theory driven research of situated coding exists, so this section describes the decisions I made to bring a such an approach to studying coding. As I established in Chapter 1, writing practices involve the use of a language that communicates and mediates everyday events, problems, and ideas. I also heed Vee's (2017) call for researchers to "take code on its own terms, rather than as a poor substitute for speech or writing"(p. 118). In other words, coding is a form of writing, but its differences must be taken into account. Consequently, my methodology draws from past studies of *in situ* writing, and I will use this section to discuss how I adapted past approaches to suit this case of Ray's coding.

In this section, I discuss my theoretical approach to understanding coding as a writing *practice* (Barton, 1994; Scribner & Cole, 1981). A practice account of writing guided the following methodological decisions that I made prior to collecting data: unit of analysis, data collection design, and analysis. Below, I survey the roots of this practice account by first reviewing how I contribute to a particular strand of writing theory that is grounded in a Vygotskian epistemology (Haas, 1996; Witte, 1992).

2.2.1 A Vygotskian epistemology: Writing and its mediational means

Principle to my understanding of the nature of writing are the theories of mediation developed by semiotician and psychologist Vygotsky (1934/2012, 1930/1987). His Marxist approach to studying the relationships between practical activity and human consciousness helps me to investigate the materiality of writing activity. Such an epistemology tests the claim that material artifacts used in goal-oriented ways mediate the skills and knowledge people develop, or what Vygotsky referred to as higher psychological functions (*cf.* Luria, 1962/1966). Vygotsky (1930/1987) argued that physical tools and psychological tools (sign systems) shape an individual's behavior in society. He theorized that mediation was a dynamic, dialectical process between cultural (higher) and natural (lower) psychological functions. In sum, a Vygotskian epistemology assumes that humans attempt to control and alter their material conditions using tools – both internal and external – in culturally-patterned ways. From this theory of meaning-making, an individual's practical activity is mediated by external and internal tools, which provides durable – yet, dynamic – sites of recurrent systems of action that potentially build sign systems shared among people. As a result, such dialectical processes make cultural sign systems and are made possible by cultural sign systems linked to an individual's history of media use across time and space.[22]

Scribner and Cole (1981) applied a Vygotskian epistemology in their landmark study of the Vai culture, where they theorized the act of writing within a *practice* framework. For them, a *practice account of literacy* served as a conceptual framework to better understand the complexity of any form of writing. Their findings provided evidence that

[22] For an extended example, refer to chapter 5 in Vygotsky (1934/2012) Vygotsky explains a range of experiments conducted in his lab, wherein him and his colleagues studied how children and adolescents develop concepts through activity. Also, note that Witte (1992) starts the heavier lifting to bridge writing studies and Vygotsky's work with *words* and signs more generally.

advanced knowledge about the materialities of writing as more than speech without context and more than merely a technical, generalizable skill. Their research inspired other writing researchers to develop better ways to understand how individuals integrate writing acts within their cultural contexts. Their practice framework helps researchers study how an individual's writing activity utilizes particular writing technologies, knowledge, and sign systems that have been accumulating over time to become culturally important or appropriate. Overall, my case-study adheres to their argument that researchers cannot more fully understand the consequences of a particular literacy (skills) without first identifying how people employ the aforementioned 3 properties of it *in situ* (p. 237).

Scribner and Cole's practice approach provides me with the basic framework for my case-study of Ray's coding practices. Reading and writing code shares these general practice properties with traditionally-held forms of writing: an individuals culturally-patterned use of tools, skills, and knowledge. As I argued in Chapter 1, the code and computations that developers write are no exception. Programming languages, code, and computations are not platonic ideals any more than the English language and the texts people produce and use.[23] The writing systems developers use are inherently cultural and linked to material tools. Consequently, all writing, including coding, is mediated by material and symbolic representations and operations, all of which coordinate and facilitate a developer's understanding, communication, and configuration of writing activities. In addition to the physical-material and symbolic, my dissertation takes another layer into account for coding: digital data. A *practice account* of Ray's coding would be under-developed without observing how Ray's creates, uses, and employs computational modes developed within the particular programming language of choice in this domain: JavaScript / Node.js. Accordingly, Scribner and Cole's practice

[23] Refer to Haas' 1996 survey of a range of scholarship about writing and its nature that span over 2 millennia. She uses this broad view of ideas about writing and its material form to frame her "Technology Question"philosophically (Plato, Derrida), empirically in diachronic investigations (Goody & Watt, 1968; Havelock, 1976; Ong, 1958), and empirically with regards to concerns about writing's affect on culture and cognition (Vygotsky, 1934/2012; Scribner & Cole, 1981).

account led me to learn more about how writing scholars apply such theories of writing and mediation to the research.

Haas (1996) positions such an epistemology to develop a central guiding question for literacy researchers—a question that she calls the *Technology Question.* According to Haas (1996), "writing is language made material"(p. 3) and requires some form of material technology to accomplish it, so any inquiry into the materiality of writing must include questions about technology. Witte (2005) also allies with Vygotsky's theory of mediation and uses the concept of *mediational means* to help researchers ground their inquiry into writing activity. He defines mediational means as the material object that is always-at-once a complex of signs and material object within a use context (p. 144). According to Witte, mediational means regulate the relationship between writer and the writer's goal(s) through its convergence of form, function, and substance (p. 148). *Mediational means* serve as a link between individual's and their historical-cultural uses of it. Such a relationship between past and present uses valorizes particular mediational means in their uptake and iterations, but also the individual's interpretive framework for its use. As a result, Witte positions *mediational means* as the focal point for researchers to craft their units of analysis (see also Haas, 1999). In my case, I used this methodological construct to identify the recurrent mediational means of Ray's coding practice, which helped me to begin narrowing the focus of my unit of analysis. (See Section 2.3.3 for a discussion about my process.)

To help with this process, I also relied on Bracewell and Witte (2003) and their 2 theoretical constructs: *tasks* and *ensembles.* According to Bracewell and Witte, *tasks* develop from participants' topics of discussion, and are typically "marked"as a particular goal, which is subsequently translated into a task (p. 541-42). Such problem and goal definitions that result in a task organize activities and the people who carry out such activities (p. 541). A task, then, "constitutes a unifying theme of the ensemble"(p. 546). In other words, I approached the design of my study, understanding how a writer's process to construct tasks help writers and me, as a researcher-observer, define the ensemble of people and how and when participants "constrain and valorize *mediational means*"(p. 546) to complete such a task. They define an *ensemble* as "the smallest group of people who collectively use sign systems in conjunction with other tools and technologies to realize an appropriate solution to a complex problem within a specific work context"(p. 528). They claim that more specific characteristics of an ensembles' actions and sign systems are identifiable through "the discourse of the participants and actions that attend the discourse (p. 547).[24]

I employ these definitions to study Ray's coding tasks and ensembles, since writing code includes tasks and ensembles, too. Ray is part of a work ensemble, which is subject to change. During his coding tasks, he uses mediational means to develop a mediating structure (tools, skills, and knowledge) that, over time, become his coding practice. In the next section, I outline how I put this epistemology to work with a grounded-theory approach to collecting and analyzing data.

2.3 A Grounded-Theory Approach to Data Collection and Analysis

In this study, I set out to develop a *practice account* of Ray's reading and writing of code that does not assume what his coding practice involves and is within his workplace context of the data team. In other words, I designed this case-study in order to construct a "thick description"(Geertz, 1973) of Ray's coding acts. Clifford Geertz's concept of "thick description"refers to a researcher's ability to differentiate complex, social meanings between very similar or mundane acts or events. He defines it through an extended example of blinking – a twitch, for instance, versus winking – detailing how the 2 share the same process, but differ greatly in a person's intent and another's response.

[24] Haas (1996) makes this case as well, asking researchers "to do more than merely observe as the discourse of our culture 'make'technology and as that technology, in turn, remakes discourse"(p. 23).

As a writing researcher, I understand how sign and symbol use are bound to the dynamics of the people using them; namely, how researchers must straddle the lines between the immediacy of the situation, its broader context, and historical perspectives by which a writing tactic and function is produced and interpreted. In order to more systematically attend to how Ray's particular acts of coding are and became meaningful, I employed ethnographic methods (Heath & Street, 2008; Dyson & Genishi, 2005) informed by a lineage of mediation methodologies (Bracewell & Witte, 2003; Haas, 1996; Scribner & Cole, 1981; Wickman, 2010) specific to the discipline of Writing Studies. Grounded theory become an integral part to enact this epistemological foundation.

In order to develop a principled approach to constructing a substantive theory of the materiality of Ray's coding, I used a modified approach to Glaser and Strauss's (1967/2009) grounded theory (GT). GT served as appropriate methodological approach to develop an account of Ray's coding practice, because it helped me examine and adjust to the nuances of meaning surrounding Ray's work within his domain. In the following sub-section, I provide an aggregate view of the different movements made with my GT approach to this case-study. Once I provide this broader overview, I will discuss some of the specific decisions about what and how I developed particular embedded units of analysis.

2.3.1 An aggregate view of my grounded-theory approach

For this case-study, I designed and implemented a recursive and iterative process of data collection and analysis over 5 months of ethnographic observations. These observations incorporated what Farkas and Haas (2012) refer to as a 2-movement grounded theory approach to data collection and analysis. (See Table 2.3 below for the breakdown of these research movements.)

According to Farkas and Haas, the first movement should be conceived as *expansive*, while the second movement as *contractive* (p. 89). My aims in the first movement of

Table 2.3: Modified version of Farkas and Haas’ table: “Research Movements, Phases, and Activities in Grounded Theory Approach”(p. 86, Table 4.1).

Phase	**Coding type**
Movement 1: Pushing out/undoing/fracturing *Recurrent activities: Constant comparison, memo writing*	
Phase 1	Open coding
Phase 2	Dimensionalizing
Movement 2: Pulling in/redoing/building theory *Recurrent activities: Mapping, memo writing*	
Phase 1	Selective coding
Phase 2	Integration

observing Ray’s coding activity involved gathering an understanding of the broader context of his work and history with coding. During the second movement, I designed and used finer-grained research protocols and analysis strategies, so I could begin integrating a theory of coding with writing.

Specifically, the first movement included analytic methods to configure and reconfigure data, or “‘fracture’”it (Glaser & Strauss, 1967/2009 qtd. in Farkas & Haas, 2012, p. 86), through the act of data collection and open coding. Open coding is the initial interpretive act of describing writing-related phenomena, when the goal is to “‘open up the inquiry’”(p. 87) by comparing and contrasting texts and other writing-related artifacts by their particular properties.

To aid in this fracturing process, I utilized the following methods within this first movement: Observations, Audio/Video of screen activity, observational interviews, semi-structured interviews, and artifact collection.[25] In Table 2.4, I provide a comprehensive breakdown of these methods, and I will discuss each method in more detail within the following subsections about each movement. During phase 2 in movement 1, I *dimensionalized* the data by employing different comparison points to recognize particular patterns: yes-no dyads, scales of magnitude, frequency of use, etc. Farkas and Haas note how such an analytic process yields preliminary results, identifying recurrent trends and outliers to test and observe as the next research movement is made (p. 88).

Table 2.4: Movement 1 Method – Respective aims, outputs, and analyses.

Method	**Aims**	**Output**	**Analysis**
Observations	Observe and document Ray's situated coding work; Gain insights about his coding process	32 coding sessions; 60-120 minutes per session; 250 pages of handwritten fieldnotes & memos: Log questions and un-recordable details	Supplemental data to cross-reference with audio/video
Audio and video of screen activity	Capture *in vivo* coding activity: Decisions, resources, and discussion	28 videos with audio for duration of coding session	Compare coding projects against each other and *in-beta* project against complete version
Observational interviews	Collect Ray's perception of particular coding moment or artifact; Verify my inferences about his code and coding decisions	Video/Audio of Ray responding to questions derived from coding/code	Compare Ray's perceptions against my inferences: both *in situ* and between sessions.

Continued on next page

[25] All of the digital data was stored on an encrypted laptop and backed up on an encrypted external hard drive. The hard drive, along with any material artifacts (fieldnotes or memos) were stored in a firesafe lockbox, as I outline in the IRB-approved proposal to the University of Minnesota's IRB.

Table 2.4 – continued from previous page

Method	Aims	Output	Analysis
Semi-structured interviews	Collect Ray's perception of historical coding experiences: both personal and professional	Three 90 minute interviews: 1 during first movement; 2 during second movement	Develop history & context of Ray's coding experiences; Compare against synchronic data
Artifact collection	Capture projects in current state for all 34 observational periods; Collect discussion items and decisions	Project code; Emails and Slack messages	Compare against situated data

Movement 2, *building theory*, involved 2 major phases: *selective coding* and *integration*. *Selective coding*, according to Farkas and Haas, includes the researcher's decision to examine a particular dimension in more detail. Selectively coding involves comparing the chosen preliminary core category to the data and against other categories. This initial phase in movement 2 provided the broader structure and sense to selectively code existing collected data and guide my decisions to use more fine-grained methods to collect data about more specific coding phenomena. In Table 2.5, I show the additional protocols, which I used more selectively with particular coding tasks.

I used these more fine-grained, data-collection techniques to understand Ray's *in vivo* reasoning during particular types of coding tasks. I used the think-aloud protocols (Schriver, 1991; Swarts, Flower, & Hayes, 1984) during the selective coding phase of my research, since I had more of a clearer sense about what types of coding tasks were of interest. I also used retrospective accounts (Greene & Higgins, 1994) of selected video-recorded tasks (Mackiewicz & Thompson, 2014; Thompson, 2009). Specifically, I used the retrospective accounts to collect data about moments from a prior project from movement 1. Both protocols asked Ray to talk through the task, and provided me with

Table 2.5: Movement 2 Method – Integrated the 3 following protocols amongst the Movement 1 method protocols.

Method	Aims	Output	Analysis
Think-aloud protocols	Collect Ray's perception of his thinking during particular coding task(s)	5 audio/video recorded TAPs of screen activity; 8-20 minutes	Compare this granular, situated coding task data against other observational data
Retrospective accounts	Collect Ray's perception of 1-3 minute moment of a coding task	2 audio/video recordigs of Ray recounting his coding decisions by watching previously recorded screen activity	Compare Ray's perceptions against my inferences about a particular moment of interest

data to compare against my developing core categories. Finally, I used observational interviews, as I had already done so within the first movement, which enabled me to ask Ray a series of questions regarding particular moments, objectives, or tasks. Overall, these methods provided a means for me to theoretically sample the ongoing investigation into Ray's coding work. I will discuss my application of each of these methods in Section 2.3.3.

From there, I completed my observational period and carried out what Farkas and Haas refer to as *integration* work. During this *integration* phase, I took developing preliminary findings from the first movement and began to compare it against the theoretically-sampled data from the second movement. In the following 2 sub-sections, I provide a more detailed account about how I arrived at my unit of analysis.

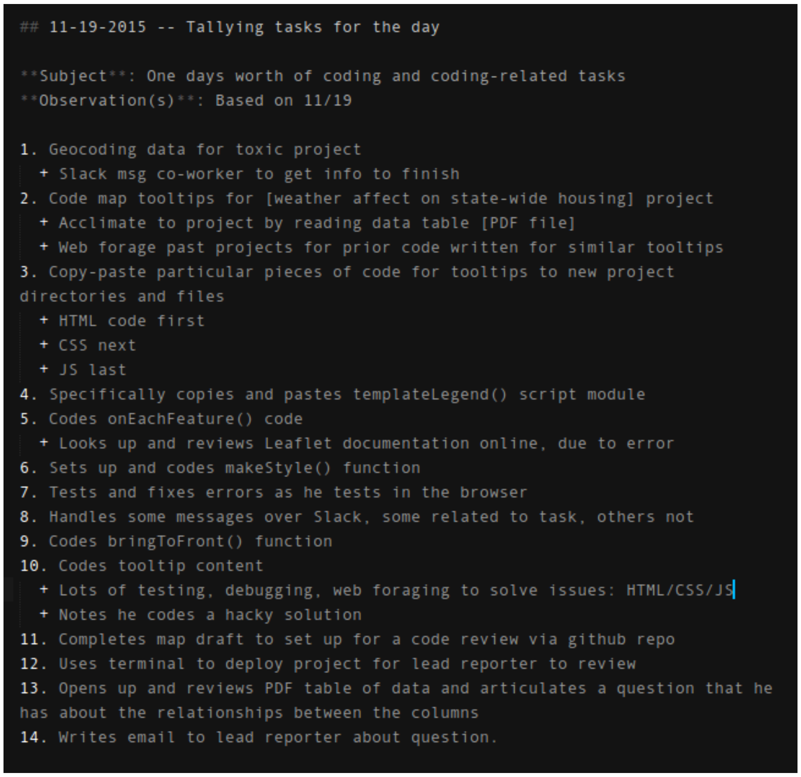
2.3.2 Movement 1: Opening up the case

In this section, I discuss some of the major methodological decisions that I made during the initial data collection and analytic movement of this case study. Due to the exploratory nature of this case-study, I report decisions related to developing a rich description about the different kinds of coding work Ray conducts within this context of the data team. In the next section about *movement 2*, I report how these decisions accrue into the formation of 4 embedded units of analysis (Yin, 2014, p. 55), as constructed according to my tasks and ensembles framework and GT's theoretical sampling technique (Strauss & Corbin, 1998, p. 202).

During the first 2 months, I observed Ray in his typical work setting within his home. As I note in the prior section (see Table 2.4), I opened up the inquiry in movement 1 with a fieldnotes, video data of Ray's screen activity, observational interviews, and one semi-structured interview, which I conducted later in the first movement. My goals during this initial phase were to open up the inquiry through ethnographic method of collection, transcription, and open coding of data. In so doing, I used open coding techniques to develop a robust description of Ray's context: what kinds of projects, tasks, and ensemble of people and tools make up the material of his work-life.[26] For example, Fig. 2.1 shows an example open-coding memos derived from the fieldnotes during my first week. It lists out the major tasks and objectives that Ray had conducted during my visit in the order of their completion.

Early memos such as these served as templates for my transcription technique, which enabled me to blend transcription with open coding. In Figure 2.2, I provide an example screen capture of my transcription method. I modified a web browser add-on tool for viewing Markdown files, so I could use the different heading levels to demarcate particular coding tasks (as seen on the right-hand side of the screen). Through this method, I was able to isolate particular moments during the video recordings by transcribing

[26] Again, this methodological means follows my adherence to a *practice account* of writing (Scribner & Cole, 1981) and other ethnographic studies of writing (Heath & Street, 2008).



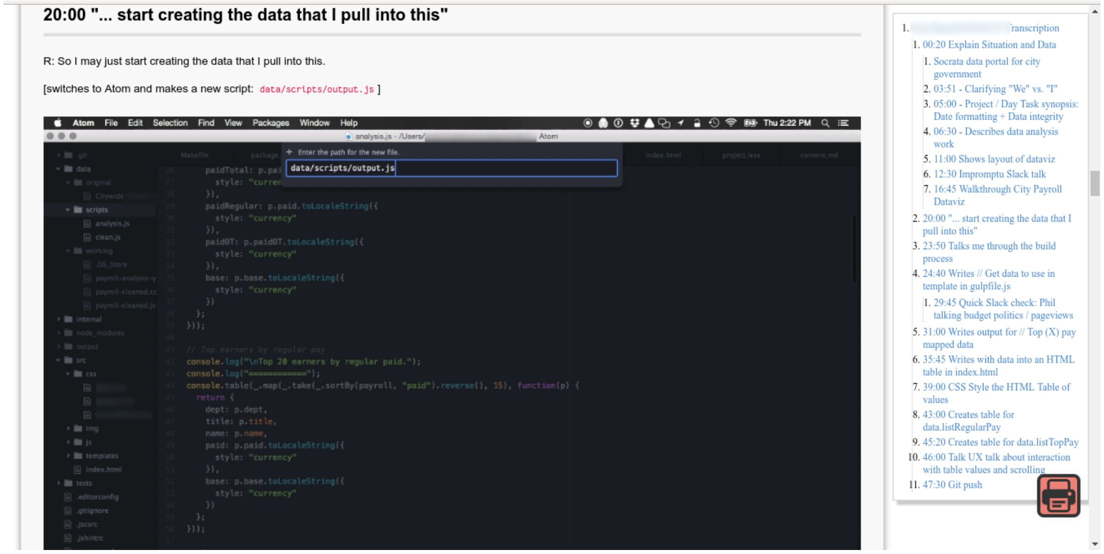
```
## 11-19-2015 -- Tallying tasks for the day

**Subject**: One days worth of coding and coding-related tasks
**Observation(s)**: Based on 11/19

1. Geocoding data for toxic project
  + Slack msg co-worker to get info to finish
2. Code map tooltips for [weather affect on state-wide housing] project
  + Acclimate to project by reading data table [PDF file]
  + Web forage past projects for prior code written for similar tooltips
3. Copy-paste particular pieces of code for tooltips to new project
directories and files
  + HTML code first
  + CSS next
  + JS last
4. Specifically copies and pastes templateLegend() script module
5. Codes onEachFeature() code
  + Looks up and reviews Leaflet documentation online, due to error
6. Sets up and codes makeStyle() function
7. Tests and fixes errors as he tests in the browser
8. Handles some messages over Slack, some related to task, others not
9. Codes bringToFront() function
10. Codes tooltip content
  + Lots of testing, debugging, web foraging to solve issues: HTML/CSS/JS
  + Notes he codes a hacky solution
11. Completes map draft to set up for a code review via github repo
12. Uses terminal to deploy project for lead reporter to review
13. Opens up and reviews PDF table of data and articulates a question that he
has about the relationships between the columns
14. Writes email to lead reporter about question.
```

Figure 2.1: Screen capture of early open coding, descriptive memo, which details what types of tasks Ray conducted throughout the observational period.

actions and dialogue, incorporating images of the screen, and including timestamps of each particular task from start to finish. This helped me open the code the data, but then set me up for a quicker coding pass later. Additionally, this transcription/open-coding technique helped me prepare for observational sessions. For instance, if I had a few questions after open-coding a 1-2 minute moment, I could demarcate a representative image or short video clip with the heading-level feature and use that object for a observational interview with Ray during my next observation.[27]

During the first movement, I used observational interviews (Katz, 2002) to help me develop a richer understanding of Ray's coding domain and verify *in-beta* interpretations

[27] I also developed this custom transcription method out of necessity. I use the Ubuntu operating system, so no standard transcription and coding software exists that support this particular system.



20:00 "... start creating the data that I pull into this"

R: So I may just start creating the data that I pull into this.

[switches to Atom and makes a new script: `data/scripts/output.js`]

```
data/scripts/output.js
```

```
    paidTotal: p.pai
      style: "curren
    }),
    paidRegular: p.paid.toLocaleString({
      style: "currency"
    }),
    paidOT: p.paidOT.toLocaleString({
      style: "currency"
    }),
    base: p.base.toLocaleString({
      style: "currency"
    })
  };
}));

// Top earners by regular pay
console.log("\nTop 20 earners by regular paid.");
console.log("============");
console.table(_.map(_.take(_.sortBy(payroll, "paid").reverse(), 15), function(p) {
  return {
    dept: p.dept,
    title: p.title,
    name: p.name,
    paid: p.paid.toLocaleString({
      style: "currency"
    }),
    base: p.base.toLocaleString({
      style: "currency"
    })
  };
}));
```

1. [redacted]ranscription
   1. 00:20 Explain Situation and Data
      1. Socrata data portal for city government
      2. 03:51 - Clarifying "We" vs. "I"
      3. 05:00 - Project / Day Task synopsis: Date formatting + Data integrity
      4. 06:30 - Describes data analysis work
      5. 11:00 Shows layout of dataviz
      6. 12:30 Impromptu Slack talk
      7. 16:45 Walkthrough City Payroll Dataviz
   2. 20:00 "... start creating the data that I pull into this"
   3. 23:50 Talks me through the build process
   4. 24:40 Writes // Get data to use in template in gulpfile.js
      1. 29:45 Quick Slack check: Phil talking budget politics / pageviews
   5. 31:00 Writes output for // Top (X) pay mapped data
   6. 35:45 Writes with data into an HTML table in index.html
   7. 39:00 CSS Style the HTML Table of values
   8. 43:00 Creates table for data.listRegularPay
   9. 45:20 Creates table for data.listTopPay
   10. 46:00 Talk UX talk about interaction with table values and scrolling
   11. 47:30 Git push

Figure 2.2: Screen capture of how I blended transcription work with open-coding work.

of the data. During the first phase of this movement, I adhered mainly to *descriptive question*, interviewing techniques (Spradley, 1979), as a way to solicit Ray's direct language use within this context, rather than implicitly or explicitly request his translation of the coding work being done. Spradley (1979) refers to direct language use within any context of study as a domain's set of symbolic categories (p. 100). He suggests that ethnographers carefully listen to and inscribe how informants use language by identifying *cover terms*, *included terms*, and *semantic relationships*. Cover terms typically serve as a broader category for other included terms, while the semantic relationship serves as the link between such terms. For example, I repeatedly heard Ray use the word *aggregate* in various ways in relationship to *data sets*, so I asked him descriptive questions about the 2 terms. I report more about the relationship between the 2 terms in Chapter 3, but I learned over time that aggregate typically served as a cover term to describe a data set's perspective of some phenomena of interest. Additionally, I observed how aggregate communicated a particular understanding of a data set (noun), but also was used as a verb to describe coding to produce a new data set. In other words, the cover term, aggregate, and its semantics seemed to shift, as contingent on its situated use. Spradley's interviewing strategies helped me listen to how Ray described and talked about his coding and coding-related phenomena with his co-workers and myself. Accordingly, this particular descriptive interviewing strategy helped me develop a richer account of the context and domain of Ray's coding practices.

The open-coding of these observational interviews and observational data helped me to develop the set of open codes (see Table A.1 in Appendix A.5). Due to the amount and variance of projects, exigencies, tasks, tools, and situations generated a large amount of codes across the existing data during the first 10 sessions. I generated these open codes from observing 3 projects of significance: 2 of which included data processing, analysis, and visualization work, while the other involved the tooling of a mapping tool for reporters to build interactive and embeddable maps.

The open coding of computer coding work is somewhat unprecedented, so making choices about the unit of coding, even if open coding, remains largely undefined. Based on my own search of other grounded-theory approaches to studying coding activity, Salinger, Plonka, and Prechelt (2008) remains the one of the lone applications, if not simply the most prominent. Like myself, they applied it as a way to explore their particular domain of study, but they modified their grounded-theory techniques to conform to pre-existing software industry terminologies, since they found it incredibly difficult and were "drowning in detail" (p. 14). As a way to begin developing a thick description, I started open coding at a broader level. Rather than focus on the content of the code or other textual artifacts, I followed (Bracewell & Witte, 2003) and coded at the level of tasks and ensembles of Ray's coding work. Accordingly, the open codes include Ray's language to describe particular coding projects and its accompanying people, tools, and other mediational means.

The open coding yielded a wide array of descriptive properties of Ray's coding work and context. After generating these open codes, I began to plan the first of 3 semi-structured interviews with Ray. In preparation for this interview, I reviewed my descriptive memos throughout the first month of observations and the connections that I was beginning to make across the data. In so doing, I begin phase 2, *dimensionalizing*, in order to prepare a set of questions and topics for the interview. As I reviewed the data, one memo specifically helped me start dimensionalizing Ray's coding against particular tools, knowledge, and skills. (See Fig. A.1 in Appendix A.4.)

In this memo, I questioned how Ray seemed to understand what kinds of data sets could be combined to test and verify his and his team members' sense about potential avenues for the developing news narrative. After these initial passes through the data, Ray's coding tasks seemed to be organized by the project's broader objectives (data processing and analysis, and web application work). Additionally, the objectives seemed to be shaped by the dynamics of the project's team and tools. Consequently, I wanted to learn more about these reasons for coding and how Ray and his team seemed to have a sense about what to do with the data and how coding was central to moving the project along. As the next step, I developed the following broader dimensions that would become the basis for topics in my initial semi-structured interview: *data work*, *project dynamics*, and *web application work*. (See Table 2.6 below.)

Table 2.6: List of initial dimensionalized codes; created in preparation for first semi-structured interview.

Dimension	Codes
Data work	processing data / munging data / combining data / reading data / writing data / analyzing data / visualizing data / archiving / toxic sites / train times / natural disasters / state government building project / assessments / testing / checking / verifying / tallies / totals / averages / correlation / significance / consistency / regularity / integrity / story / reporting / aggregate / 'create aggregate' / 'aggregate view' / patterns / data bias / incomplete data / 'raw' data / original data / data source / our data / data points / ID / data structures / reading in / file paths / input/output / final output / parsing / file formats / CSV / JSON / geoJSON / topojson / TSV / shape files / PDF tables / spreadsheets / OpenOffice Calc / Mac Numbers / text files / HTML tables / data miner / scraping / new data / old data / mapping / projection / reprojection / state projections / coordinates / neighborhoods / boroughs / municipalities / parcels / lots / districts / census tracts / demographics / race / poverty / income / CartoDB / FactFinder / Census Reporter / URL endpoints / Google Maps / GTFS (Google/General Transit Feed Standard) / regex / rendering / tiles / addresses / standardizing / web cache

Continued on next page

Table 2.6 – continued from previous page

Dimension	Codes
Project dynamics	projects / project-based / browser / inspector tools / project architecture / folders / files / organization / junk / README / documentation / identifier / Google search / Github / git / repositories / copy-pasting / translating code / dependencies / package.json / Makefile / shell script / Terminal / console logs / geocoding / API / rate limits / expenses / code modules / code libraries / proj4 / lodash / CRON / server / Ractive.js / Node.js / D3.js / Javascript / CSS / HTML / templating / templates / project exigencies / data + goal / data + storyfinding / data + goal and storyfinding / Ray + data + storyfinding / data consultation / Slack / email / Google Hangout / standup meetings / Slack messaging / Slack channels / direct channels / team channels / project channels / data questions / describe the data / code questions / project questions / context
Web app work	tooling / linting / gulp / code styles / open source / UX / whiteboarding / mockups / task delegation / end-user / end-user programming / option trees / place to code / canvas / markers / aspect ratio / map styles / Mapbox / cartography / search locations / labels / tooltips / prior coded feature / zoom / builds / generate / scripting / functions / conditionals / if-then / drag-drop / menus / icons / collapse / responsive / mobile / clone / parameters / arguments / this / return / stackoverflow.com / comments

Overall, this dimensionalizing served as the first step to develop a thick description of the context of Ray's coding work. As Spradley notes, it is important to not only find out what informants know, but how their descriptive language is socially organized within the setting. In Figure 2.3 below, I provide the question and topics, which I developed from these dimensions to continue the descriptive work of the first movement of my observational period:

Figure 2.3: List of questions and topics to discuss during the first semi-structured interview with Ray.

- Tell me more about your position at WWWC and the coding work that you do here?
- What kinds of projects have you worked on here, and how do you go about accomplishing them?
- What kinds of knowledge do you bring to coding here? I've noticed your background and use of mapping in particular. Can you tell me more about your experiences with coding and maps?
- Can you tell me more about your past professional coding experiences?
- Can you tell me more about your team members: Their roles and responsibilities?
- I've noticed that much of your work and discussions with your co-workers revolve around data. Can you tell me more about you understand data in relationship to your coding?

This interviewing structure adheres to Spradley's (1979) array of *grand-tour* interviewing techniques (pp. 86-88). I used these types of questions, since I was still new to Ray's work context and his position on the data team. These grand-tour questions asked Ray to describe his overall sense of the typical, recurrent processes, people, and artifacts of his work on the data team at WWWC, or as Spradley puts it, questions that invited Ray to describe "significant features of the cultural scene" (p. 87). With such an aim for descriptive language, I explored particular aspects of his coding context. I used follow-up strategies, such as *restatements* (p. 81) and *mini-tour* questions (p.

88), to dig deeper into smaller units of Ray's experiences across each of the respective dimensions noted above in Table 2.6.

After this interview, Ray and I took a roughly 2 week break over a holiday season. This provided an opportunity to code the interview data to subsequently compare against the other codes and data developed during the open and dimensionalizing phases of movement 1. This coding pass helped me begin to plan the second movement of the research. Specifically, the interview with Ray yielded a richer description about the different ways data mediates his coding work. I report more details about some of the methodological decisions and the development of my findings in Chapters 3 and 4.

I will discuss this particular set of findings in the following 2 analysis chapters, but in what follows I provide some of my initial memos and ideas to show how I prepared for the second movement of my data collection and concurrent analysis. My aim in doing so helps establish a grounded-theory approach of 'showing my work'; that is, by what means did I decide to *selectively* code and collect data about particular coding activities and tasks.

2.3.3 Movement 2: Reducing the data and carving out units of analysis

The analysis of the interview data in relationship to the codes generated a particular dimensionalizing of Ray's coding work; notably, his work with data sets in this domain. During the first movement of the observational period, data sets seemed to be an integral *mediational means* for Ray's coding work. Much of his coding manipulated data sets, so he could process, analyze, or visualize data. His context is defined by the notion of data sets. Namely, he works on a data-driven news team, and he estimated in the interview how roughly half of his coding work revolved around working with data sets. Nine of the 11 days of observation included this type of data work. As a result of this frequency in data work, I decided to selectively code and theoretically sample (Strauss & Corbin, 1998) particular kinds of coding related to data work during the second movement of data collection and analysis.

During this selective process, I also organized and reduced the data into more meaningful units of analysis. According to Smagorinsky (2008), researchers, who take a social-science approach, must make their larger set of data "...and reduce it to something comprehensible and useful" (p. 397). Smagorinsky notes how such a description helps other researchers avoid impressionistic data and understand what data was used and why. In this section, I describe the major decisions that I made to take my observational data between movements 1 and 2 during this exploratory case and reduce it into manageable units of analysis. Such a description provides others the means to know the whole of my data in relationship to the reduced portion reported in this dissertation.

This data reduction included a 2 major decisions methodologically. First, while I had already started to focus more acutely on Ray's data work, as evinced contextually by the number of days dedicated to data work and individually by my topics for the first interview, I made a more explicit decision to craft a unit of analysis surrounding this coding activity with data sets. Second, this decision guided my choices to apply the more fine-grained methods to collect data during particular *in situ* coding tasks: think-aloud protocols (TAPs) and retrospective accounts. (See Table 2.5 for a description of each method and their purpose(s).)

In Table 2.7 below, I provide a list of these methods, the number of times that I applied them during movement 2, as well as which projects and tasks to which they were applied. My theoretical sampling with these methods included data processing and analysis tasks with some notable exceptions. By theoretically sampling, I refer to Strauss and Corbin (1998), who state that core categories and concepts are not predetermined from the onset of a grounded-theory study, so the researcher collects and folds in new data to test the core categories. They state that "the aim of theoretical sampling is to maximize opportunities to compare events, incidents, or happenings to determine how a category varies in terms of its properties and dimensions" (p. 202). I also used these methods to define a more meaningful unit of analysis to develop a substantive theory of Ray's coding as writing. Accordingly, I used the following methods to do so.

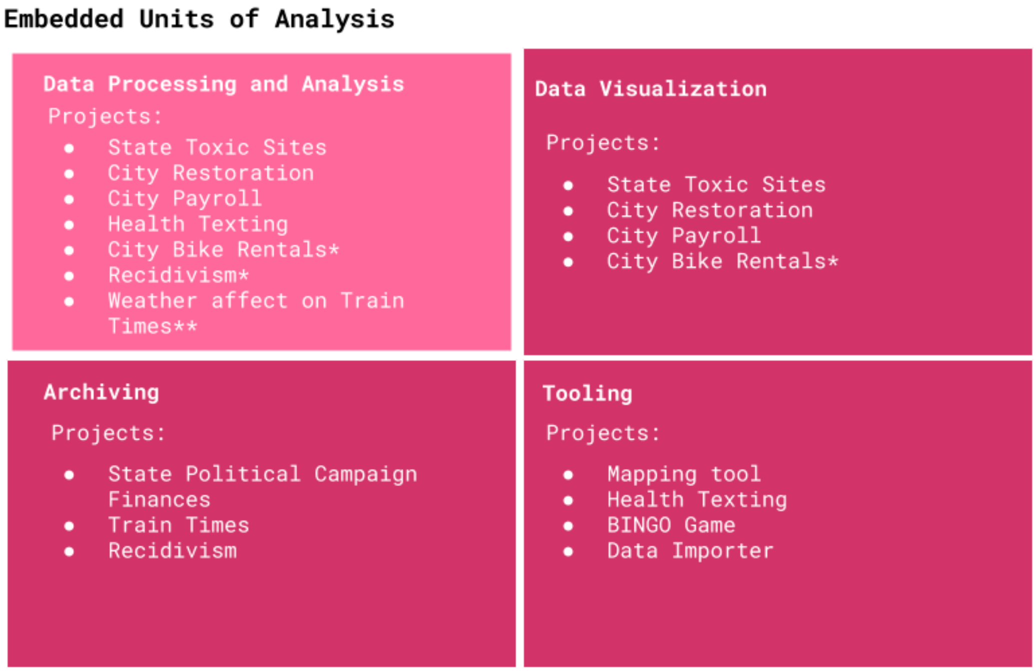
Table 2.7: List of finer-grained methods and their respective project and task.

Method	**Number of Times Applied**	**Project(s)**
Retrospective Account	2	Toxic Sites
Think-Aloud Protocol	5	Texting App (4) BINGO game (1)
Observational Interview	11	Toxic Sites (1) Texting App (1) Recidivism (2) City-Financial (6) City Bike Access (1)

I conducted TAPs to any particular coding task mediated by data-set work, because I wanted to develop a better understanding about how Ray made particular decisions while writing code in these situations. I knew that I could make such a decision, since Ray reviewed his daily agenda with me at the beginning of every observation. I decided to conduct 2 retrospective accounts about the State Toxic Sties project, since a majority of the days that I observed included such work (6/11 days). In addition to these methods, I also conducted 11 observational interviews, which also relate to data-set work. Below, I briefly discuss each method, the decision for the application, and what data each garnered. However, before I discuss these methods, I must also note a particular complexity regarding my decision to focus on this data-set unit of analysis, which serves as the focus of my dissertation.

During the first movement of data collection and analysis, much of the coding work that I observed Ray doing can be characterized as data processing and analysis work. I also observed some days, wherein he visualized the data. A few other days included some coding work to develop a mapping tool for reporters. Overall, though, I garnered a sense that his coding on this team encapsulated many different forms of writing and reading code. Despite this robust sense of Ray's coding objectives in this domain, I decided to focus on his data processing and analysis as a unit of analysis, so I could theoretically sample the data and develop a richer description and sense about this coding work. While he indeed conducted more of this type of coding activity, he also was tasked to code a particular web application for a podcasting team. This application collected data to be processed and analyzed later. Overall, though, numerous observational days included more tooling and development work, than in the previous movement. He also started developing a data importer tool, and he also refactored, or revised, the code for a recreational, browser-based BINGO card game. Additionally, he conducted lots of data-processing work with the objective to archive large sets of data for reporters to more easily use. In sum, I observed and collected multiple data types regarding these other types of coding work: tooling, archiving, and data visualization work. I decided that each of these coding objectives constitute what Yin (2014) refers to as *embedded units of analysis* (pp. 53–56); that is, smaller units of activity that makeup the concerns of the broader case, as represented in Figure 2.4 below.

I recognize the other embedded units as future analytic work to enrich the findings that I present in this dissertation about Ray's coding as writing. Put differently, due to the constraints of time and time spent developing this insight of my own, I report findings derived from my analysis of Ray's coding in relationship to data processing and analysis work. Accordingly, I have started to plan how to transcribe and compare these different embedded units against each other, but focus on the unit of data processing and analysis work, which uses sets of data in a particular way, as the focus of this dissertation. In what follows, I discuss my application of each finer-grained methods as related to the particular unit of data-processing and analysis coding work.



Embedded Units of Analysis

Data Processing and Analysis

Projects:

- State Toxic Sites
- City Restoration
- City Payroll
- Health Texting
- City Bike Rentals*
- Recidivism*
- Weather affect on Train Times**

Data Visualization

Projects:

- State Toxic Sites
- City Restoration
- City Payroll
- City Bike Rentals*

Archiving

Projects:

- State Political Campaign Finances
- Train Times
- Recidivism

Tooling

Projects:

- Mapping tool
- Health Texting
- BINGO Game
- Data Importer

Figure 2.4: A representation of my embedded units of analysis and their respective coding activities and data team projects.

* Project was not observed, but discussed in an interview.

* * Project was observed as a phone conversation about a new data set.

Retrospective Accounts

During the first movement, much the the data-processing and analysis time included the Toxic Sites project, so I conducted 2 *retrospective accounts*, which asked Ray to narrate over a particular coding task. (Greene & Higgins, 1994) note how researchers developed retrospective accounts in response to criticisms of TAPs; namely, how TAPs involve a great deal of mental focus on the task at-hand. Since participants might not be able to verbalize much of what they were thinking during a task, retrospective accounts were created to collect a participant's explanation of an task "without interfering directly with their attention to the task" (p. 118). I used these accounts to collect Ray's thoughts about particular moments during the State Toxic Sites project. Since I could not conduct TAPs on this project during the first movement, the accounts served as a different method to collect task-based data.

I conducted this method only twice, since it was more obtrusive to the typical pattern of observational interactions and rapport that I was cultivating with Ray. Namely, the narration method took much longer than usual, and it focused on prior activity, rather than Ray's current coding situation. Overall, I was sensitive to issues of time taken away from Ray's workday, so I limited the number of times I applied this method. While I conducted interviews that sometimes took ten to 15 minutes during the onset of an observational session, such as *observational interviews*, they served the broader purpose of engaging the tasks at-hand, rather than task conducted in the more distant past. Being sensitive to how much time I asked of Ray, and knowing that I would be asking him to conduct think-aloud protocols in the future, I limited my use of this method to 2 instances. I will note, however, that I provided a fair advance notice of my application of this method, and that it may take some time. Overall, this method, while collecting very useful data, asked him to halt his own activities, which I promised I would not do too much before beginning data collection. Furthermore, I used observational interviews instead, either after a particular coding task or at the onset of any data collection period as way to garner reflective data in the moment or between observational sessions, rather than much later.

As I mentioned above, these accounts were comprised of coding tasks during the State Toxic Sites project. In one clip, Ray needed to write code to combine 3 data set files provided by the state Department of Environmental Protection Agency. In the other, I wanted finer-grained information about some coding activity that led to his discovery that the data was old and needed updating. Both coding tasks involve explicit work with data sets and data processing work. In sum, both moments seemed mundane and not too important at first blush, but once I carved out this particular unit of analysis, I realized how important his coding tasks were in the work to understand the data sets within the scheme of the reporter's developing story.

When conducting both accounts, I adhered to the following general procedure:

1. Prepare a video clip as a digital file to play on Ray's laptop computer.

2. Prepare a statement to state before playing the clip on Ray's laptop, which supplied the broader context of the clip: what project, what particular day, and why he was tasked to start writing a particular script. Note: In this case, I used correspondence from his colleague to re-count this context for him, so it was not conjecture on my part.

3. Asked Ray to verbally account for why he was doing what he was doing during the recording.

4. Also asked Ray to use the mouse's cursor to gesture to particular screen-based discussion items, as he narrated.

5. Screen-recorded (video and audio) the retrospective account.

6. Planned for Ray to stop at particular moments of interest in the clip, as it related to particular analytic codes. For example, during one of the accounts, I prepared some of the following times and questions to ask Ray (the timestamps demarcate the coding activity of concern):

   a. @1:45 and @5:30 – What prompted Ray to write each of these particular ternary conditional expressions, after reading a data set?

   b. @10:40 – What prompted Ray to write a message to Jun on Slack?

My application of the retrospective accounts did not adhere to strict narration only. Interestingly, Ray 'took the wheel' of the video, moving the time around if he had any specific curiosities about the moment. I did not discourage this non-linear approach to the narration in the moment, since it seemed to help him develop an account of the situation too. For example, during one of the account, he self-paused at 2 particular moments that helped him elaborate on how he was making sense of aspects of the data set and attempting to figure out how to create a new data structure befit for it within the context of the project.

Another way that my retrospective accounts differed from Greene and Higgins (1994) is how I prepared a few noteworthy places in each video to ask Ray about after the narration activity. The moments all concerned how his coding activity were linked to his reading and re-writing, if you will, of data sets. As a way to prepare for these post-narrative accounts, I blended Spradley's (1979) *mini-tour questions* with the retrospective account's use of the video, calling it a *mediated-tour interview.*

Spradley defines the mini-tour questions as akin to grand-tour questions, but mini-tours are focused on "a much smaller unit of experience" (p. 88). However, Spradley's interviewing strategies alone proved insufficient, since they focus on interviewing multiple informants, and how informants talk about particular activities and issues. In short, they lacked the actual artifacts and processes that make up an act conducted *in situ.* Retrospective accounts on their own proved insufficient, too, since I was not sure exactly what aspects of the task Ray would narrate or with what specificity. Of course, this unpredictability is important for the inductive development of describing Ray's coding work. However, the unit of experience that I was concerned with – writing and reading code and data sets – meant that I wanted to learn more about the relationship between code and sets of data: 2 particular *mediational means.* I did not tell Ray about this detail, before conducting the retrospective accounts, nor before this mediated-tour interview. My specific concerns about the unit of experience mattered a great deal in my case-study, since this exploratory study includes one main participant of which I could not garner more detail from other similar informants. I needed an interviewing strategy that combined Spradley's strategies to gather details about a specific moment and/or task and the task-driven nature of retrospective accounts. I took elements from Spradley's interviewing strategies – task-specific tour questions from mini-tours – and combined them with the task-driven nature of the retrospective account.

In this case, I needed to learn more about Ray's ability to complete a particular set of tasks during the State Toxic Sites project. By combining the 2 aforementioned methods, I was able to garner Ray's own narration of the task, but also follow it up with specifying *tour* questions – Can you tell me more about what you did here? – since the video was already watched and available for use. I cross-compared this data against previous analytic codes and of other similar moments, where Ray's coding was mediated by reading data sets.

Observational Interviews

Another way I continued to theoretically sample and selectively code during the second movement included observational interviews. Observational interviews ask participants about developing ideas related to the writing work. Katz (2002) notes how they enable researchers to shore up situated details of previous observations in lieu of the participants goals for the day (p. 32). In that way, I was able to clear up uncertainties, test my interpretations, and/or gather up implicit details about the writing work in-between sessions. In effect, I used them as a means to develop a "'practical validity'" (Doheny-Farina, 1993 qtd. in Katz, p. 32). Specifically, I used observational interviews as a way to collect more of Ray's thoughts about his rhetorical choices with regards to any particular code written in the previous session, and how it related to his goals to code for the current day. I usually conducted this at the onset of most observational sessions, as needed. The 11 observational interviews, which I noted in Table 2.7, make up the total number of such interviews as pertinent to this embedded unit of analysis of data processing and analysis work.

Think-Aloud Protocols

According to Schriver (1991), researchers use TAPs to collect data about how writers plan, solve problems, make decisions, use and make sense of texts, and use tools (p. 8). Swarts et al. (1984) discuss how TAPs allow researchers to build theories and understand processes (p. 70). TAPs helped me investigate how Ray's coding work was mediated by sets of data.

In Writing Studies, Swarts et al. (1984) review the multiple ways TAPs help researchers collect data differently based on the researcher's goals. They note how researchers use TAPs to explore writers' processes, which can help a researcher (and future researchers) develop a theoretical framework (p. 68). Relevant to my case-study, I used TAPs to 1) explore what problems Ray was attempting to overcome during particular coding tasks related to data-set work, and 2) begin to understand how Ray's coding practice by more thoroughly engaging the moment-to-moment aspects of a task. Accordingly, I applied TAPs to garner finer-grained data, so I could compare it against other data related to similar coding tasks with data sets.

Additionally, I applied TAPs, since I wanted data that provided richer descriptions of Ray's coding process during a concurrent exercise, so I was not simply relying on more retrospective accounts and interviews. Swartz *et al.* note that participants are of course limited in what they can verbalize, but they highlight how TAPs provide details into the sequencing of a writers process that a researcher would otherwise only wonder about, or how a text alone lacks this more robust picture of writing (p. 53). While there are limits to what a participant may be able to verbalize, TAPs are more revealing than the texts alone, and are complimented by the other data types collected during this period of the study.

As noted in in Table 2.7, I conducted 5 total TAPs. Four of the TAPs are pertinent to the embedded unit of analysis: data-set work. The other TAP raises up one noteworthy exception to the embedded units generated in this dissertation. This other TAP represents, in a way, the development of the other embedded units, since it was around this time in the second movement that Ray began to do different kind of coding work with more frequency.[28] Ray still worked with data sets throughout the second

[28] I also conducted more observational interviews than noted above, but do not count all of them in the table, due to their larger number per observational session. I use this more simple and recognizable number of TAPs as a point of contrast, so I can highlight how my grounded approach to data collection and analysis remained in constant check.

movement in different ways, but he also, as I discuss earlier in this broader section, conducted a greater diversity of coding work—even with data sets (archiving data, for instance). Due to an increased frequency of work that I found distinguishable from data-processing and analysis work, I decided to apply a finer-grained TAP to a moment wherein Ray was reading code from a project written by another developer: a browser-based and interactive BINGO card game. I thought that this particular task to read someone's old code with the objective to 'refactor' it[29] could potentially afford interesting cross-comparisons with the data-processing and analysis work with data sets. My preliminary sense suggests that this is the case,[30] but, as I have mentioned before, my dissertation focuses on the one particular embedded unit.

I followed Schriver's advice to provide participants with a basic form of instructions about how to perform TAPs. I provided Ray with a "scenario approach" (Schriver, p. 11), which involved providing him with a handout (see Appendix A.6) for us to review together. I used this form as a means to tell him what TAPs are and also some example scenarios and statements that I thought might help Ray understand what 'think-aloud' means for this protocol. Also, the form helped me explain that I would simply ask him to "keep talking," if he there was a longer pause in his verbalizations.

As noted above, I conducted 5 TAPs, four of which are pertinent to the goals of this dissertation: Ray's coding work with data sets. Due to time constraints, I only report findings derived from an analysis of one the 4 pertinent TAPs. Later, in section 4.2, I explain more about what TAPs were used and why. Overall, I used TAPs to triangulate developing findings and emerging categories and codes. Such a move to corroborate evidence is due to the limitations of TAP data. Smagorinsky (1994) notes

[29] Broadly conceived, developers *refactor* code by assessing its function, goals, and the more general structure and way of accomplishing those goals. Mainly, refactoring code is meant to improve its readability and its performance. It could be compared to acts of revision involving macro-level changes to a text, which stimulate other types of more micro-level changes later.

[30] For example, Ray had very little context for the code and no documentation to rely on, so he needed to read the code – the HTML, CSS, and JS – to understand how it worked the way it did. Data structures and other structured content in the other files indeed shaped his ability to guess how it worked and his refactoring decisions.

that TAPs are “subject to personal bias, interaction factors between researchers and subjects, problems of interpretation, and other aspects of human caprice” (p. xiii). Accordingly, I do not rely solely on one data type or isolated coding task to construct the core categories develop in this dissertation. These TAPs indeed do not encompass the full-breadth of Ray’s thinking processes during his coding tasks. Yet, the full range of these different data-types – TAPs, retrospective accounts, observational interviews, screen recordings, artifacts, and semi-structured interviews – helped me develop a richer picture of Ray’s coding work. Below, I list the reduced data used to derive findings, as reported in the following 2 chapters:

- 13 of the 32 total days[31]
- 6 Projects (See Figure 2.4)
- 16.5 hours of screen recordings of coding activity
- Project code and correspondence from for four of the 6 projects
- 100 pages of fieldnotes and theoretical memos
- 1/5 TAPs
- 2 Retrospective Accounts
- 11 Observational Interviews
- 3 Semi-Structured Interviews

2.4 Data Integration

In each of the 2 following findings chapters, I begin the *integration* work of the second movement. I provide more methodological detail about how I reduce the data within

[31] One observational day included work on 2 different projects.

this particular embedded unit of analysis, which facilitated the process to generate my 2 core categories: aggregate narratives and provisional texts.

Chapter 3

Findings: Aggregate Narratives

3.1 Introduction: Finding Stories in Data

In this chapter, I examine Ray's coding work with data sets through the question of *What were the recurrent objectives and outputs linked to Ray's coding to process and analyze data?* Over time, I observed how Ray's data processing and analysis work typically was bent toward becoming some form of web-bound artifact or interactive experience. Such objectives and outcomes have become increasingly shared across journalism domains. At a recent journalism conference (*source redacted for confidentiality purposes*, 2015), the editor of Ray's team, Vince, discussed how this team of investigative reporters, designers, and coders support journalism across the newsroom. Vince discussed how the combination of more accessible data and coders enable new kinds of questions to explore more easily and quickly than ever before. Vince put it as, exploring "What the data can see."

Take, for example, how Tampa Bay Times reporter Montgomery (2017b) led an investigation of police shootings in Florida. Montgomery and a team created a custom database of police shootings, which amounts to over 50,000 pages of records and cost the Times over $4,000 to receive from the Florida Department of Law Enforcement

(Montgomery, 2017c). Since 2014, Montgomery and numerous other people explored "what the data can see" by cutting it up into 34 different categories. Montgomery then used these categories to paint a picture of police shootings in relationship to being black through multiple visualizations within an interactive story (Montgomery, 2017a).

According to Vince, such work with data falls under his particular editorial pursuit of a "data-geek" culture, wherein he wants his reporters and developers to know what data are available and potentially viable to collect and archive, so that his team can develop what he referred to as "prescient" sense; that is, helping his team better and more quickly connect data with exigence. Through the cultivation of this data prescience, he explained how he and his team can more quickly and adeptly respond to high-impact exigencies derived from and for the community. During my observations of Ray's coding, I learned how his coding indeed played an integral role in the activity to *find stories in the data*.

Such a notion – to find stories in data – has become a social phenomena in and of itself. Across domains of study and practice, numerous professional communities describe similar experiences linked to the act of telling stories with large sets of data. In section 2.1.1, recall how data journalism has emerged as a veritable field of practice, wherein coding and data-set work involves lots of work to "interrogate" the data sets (Wiggins qtd. in Abelson et al., 2015) or create sets of data of their own (Boyer qtd. in Royal & Blasingame, 2014). Data science, as a broader and emerging practice, shares similar conceptions of data analysis and visualization. For example, Perez (2014), creator of the IPython Interactive Notebook,[32] emphasizes how blending narrative and computational data work was central to the development of this important programming environment for people who work with data sets. In the public domain, data visualization designers Lupi and Posavec (2016) developed a data visualization project, where they sent each

[32] IPython Notebooks are interactive Python programming language shells, which enable people to write multi-line code in relationship with other forms of text and rendered visuals, as well as organize the ideas within easily manipulated folders and files from within the programming environment. In essence, it is a science lab notebook (*cf.* Wickman, 2010) remediated in a computational environment.

other post cards every week for 1 year and each card included the visualization of some small aspect of their everyday life during that week: how many times they said ‘Thank you,’ how many times they checked the time, or what kinds of items they purchased. There is even a podcast devoted to such a subject: Data Stories (Bertini & Stefaner, 2017). Overall, each of these example domains surrounding data-work describe the importance of finding stories within the data.[33]

When I asked Ray about how he got into data analysis and visualization, mapping in particular, he explained how stories and experiences played a significant part. Specifically, he recalled how he purchased the original iPhone, since it “was the only good option at the time” to geocode and timestamp images taken with a camera. In an interview, he described why those particular inscriptions motivated his desire to visualize information as digital map experiences:

> When I went back to that picture, I could know *where* it was and *when* it was. But that *where*, to me, it’s a very powerful, nostalgic quality of something. Sometimes, instead of going to a place, just going and looking at it on a map, a place that I’ve been, either just the road map or the satellite imagery, even if its a top-level, limited view, you can still see so many things that bring back lots of memories. It’s a very powerful way of re-experiencing something. (emphasis added)

Space and time, Ray noted, became important datapoints for new digital and computational expressiveness for him to create, share, and experience. What caught my attention about this moment, among others during my observations, is how he understands that this highly visual information, such as digital images and maps, are also highly textual. Additionally, such an instance is indicative of this broader sense about

[33] More extensive secondary research could be conducted here, such as scouring through major journalism conferences and journals, or how this emphasis on stories, sense-making, and structured data can be historicized across disciplines more generally. However, for the purposes of this dissertation, I review some prominent figures and sources.

data and how coding enables Ray to learn to tell different stories with data—thanks in large part to the textual nature of these digital media and his coding.

As a web developer, Ray reads and writes code that manipulates the structured information that makes up such digital media. For example, the inscriptions that Ray refers to above, geocodes and timestamps, are now commonly found inscriptions[34] as metadata within digital images. Ray elaborated how he understood that such information can be "read in" and combined with other information, which involves coding it as new data structures in whatever project he is creating at the time.[35] Such acts to take information from one or more sources from sometimes different contexts and combine them within a new output file involves work to contextualize such inscriptions for the new project at-hand. Ray's coding work within the newsroom includes a process to contextualize data sets from sometimes multiple sources by making decisions about what datapoints should be translated into the new project as newly structured data.

In this chapter, I report findings about Ray's coding to contextualize data sets from its original context into the objectives of the data-driven team. As I discussed in Chapter 2, I generated findings from the embedded unit of analysis, wherein Ray coded to process and analyze data sets. From this unit, 2 core categories emerged: *aggregate narratives* and *provisional texts.* I report findings regarding how Ray demonstrated an understanding about the relationships between data sets, certain combinations of datum, and narratives important to the conduct in the newsroom. These properties make up *aggregate narratives*; that is, stories developed and hedged through the coding work to process and analyze data. This recurrent objective to tell stories with aggregate information manifest through interviews, as well as key moments when reading data sets, writing code, and discussing data with reporters and editors inquiring about data sets. Ray's coding was implicated in the act to find new lines of inquiry and stories by

[34] As defined in section 1.3, *inscriptions* are representations of social and material phenomena.

[35] This "reading in" of information has a long history in computing. See McIlroy, Pinson, and Tague (1978) and their discussion of the UNIX operating system and its widely known philosophy of computing.

combining and analyzing large sets of data, which was often mediated by data sets.

In this chapter and the next, I analyze Ray's processing and analysis of structured data in and through his coding activity. This chapter reports findings that were generated from the following specifying questions, which I use to develop a thick description of Ray's coding during activities to process and analyze data:

- What are the recurrent exigencies, situations, and objectives that make up Ray's coding?
- What does Ray's process to read and write code to process and analyze data look like?
- What are the properties of the *mediational means* – data sets and code– that shape Ray's coding decisions to create the aggregates?

Such a *thick description* will provide important details that I later integrate with findings generated through a more systematic analysis of Ray's *in situ* coding work. Specifically, this chapter setting the stage of this embedded unit of analysis, as a way to establish the context for the *how*-questions answered in the next chapter. As a result, the next 2 chapters follow an inverted triangle approach to understanding Ray's coding activity. This chapter and the next illuminate how meaning for Ray is not inherent in the code itself, but generated amongst the working relationships between the reader and writer of such code.

I report these findings in the order of the aforementioned specifying questions, so the structure of this chapter is as follows. First, I discuss the methodological decisions made to develop this core category. Next, I discuss some of the properties of the data sets in relationship to their broader guiding questions that mediated Ray's coding work. Then, I report the recurrent exigencies and situations of this coding work. From there, I develop a more holistic picture of Ray's project process, and end by reporting the semiotic modes of the data sets that shaped Ray's coding.

3.2 Method

I met some challenges during the development of this core category. One challenge I faced was to create what Yin (2014) refers to as a substantive *chain of evidence* that illuminates the properties of this objective to create aggregate narratives. I needed to gather evidence and draw connections to help others understand this domain-specific, recurrent objective. I managed this challenge by focusing on how Ray's coding acts function in the context of data-processing and analysis tasks. I conducted this data reduction with the goal in generating a more robust understanding of Ray's greater context and mediational means that served as the substrate for his processing and analyzing of data sets.

Such a methodological move allies with Bracewell and Witte (2003), who called for writing researchers to identify recurrent tasks within a particular domain of writing work, which enabled me to observe, recursively collect and analyze, as well as reduce the data linked to how Ray used and produced data sets in each of the projects noted in the below table (3.1). I adhered to Bracewell and Witte's advice to pay close attention to Ray's recurrent coding tasks and its accompanying discourse and tools. Accordingly, in this chapter, I first examine the accompanying discourse collected and constructed during my observations and semi-structured interviews with Ray. Such data provides insight into the development of the broader core category, *aggregate narratives*, which represents the broader editorial objectives driving much of Ray's coding work with large sets of aggregate data.

The reduced data in Table 3.1 includes projects from both movements of research[36]. I constructed this core category from much of the accompanying discourse in relationship to the broader editorial agenda of the team. As I discussed above in section 3.1, data-journalism teams specialize in working with large sets of data. Ray's editor has made his aims explicit, when he coordinates the team's efforts to gather and synthesize large

[36] See sections 2.3.2 and 2.3.3 for an elaborated discussion about each movement.

Table 3.1: Data used to construct the core category of *aggregate narrative*.

Core Category	**Data Types**	**Data Projects**
Aggregate Narratives	13/32 observational sessions ∼12 hours of screen recordings ∼100 pages of fieldnotes/memos 11 Observational interviews 3 Semi-structured interviews 2 Retrospective accounts Selected project code Selected project correspondence	State Toxic Sites City Restoration City Payroll Health Texting City Bike Rentals Weather affect on train times Recidivism

sets of data to develop a type of "prescience" within their newsroom; that is, Ray's coding is situated within a broader objective to develop a form of knowledge about how certain sets of data will afford them particular avenues for particular kinds of stories. Often was the case, where the term aggregate was used to label the data in relationship to their reporting questions. In other words, Ray's coding was mediated by aggregate information in relationship with finding stories for reporting. Accordingly, the above data includes projects, wherein I was able to observe Ray's coding and discursive activity that focused on processing and analyzing such aggregate information: these *aggregate narratives.*

In the process of coding the data, I identified a recurrent objective to create data sets as "outputs" during his data processing and analysis work. These data-set outputs remained a central objective across Ray's various coding projects in this embedded unit of analysis. These outputs assemble and organize sets of aggregate information that Ray and/or the team deems important for the reporting or visualization steps. In Ray's context, these outputs serve as the end-goal for the data processing and analysis work, because they create sets of data that enable him (or others) to conduct further development of the reporting or the coding of web-based data visualization experiences. Due to this natural break in the project, I was able to develop this particular embedded unit to analyze as a particular type of Ray's coding in this domain.

In this chapter, I report findings derived from an analysis of his coding tools, discourse amongst team members, and the coding work with data sets[37] that Ray uses to contextualize them. From this analysis, I argue that 1) Ray contextualizes data sets originally used for sometimes very different purposes by writing code that organizes, computes, combines, and labels it and more in lieu of the goals of his current project; and 2) Ray's coding was shaped by numerous semiotic resources shored up through both the data sets and the code, which helped him produce variations of aggregate information with the objective to tell stories.

3.3 Analysis and Findings

In this section, I report how Ray's coding work within this context is socially organized by an array of people, tools, and texts to create data-set outputs: aggregate narratives. I also show how Ray contextualizes the data set(s) from their original source(s) into the goals of the current project by comparing the original and output data sets in relationship to the code that enabled Ray to do such work.

3.3.1 Ray's coding domain: Elements of the publishing cycle

As I discussed in Chapter 1, studies on the materialities of writing consider how contextual factors of a domain shape writing practices as well as the domain itself. Scribner and Cole (1981), for instance, contend that "in order to identify the consequences of literacy, . . . , we [researchers] need to understand the larger social system that generates certain kinds of practices (and not others) and poses particular tasks for these practices (and not others)" (p. 237). In sum, this chapter's development of a thick description

[37] See section 2.2.1 for a discussion about mediational means.

attempts to tether evidence about the social system of Ray's data-journalism context, as a way to better understand Ray's individual coding decisions during these projects.

Accordingly, in this section, I describe some of the factors of Ray's coding domain drawn from the data noted in table (3.1).[38] I compared multiple data types as a way to help me develop a thick description of some pronounced factors that seem to play a role in Ray's coding work. Specifically, I report some of the recurrent coding project exigencies, its broader steps to complete each project, the main objective driving those exigencies and steps, as well as what types purposes fuel Ray's reading and writing. By reporting these elements of Ray's coding domain, I establish some broader features of Ray's context to process and analyze the prevalent *mediational means* driving Ray's coding: data sets.

Project exigencies

In my initial semi-structured interview with Ray, I asked him about the typical ways that he comes to projects that work with large sets of data. In response, he noted how many of the initial exigencies of what he deemed the "data-side" of his work adhered to the 3 following types:

1. Reporter / Data team obtains new data set(s);
2. Reporter has been conducting more qualitative, or "on-the-ground," reporting, then adds, or wants to add, data-set element;
3. Reporter has been conducting data analysis with large sets of data, has a narrative of the story with data sets already in play, but needs help to verify particular set of claims or continue data processing and analysis work to refine the story.

I compared Ray's claims against the observed data team projects, and I found that he joined or began these projects along these broad ideas (see Table 3.2 below). In each

[38] I understand that these findings present only a limited perspective and perception of Ray's coding domain and its contextual factors.

of the projects, Ray's exigence aligned with ways that he mentioned in the interview noted above. For instance, Ray learned about the release of the new City Payroll data for the previous year, which he queried from an online Open Data Portal. For the Weather Times Affect on Train Times project, a reporter consulted Ray about their idea to combine their recently acquired government transit data with the news team's database of train headway data. In this situation, a potential effort to process and analyze data prompted the reporter to call Ray and seek his counsel on the matter. During the State Toxic Sites project, Ray was brought on after the reporter, Rosa, had already started the reporting work qualitatively, but then learned about the available state government data related to tracking and managing toxic sites. For the City Restoration project, a reporter had already produced a table of computed values, but wanted Ray's help to test the values against census data, which required more data processing work.

Table 3.2: Ray's project exigencies

Project	**Ray's Exigence***
City Payroll	1
Weather Affect on Train Times	1
State Toxic Sites	2
City Bike Rentals	2
City Restoration	3

* Exigence marked by number, as demarcated in the aforementioned list.

Ray noted how the data projects and their exigencies typically follow a 4-part process:

1. Getting Data

2. Process / Clean / Munge Data

   Combine Data

3. Analyze Data

4. Output Data

He described getting the data through activities, such as "downloading or scraping or stuff like that." He said that data-processing serves as a "cleaning step," which enables them to format the data into a "more usable" state. He said that this processing work helps on the "programming side" or perhaps to help a reporter or himself use the data in a particular program application. He also noted how "there's a possible combination step," which is sometimes necessary when they work with multiple sources. After the data is in a usable state, Ray said that they are ready for analysis. The analysis work helps the team "say that you know what the trends are or what are the median/means—that sort of stuff." When all of this work is complete, he said that "there's usually an output step, since we're usually building an interface or some sort of output, so its outputting the data in a way that we can build a visual thing." I compared Ray's narrative of the data-work steps to the documentation in each project, and they all note these steps in the README file. In Figure 3.1, I use the State Toxic Sites project as means to show this typical data-work procedure.

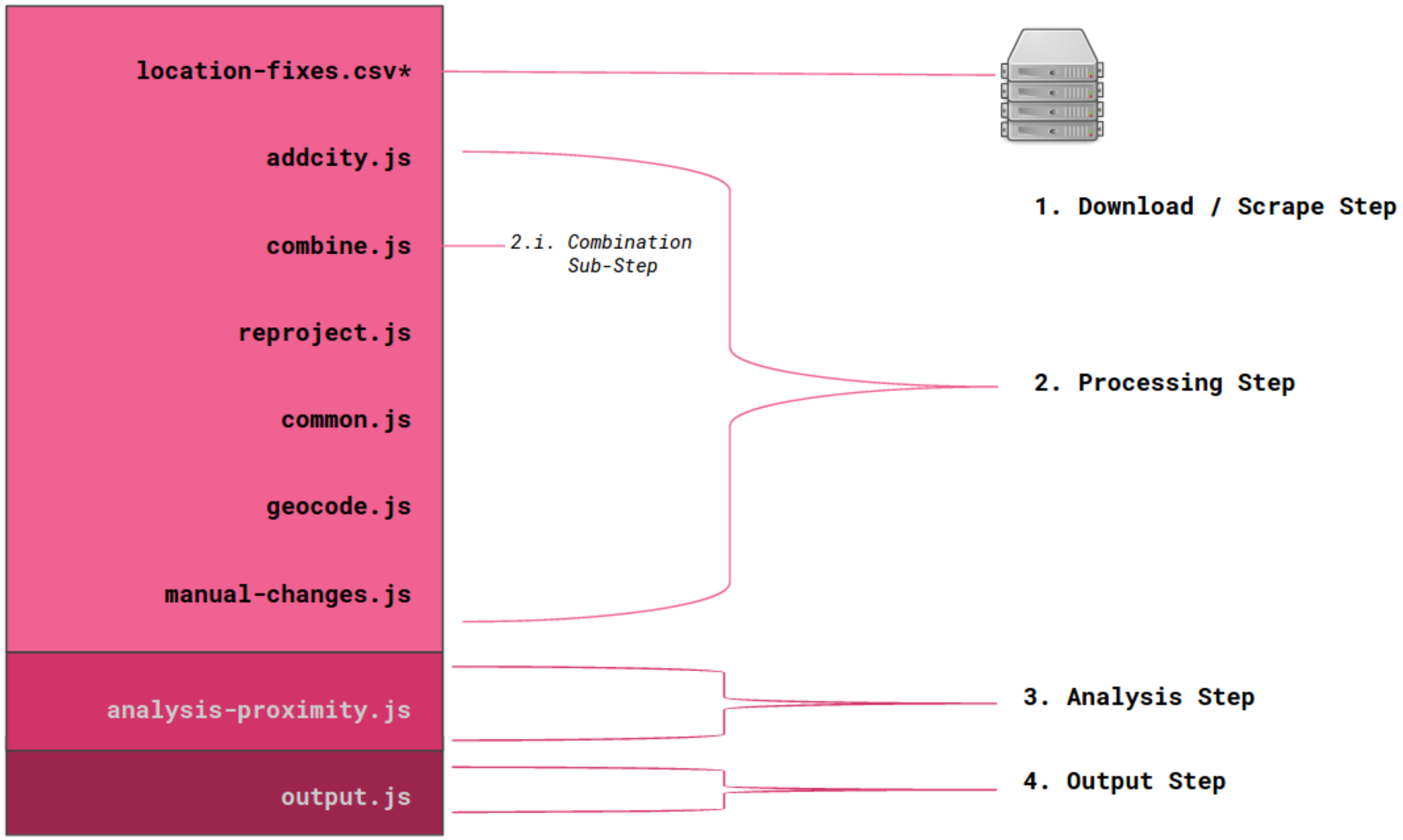


Figure 3.1: Representation of Ray's coding work to process and analyze data as compared against interview data and a project's documentation and files.

* At the end of this project, the team combined 7 data sets, which were both downloaded and requested.

One project did not quite fit this particular framework: Health Texting. During this project, Ray first developed a web application, which enabled one of the newsroom's podcasting teams to conduct a mobile texting campaign. What sets Ray's exigency apart from the others is how the web app – specifically, its texting prompts issued to tens of thousands of participants – collected the data that Ray processed and analyzed during and after the project. Accordingly, he did not simply obtain or download a data set. Instead, Ray created a per-person instance database of responses, resulting in a large corpus from which to sample. So, while this project could be categorized as #1, *Obtain a new data set*, this project led to new questions about the complexity of Ray's coding context and the situations in which he reads and writes code to process and analyze data. Notably this particular instance led to new questions about how Ray's coding situations were quite diverse, since the Health Texting project required him to develop a web application before working with the data sets. Recall how this question about the diversity of Ray's coding situations helped me develop my embedded units of analysis (see section 2.3.3); that is, reading and writing code to

- Process and analyze data,
- Visualize data,
- Archive data, and
- Tooling new technologies for reporters.

In this chapter, I focus on Ray's processing and analysis of data, and the Health Texting project serves as variance in how Ray described the typical ways he came to data-driven news projects. This particular exigency directed my attention to how Ray conducted other kinds of coding activities with different objectives, mediational means, and people. In the remainder of this chapter, I describe some of the complexities of Ray's coding domain through other artifacts and situations across the projects noted in Table 3.1.

Storyboard matrix and publishing cycle

Another artifact enriches the picture of how Ray responds and develops exigencies within this domain. During an observational interview, Ray mentioned that the team maintains a storyboard with project ideas. (For a picture of the board, see Figure A.3 in Appendix A.8.) In the first semi-structured interview, Ray noted how the board organizes their project's within a 2-week publishing cycle, which Vince developed for the team to produce content more regularly. In fact, the 2-week publishing cycle was a new editorial plan, which Ray noted began 4 months prior to this interview.

They positioned this rectangular whiteboard along its long-edge on a movable cart. The team places small post-it notes with potential project names along different labeled axes. The axes delineate the different statuses and judgments about such projects. The **left side of the whiteboard** includes categories, such as *Ready to Build*, *Actively Exploring*, and *Love*. *Love* includes a continuum along a horizontal x-axis, which is marked as projects moving between Vague (far-left) to Specific (far-right). Accordingly, when they move a project more to the right. it means they "refine" the story idea. A y-axis delineates an affective "learn to love" movement. Essentially, the higher up along the vertical y-axis, the more desirable the story. **Along the right side of the whiteboard**, the team identifies a particular publishing date within their 2-week cycles. Such a move to place a project along this rotating calendar puts it into a tighter time constraint and more dedicated work to get it picture-ready. According to Ray, however, much of the board includes projects that need development and take a varied amounts of time to complete, usually longer than 2 weeks, so it helps them organize their time and energy with an editorial purview. He notes how:

> **Ray**: . . . each of us [data team], less me, have ideas that we throw out. You know that poster board with all of those post-it notes, those are all ideas.
>
> **Chris**: That's always there?
>
> **Ray**: Yeah. But they're not necessarily good ideas. They're just like, 'These

> could be interesting,' but we just don't know.
>
> **Chris**: So you keep them there in case you need them?
>
> **Ray**: Yeah, and a lot of them now, since we switched to the 2 week publishing thing. You know, the every other Tuesday. We switched in July, which really has been helpful in producing and producing regularly. But, a lot of those ideas, a lot of those post-it notes, require more than 2 weeks to get out. So we haven't found the stride of really thinking ahead with some of those. Longer-term is very relative here. Like a month is sort of our more longer-terms. I mean there's definitely pieces and stories that have gone on farther than that [1 month]; especially as far as the active work—those usually go on about a month.

This whiteboard and the publishing cycle that it organizes marks how the team is attempting to juggle multiple potential stories, which vary in their length of time to go from idea to published report. The whiteboard also opens up a different perspective of Ray's exigencies and coding situations in lieu of the team's newer coordination efforts. Specifically, these contextual details helped me learn more about how each project, wherein Ray codes to process and analyze data sets, includes recurrent, yet dynamic, situations—situations that Ray both respond to and creates with and through his coding. Namely, it started to illuminate how Ray's role and contribution(s) in these data team projects varied, due to project dynamics.

Ray's slippery roles and project dynamics

Ray indicated above how nearly all of the post-it notes on the board belong to the others on the team. He refers to himself as the "technical" person on the team, and in an earlier observational interview he suggested that anything that he creates isn't as "hard-hitting":

> **Ray**: Reporters have their ear to the ground; they know these subjects

> better than we do. [But] they don't have the data skills to do this stuff that they would like to do. That's where we [developers] come in. Most of this stuff that we [developers] do on our own is not really as hard-hitting as the stuff that we do with reporters. We're just not that tied to stuff. We are really good at getting data, [and] we're really good at processing data, so if someone can be like 'Hey, here's this is an awesome data set that no one else has, we think there's something in here.' That's usually good.

Due to this more "technical" role, Ray often bounced across reporting projects, and rarely started a project from the onset. For example, during my 5 months with Ray at WWWC, he initiated 1 project: City Payroll. He retrieved the data from an open data portal online, processed and analyzed it, sent preliminary draft ideas to his teammates for feedback, and even created a draft report with visualizations. This different opportunity for Ray came about, since many of the staff were going to be gone for a few weeks. Consequently, the team needed some simpler projects with quick turnarounds during their newer 2-week cycle. Ray noted this in an interview:

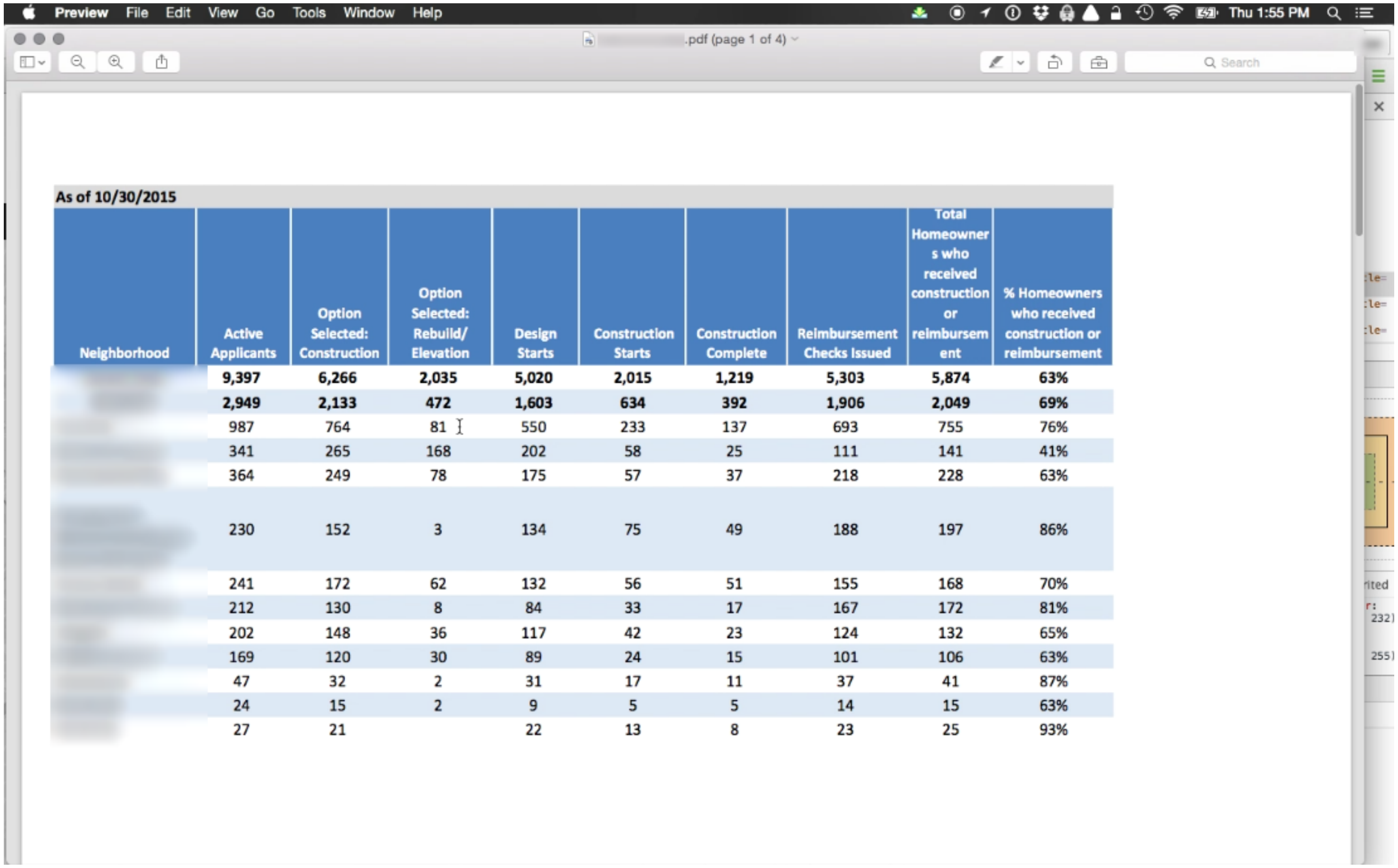
> **Ray**: So, the background [for City Payroll]; I think I explained yesterday that we have a couple Tuesdays coming up where no one is going to be around, so I was trying to find a simple project, so I went into the data portal for [city name redacted]. I basically just sorted it by the newest data sets, and this one was released in November.

Despite all of this work on Ray's part, the report never was published. As per a discussion on Slack, the editorial decision not to publish it was due to how the data did not have enough context surrounding Ray's initial findings. Vince noted how "it [the findings] feels like it needs to be reported out . . . even if we're not naming names it seems like we need to call some of these departments and try to get explanations about what these jobs do, about why they might have more overtime than others." Ray replied by saying that he wouldn't be able to do so within the alloted time to publish it: “Do we have a reporter that might already be connected to the payroll agency? I am not entirely sure I could do that (well) in the next few days. But maybe thats my inexperience talking.”

Overall, this project represents what seems to be a newer experience for Ray and his more usual ’technical’ role. Ray has worked in the journalism domain for 3 years prior to this position. Yet, this exchange in connection with his earlier comment about being a developer who specializes in the processing, analysis, and visualization – this in contrast with reporting work – implicitly indicates how Ray’s role typically revolves around those 3 coding-focused areas within any given data-driven reporting project.

More typical to Ray’s coding work are projects like that of the City Restoration. For this project, a reporter asked Ray to process the data, so the reporter could test for any potential disproportionate issues for applicants along particular demographics. To accomplish this analysis, the data sets needed to be aggregated into a different perspective. The original state government data was delimited by *per neighborhood borough*, while the available U.S. Census Bureau demographic data is delimited *by county*. Accordingly, Ray needed to write code to reconcile this difference between the aggregate data before conducting the analysis. After conducting this data processing, then the analysis, (which did not identify any correlations), Ray created an interactive map. The map enabled readers to explore the percentage of “Construction Starts” in each neighborhood with more than 50 eligible applicants.

Interestingly, after Ray completed the map, Ray told me that he was confused about the numbers in the original table of data provided by the reporter. (See Fig. 3.2 below.)



As of 10/30/2015

Neighborhood	Active Applicants	Option Selected: Construction	Option Selected: Rebuild/ Elevation	Design Starts	Construction Starts	Construction Complete	Reimbursement Checks Issued	Total Homeowners who received construction or reimbursement	% Homeowners who received construction or reimbursement
[redacted]	**9,397**	**6,266**	**2,035**	**5,020**	**2,015**	**1,219**	**5,303**	**5,874**	**63%**
[redacted]	**2,949**	**2,133**	**472**	**1,603**	**634**	**392**	**1,906**	**2,049**	**69%**
[redacted]	987	764	81	550	233	137	693	755	76%
[redacted]	341	265	168	202	58	25	111	141	41%
[redacted]	364	249	78	175	57	37	218	228	63%
[redacted]	230	152	3	134	75	49	188	197	86%
[redacted]	241	172	62	132	56	51	155	168	70%
[redacted]	212	130	8	84	33	17	167	172	81%
[redacted]	202	148	36	117	42	23	124	132	65%
[redacted]	169	120	30	89	24	15	101	106	63%
[redacted]	47	32	2	31	17	11	37	41	87%
[redacted]	24	15	2	9	5	5	14	15	63%
[redacted]	27	21		22	13	8	23	25	93%

Figure 3.2: Screenshot of the reporter's spreadsheet of data used in the City Restoration project.

During an observational session, Ray paused and described how the structure of the data suggested multiple kinds of hierarchies amongst each column of values, which opened up multiple readings about what the numbers may or may not represent:

> **Ray**: I am a bit confused on the numbers. Not on the math, but on what the numbers are. And what they mean. Like this whole spreadsheet. Like I don't understand what is a subset of another thing. It doesn't make much sense to me.
>
> [pause]
>
> So, the idea is that the 'Active Applications' and the people who opted for 'Construction' and the opted for 'Rebuild' should be a subset of these [the prior 2]. Then the 'Design Starts' and then 'Construction' – I would assume, like, 'Design Starts' should be a 100% of these [opted to 'Construct' and 'Rebuild'], but, . . . Or, I guess that some could not have gotten to that phase yet. But, then, so then 'Construction Starts' should be a subset of 'Design'; 'Construction Complete' should be a subset of that ['Construction Starts']. But then you have 'Reimbursement Checks' which should be a subset of all of it, but it's more than 'Design Starts'.
>
> But, I don't really know what these numbers really mean. So, I just don't want to be the one making decisions [saying while laughing] about what we are actually displaying [in the interactive map].

In lieu of this reading of the data, Ray sent the reporter an email, asking for them to clarify the relationships that he notes in the above quote. Here, Ray used the structured data provided to him, as is, by the reporter. However, after coding the interactive map, which involved a process to navigate between the data set and his project code, created an environment in which he started to interpret the data in multiple ways. Here, context was missing from Ray's understanding of the structured data.

Similarly, during the State Toxic Sites project, Ray was brought onto the project intermittently, after it had been developing for a few weeks prior. Even after discussing the data and project with the producer, Jun, and conducting a few small tasks, Ray had about a 2 week gap in his involvement with this project before it was deemed as a project ready for publication. Prior to the following exchange below, Rosa and Jun were geocoding the site location data, which costs the team money to perform. As a way to potentially save money, they asked Ray was to match up 2 data set files from the same government source, since the 2 files may contain some overlapping site information. Below, Ray provides me with some context about what he is feeling after jumping back into this project, after a longer hiatus:

> **Ray**: It seems like most of the time I have no idea about what is going on with the projects coming, which is sometimes the case, because I usually come on after the data has been gathered and looked at a little bit. So sometimes it's ... often reporters ... I don't know if I miss the initial like context meeting, but I feel like they are just, like you know, throwing out a bunch of acronyms and assuming everything. And, for me, since I don't live in [city name redacted], and I am relatively new about stuff ... I feel like maybe ... I don't know. Ahh, I will pick it up, but I also feel like there's never that step back to, you know, describe it a little more.

Here, context shapes Ray's overarching understanding about the project aims still under development, and how such aims relate to his coding task. Even if the task may be more simple, as it was in this case of matching location data, Ray still needed to work with the data sets with the code that he had already written prior to this task. Such a small task may become larger, if the project itself is not clear, nor its folder and file organization, nor any or little documentation about where the data originated and what processing and analysis may have been conducted thus far. Such is the case with the State Toxic Sites project.

Ray made the comment above was in response to 2 main problems: 1) Ray hadn't been engaged in the project in any significant way yet, so he had very little contextual information, such as the origins of the multiple data sets and existing source-code scripts in the project directory, and 2) the project directory had little to no organizational scheme, file-naming scheme, or any documentation to help understand the overarching goals of the project and its corresponding files.

Indeed, the project did not have the typical organization that Ray typically uses. Ray noted how the other team members typically organize their project files differently – each with their own idiosyncracies. In this moment above, the State Toxic Sites project folder had a `/data/raw` and `/data/census` organizational structure. Within the parent folder, there were 28 miscellaneous data files and source-code scripts that Jun and Rosa had been collecting, creating, and using. Despite the relatively easy coding task, Ray had difficulties understanding what data sets to use or why (see Fig. A.4 in Appendix A.9). Consequently, Ray requested that Jun provide some more details over Slack before he started to write the JS script:

> **Ray**: So, maybe you can describe the data a bit again. What are the 'abandoned-sites-20151117' and the '[redacted]_active.csv' all about?
>
> **Jun**: abondoned [sic] are sites without a Licensed Site Remediation Professional.
>
> **Ray**: So that's the main one we want to look at, right?
>
> **Jun**: Right. . . . because [redacted]_active is the full list of contaminated sites. . . . but that list is pretty outdated.
>
> **Ray**: Ah.
>
> **Jun**: so i [sic] was looking at matching PI numbers in [redacted]_active for the location and geocoding the rest of abandoned-sites-20151117
>
> **Ray**: Ok. That makes sense. I can do that.

Overall, these situational instances reveal 3 main take-aways about Ray, his domain, and the relationship between his coding and data sets. First, Ray's role on many of the projects is supportive and task-based, and his coding work is deemed the technical, development side of the data-driven work in this domain. Perhaps, due to his more fleeting involvement per project, and/or because he was still quite new to this particular context at WWWC, he requested more contextual information from his colleagues to help him do his coding work. At the time of this study, he seemed cognizant about how important contextual information was for him to interpret and construct coding tasks in relation to data sets.

Such findings begin to show how Ray needs contextual information to help him perform his coding work in his shifting roles across projects; namely, how the data sets, when understood more explicitly as texts, exceed their signs. Indeed, the signs and semiotic modes of the data sets are not plain or straight forward for Ray. For instance, the City Payroll project reveals how more reporting work would be needed to deepen the insights garnered by Ray's initial analysis of the data. Both the City Restoration and State Toxic Sites projects reveal how Ray needs more contextual information about the original data sets and the reporting aims, so he has a clearer sense about how to approach his coding work. As Witte (1992) would remark, the data sets exist in a stream of multiple contexts in addition to other texts within Ray's particular context: Slack messages, file and folder names, code, etc. Furthermore, information that exists and must be shored up by asking colleagues, reading documentation (if available), or how the folders and files are named and organized may help or inhibit Ray's sense-making of the project. Some of these contextual elements include reporter exigencies, aims, and the origins of the data sets. This contextual information, which is not easily chronicled by the data sets themselves, helps Ray *read* the data sets and *write* code to manipulate them within the scope of the reporting objectives.

These initial findings, regarding Ray's shifting roles and his need for contextual information, led to more specifying questions regarding the recurrent objective threaded throughout these data-driven reporting projects: finding stories with aggregate data. In the remainder of the chapter, I report how I used the accompanying discourse during each project to develop the core category of *aggregate narratives*, which represents the main recurrent objective that shapes and is shaped by Ray's coding work.

3.3.2 Identifying an overarching objective: Talking and coding aggregates

As I mentioned in section 3.1, one of the major objectives driving Ray's coding includes working with aggregate information within different types of data sets. The core category, aggregate narratives, emerged out of both discourse amongst the team, interviews, with Ray, and a closer examination of his code produced within this embedded unit of analysis. This core category originally developed out of the cover-term, *aggregate*, which Ray used consistently enough over my observational period to warrant a closer look.

One such example case of Ray using aggregate involved Ray and a reporter discussing a potential idea to combine 2 different data sets for a story about the effect of a natural disaster on local train times. During a phone call, the reporter asked Ray about the data team's train headway time data that the team began collecting during the previous summer (about 6 months ago). The reporter asked Ray if any potential stories could be developed by combining the team's headway data with their recently acquired "On-Time Performance" (OTP) data from the city's public transit office. (See Figure A.2 in Appendix A.7.) Specifically, they asked Ray if there is any way to see if the natural disaster slowed train times in a particular area. Their conversation proceeded as follows:

> **Ray**: I don't think so [we can compare these 2 data sets], since we don't have any [headway] data prior to [natural disaster redacted]. But the data we're collecting [for the train headway times] is every train, every stop, everything, so if there's . . .
>
> **Reporter**: [*Replies to Ray, but unintelligible.*]

> **Ray**: Yeah, but it [Reporter's transit data] is so aggregate that it'd be hard to make any comparisons [with our headway data].
>
> **Reporter**: Hmm. Ok. Look around, tell me what you think looks interesting.
>
> **Ray**: Yeah, I can grab these numbers out and see them a little bit. Maybe send them over to you.

Ray's response to the reporter's idea hinged upon his quick read of the OTP data. The OTP data, he said, "is so aggregate," that it would be difficult – if not impossible – for him to test any of the reporter's questions against the team's headway data. In this situation, among others, Ray's use of the word *aggregate* served as a term for his understanding about the current and potential relationships among the inscriptions organized in and across these data sets.

By understanding these data sets and the code that Ray writes explicitly as texts, moments such as this can be examined as reading and writing activities, showing how data sets mediated Ray's coding to process and analyze information within the scope of developing stories. Ray's particular use of the term, aggregate, led me to inquire about this further in an observational interview with him:

> **Chris**: Sounds like [reporter name redacted] was looking for some stories based on this data?
>
> **Ray**: I think [reporter name redacted] angle is that due to damage from [natural disaster name redacted], trains have been running slower.
>
> **Chris**: Trying to verify that, then?
>
> **Ray**: Yeah. In theory, [hovers mouse over a column in the data set with data about region of interest], if that was so, these numbers would be coming down. Which they sort of are, but not significantly. And this is so aggregate that it's hard to have any idea.

> **Chris**: It 'is so aggregate,' what do you mean?
>
> **Ray**: Well, this [hovers over example datum] is saying for the month of January in 2010 that for this line this was 92% on time, which I think is less than 8 minutes late. But that's thousands of trains, and then they're actually measuring the time it gets to the end of the line, which isn't a bad metric, but it could be 20 minutes here, and then make up the time along the way.
>
> **Chris**: So I hear you saying that this 'so aggregate' means that the data doesn't help [reporter's name redacted] do what they want to do with it.
>
> **Ray**: Yeah, it's a good overall metric [for transit], but it's hard to say what a 2% difference really is. We can know from these numbers that it is significant or not, but we don't know necessarily what it [the particular datum over time] means. Like, "What could cause it [the 2% difference]?"; there's just so many unknowns.
>
> **Chris**: It's hard to unpack that 2%, then?
>
> **Ray**: Right. We know if it's significant or not, and doesn't look like there are significant changes, but . . . something like this would be great if we had like 10 more years, so we could see if there are any significant differences. But then we couldn't contribute it to [natural disaster redacted]; there's just too many unknowns.

In the above interaction, Ray walks me through some of the difficult epistemic work that he and his colleagues perform. I asked Ray about his conversation with the reporter. Specifically, I wanted to learn more about how and why he used the term *aggregate* during the phone call; namely, how that term seemed to have a deeper meaning behind it, since he advised the reporter that the team's headway data would not yield any insight into their investigation. In this situation, 'so aggregate' meant that the original transit data, which was requested by the reporter from transit, only affords them a certain perspective of train punctuality, due to its ”metric,” said Ray. Consequently, the transit data, and its aggregated values, may yield certain kinds of insight about the train times over time, but, as Ray notes, a 2% difference contains “too many unknowns.” Ray lists off rival hypotheses about how significant or not the percentage may or may not be. And, besides this point, the transit aggregate is vastly different than that of the per instance nature of the team’s headway data: “. . . every train, every stop, everything.” Consequently, Ray would need to “pull out” some data from their database, then compute a similar aggregate from it. However, as Ray indicates during his discussion with the reporter, they have not been collecting the headway data long enough, nor do they have the sufficient variables in the transit data, to be able to test for any significance of interest to the reporter.

Such a moment points to the objectives surrounding Ray’s processing and analysis of data in this newsroom workplace; that is, Ray codes within an editorial agenda to find stories in and around data by asking questions of and about data sets. Such an objective led me to examine how this cover term was taken up across the projects examined in this chapter.

In Table 3.3, I gathered explicit uses of “aggregate” during my observations, so I could pursue the question of its potential relevance and importance to Ray’s coding work.[39] In the table below, I provide the context of the utterance, and the set of analytic codes derived from it.

[39] I located these uses of aggregate through a file-search feature in my text editor. I use SublimeText, which is a code editor. I used its feature that can search files within project directories. I searched across all of the transcribed observational and interview sessions, as well as project code, within the set of reduced data.

Table 3.3: Data used to examine the cover term: *aggregate*

Cover Term Use	**Context**	**Codes**
"We may use individual points, but those points aren't as important as the **aggregate** view."	First semi-structured interview; response to question about showing trends with maps.	Perspective (see particular trend), significance, preference, datapoints, aggregate as adjective
"Yeah, but it [the data] is so **aggregate** that it'd be hard to make any comparisons."	Ray states during observational interview after phone call with reporter about train times project	Perspective (particular set of values), compare, difficulty, combine, potential, aggregate as adjective

Continued on next page

* Italicized text within <> participant represent actions.

Table 3.3 – continued from previous page

Cover Term Use	Context	Codes
"In theory, if that was so [the trains were affected by the natural disaster], these numbers would be coming down <*hovers mouse over data with region of concern*>* Which they sort of are, but not significantly. And this is so **aggregate** that it's hard to have any idea."	Ray states during observational interview after phone call with reporter about train times project	Perspective (particular set of values), interpretation, difficulty, reading, significance, hedging, aggregate as adjective
`// Create aggregate data`	Code comment in `results.js` JavaScript file for health mobile project.	Perspective (particular set of values), change (from one perspective to another), aggregate as adjective

Continued on next page

* Italicized text within <> participant represent actions.

Table 3.3 – continued from previous page

Cover Term Use	Context	Codes
“So what I’ve been looking at now is the more **aggregate** view . . . ”	Ray states just before writing code to output some analysis results as CSV files for health-mobile team.	Perspective (particular set of values), change (from one perspective to another), aggregate as adjective
“So now we want to **aggregate** that.”	Ray states while coding data sets for health-mobile team.	Perspective (particular set of values), change (from one perspective to another), aggregate as verb

Continued on next page

* Italicized text within <> participant represent actions.

Table 3.3 – continued from previous page

Cover Term Use	Context	Codes
<*Scrolls up to the* `// Go through Questions`> We got these different questions . . . We've essentially put in the data <*highlights the* `data: {}`> – that's what those CSVs are. We've got all these different questions. We got the response data, and then we're just trying to find a way . . . <*scrolls back down to the* `// Create aggregate data`> to structure that, and then a way to back-support it.”	Ray talking about `results.js` code for the health mobile project, which will “Create aggregate data”	Perspective (question data), change (entire database to question data), find, structure (verb), aggregate as adjective
“sometimes there's a disconnect between raw and aggregate data. You can't go from **aggregate** to raw, or from one **aggregate** to another.”	Ray states during observational interview after phone call with reporter about train times project	Perspective (one set of values to another), change (one set of values to another), raw data, aggregate data, uni-directional, aggregate as noun

Continued on next page

* Italicized text within <> participant represent actions.

Table 3.3 – continued from previous page

Cover Term Use	**Context**	**Codes**
* /reporting: JSON of aggregate data from the database;	README.md documentation file for Health mobile project	Perspective (selected data), change (entire database to selected data), purpose, output, JSON, aggregate as adjective
"Some **aggregate** reporting on [Health Mobile project]. Updated every couple minutes."	Ray's summarizing statement atop a `reporting.html` page that renders a data feed of the health mobile texting campaign.	Perspective (selected data), aggregate as automated feed output, aggregate as adjective

`// Aggregate question data`
`_.each(questions, function(q, qi) {`
`...`
	Code comment from Health Mobile `results.js`	Change (`questions` to aggregate), perspective (by each question), aggregate as adjective

Continued on next page

* Italicized text within <> participant represent actions.

Table 3.3 – continued from previous page

Cover Term Use	Context	Codes

`* 1. POSSIBLE TODO: The following process will gather data from the [public transit name redacted] page and put it in the MySQL database and aggregate it by parent stop:`
`node turnstile.js`
	To-do in the train headway project's `README.md`.	Change (transit feed to parent stop), perspective (by parent stop), process, aggregate as verb

* Italicized text within <> participant represent actions.

After coding Ray's uses of *aggregate*, I learned that he uses it to denote either a prior state, current state, potential state, or the creation of a new state of a data set. The analytic code *perspective* describes how Ray communicates – some times explicitly, other times implicitly – how data sets (aggregates) provide their users potential vantage points about some particular phenomena in question. For example, Ray wrote code to create a new set of data delimited by "parent stop" for trains or by "questions" for the health campaign data (among others). Depending upon the aggregate's *perspective* – that is, Ray's understanding about how a data set is sliced or dimensionalized – the data can be either used as-is, combined with other data, or (potentially) *changed* to suit the needs of the developing story. In sum, his use of *aggregate* denotes both his ability to understand the data and how to *change* it, so he can or could make it useful within the current project.

For Ray, *aggregate* communicated the germ of an idea for future coding work to contextualize data from its original source(s) into the developing story. The aggregate narrative remains an integral objective for the team as a whole and specifically Ray's coding work to write code that organizes set(s) of information. In this case-study, I focus on Ray's reading and writing of data sets, which he did in tandem with reading and writing code. However, as a way to learn more about this cover term, *aggregate*, in the scheme of the broader objective of finding stories, I searched for another cover term, *story/ies*, across the projects, transcriptions, and Slack logs. I conducted this search and analysis as a way to corroborate claims about the 'finding stories' objective and its relationship with aggregate information.

I widened the search to all of the available Slack logs, so I could locate any evidence of communication related to storyfinding. In so doing, I generated 247 results across all the data. I reduced the search results to 23 situations, which all occurred across 16 distinct days. Since I included all of the Slack logs, these 16 days included projects noted within Table 3.1, as well as 2 additional discursive artifacts from Ray's colleagues from different projects. One artifact included an interview, wherein Phil (the other developer on the team) discussed some relationships between finding stories and data sets. The other included an interaction between the editor, Vince, and the producer, Jun, who were thinking through how they are going to process, analyze, and visualize data regarding a developing story about school segregation. Excluded search results included items of no concern, such as URLs repeated from template files included within each project code directory; examples of excluded results include '`.../wwwc/story/...`'.

Recall how *aggregate* denotes Ray's ability to understand the data from a particular perspective, and how to potentially *change* the data, if needed. I used Ray's constructed senses of aggregate data and the *perspective(s)* it affords or not to inform this pass through the *story/ies* data. In other words, I built upon the preliminary findings generated by the above analysis of the discourse surrounding aggregate information. Specifically, the construct of *perspectives* raised a new specifying question: *How else did Ray and his colleagues discuss aggregate information in situations about stories?* Table 3.4 provides the generated codes from this pass through the 23 situations and representative examples.

Table 3.4: Codes of aggregate *perspectives*.

Perspective Code	Code Definition	Example
Perspective		
Trend vs. Single-Numbers	The aggregate data may provide a perspective about a trend or about some current moment (single-numbers).	**Ray**: Ultimately, I think we were hoping, with the correlation stuff on the State Toxic project], is that you can show more definitively that this [disparity] is going [on]. This [project] is single-numbers, which are powerful, but, you know, visually and narratively are not as impactful.

Continued on next page

Table 3.4 – continued from previous page

Perspective Code	Code Definition	Example
Degree	Data tested initially tested for more compelling issues: correlations or possible predictive qualities and disparities. If not found, used in different 'degree'.	**Ray**: Well, we had to do the combination to do the analysis [for the City Restoration project] to figure that all out. Then, based on the analysis [with city-data and census-data] and those results, we were hoping for a more distinct disparity, at least with the data we had, that was not there. We ended up making a basic map, so still using our work, but not in the degree that we hoped.
Angles	A sense that data can be interpreted and analyzed from different contextual 'angles'.	**Ray**: . . . I think what they're trying to do is be able to talk about the idea from a health standpoint–about taking in so much information. They probably have other angles of this story, and this is just one side of it.

Continued on next page

Table 3.4 – continued from previous page

Perspective Code	Code Definition	Example
Slicing	The act to dimensionalize and/or reduce the original data into a goal-directed set(s) for a particular analysis.	**Ray**: So we took a slice – July through August and just commuting days – to get it [the data set] down to a manageable thing. And then this [pointing to one of the completed project's map] looks at when bikes don't have enough docks.

The codes above were developed by analyzing the 23 situations, wherein the word story/ies was used either on Slack, during an observational sessions, interview, or retrospective account or TAP. In what follows, I compare these codes against each data-driven project and report what can be learned about the analytic codes through such a comparison. In each of the sections below, I define then discuss how each code was observed during particular projects.

Trends vs. Single-Numbers

Code Definition: The aggregate data may provide Ray and/or a colleague a *perspective* about a trend or about some current moment (single-numbers).

Train Times and Recidivism: Reporters desired to test the Train Times and Recidivism data for trends, but Ray consulted them that such an analysis could not be performed, since the data could not support such objectives. In the Train Times case, a discussion about the data led to an impasse, since Ray had a sense about the aggregate *perspective* of the city-data, which clashed with the team's finer-grained aggregate transit database.

State Toxic Sites and City Restoration: Ray initially tested both of the State Toxic Sites and City Restoration projects for different types of trends across race and other demographics: correlation (Toxic) and linear regression (City Restoration). Neither analysis produced significant results, so Ray and the team opted for using what Ray referred to as 'single-numbers" analysis.

By single-numbers, Ray was describing how the analysis only provides information about a particular moment in time about some issue in question. It also denotes how the data, the analysis it affords, and the results it yields are less complex. It may yield some interesting numbers to prove a particular point. For example, in the State Toxic Sites project, the reporter used the results that showed the disparity between people groups of color and whites, as well as across income levels. However, the single-numbers does not provide any more evidence beyond the current status of sites. In other words, the data can not help them claim that these disparities significantly relate to one another, nor are associated with each other within some trend.

City Payroll: During the City Payroll project, Ray deliberately choose a simpler set of data from a recently released city database with city employee earnings for a year. Ray coded the data into different dimensions, using various percentages to convey possible story points. Ray shared these single-numbers on Slack with his colleagues, who often responded with their questions and ideas for future inquiry. (See *Data slices* section below for more information.) Ultimately, as aforementioned, the story was deemed not ready for publication, since the single-numbers approach did not yield deep enough insights about the what the numbers themselves represent. Instead, they used the data as a potential avenue for the more qualitative reporting.

City Bike Share: From the available projects, the City Bike Share remains the only story when Ray tested for and published trend data. In this project, Ray's analysis used aggregates created from a database of what Ray referred to as "5-minute granular data for 2 years" about the availability of bikes at every station.

Degree

Code Definition: Data tested initially tested for more compelling issues: correlations or possible predictive qualities and disparities. If not found, used in different 'degree' in relationship to the developing story.

Recidivism, State Toxic Sites, and City Restoration: For each of these projects, Ray encountered more pronounced moments where he needed to hedge the original claim or question that he and/or the reporter had formulated. During a semi-structured interview, I asked Ray to tell me about his experiences working with large data sets. At one point, Ray discussed how reporters have initial hunches or claims they wish to qualify. Often, his work with them refines the degree of the story and its impact:

> If they [the reporters] have been digging into it awhile, then they're like 'I really want to do this thing' and for some reason the analysis is wrong or its

> a little inaccurate or the data is incomplete, usually we'll produce something. Which may not be as important or glamorous as the initial idea.

He went on to talk about how he the reporters usually find some use for the data, even if it is not as "glamorous" as the original story idea. For the Recidivism data, Ray explained how he needed to explain to the reporter how the integrity of the data made it difficult to test statistically. He said that "At one point, I told her that we can do data visualizations on this data, but it's just not accurate. We shouldn't. We can think about this in a whole, we can still use these numbers in ways, but we don't want to really focus on them." In this case, the Recidivism data did not lend to data visualizations, but it instead helped support an important leading point to the reporting, as well as provide information for the reporter to conduct lead generation.

For the State Toxic Sites and City Restoration projects, the degree of the stories connects back to the trends versus single-numbers code. Recall how Ray notes how trends typically make for a more insightful story with more news impact. During these projects, Ray initially tested for trends, but then hedged the stories by focusing on the single-number percentages.

Overall, in these 3 projects, Ray's coding work played a role in hedging the degree of the narrative in development as mediated by the data.

City Bike Share: For this project, Ray and his colleagues needed to develop a metric to render a means to assess some degree of issues surrounding bike access in certain parts of the city. In essence, degree represents another dimension of Ray's understanding of an aggregate's potential story, which can be explored and tested in relationship to Ray and/or his colleagues' perceived perspective of the data set(s). In this case,

> **Ray**: So this is a story about places where people can't find bikes and people can't park a bike where they want to go. One of the reporters was like, 'I never take City Bike anymore, because there's never an open dock.'
>
> . . .

> This one's interesting, because we don't have a metric to measure it; we don't have a standard way to measure that 'Hey, this dock sucks.' I think our metric was 2 or fewer open docks [spaces/bikes]. ...

Here, Ray casually mentions how a simple metric was devised: 2 or fewer open spaces or bikes on a dock. Such a metric led to a more systematic analysis of geographic trends and charts that showed more interesting and engaging insights about particular docks during peak commuting times. Degree, then, was initially delimited by a simple metric rooted in commonly shared experiences with the city bikes, wherein the hotspots of the story become those in which the impact was of a higher magnitude.

Ray's coding facilitated the development of these degrees and in tandem the potential stories derived from the data.

Angles

Code Definition: Ray's sense that data can be interpreted and analyzed from different contextual 'angles'.

Train Times, Health Texting, and City Bikes: While discussing each of these projects, Ray mentioned how the story typically runs along an *angle*, which he implicitly suggests shapes the way the data is examined and interpreted. For example, when discussing the Train Times project, Ray said that "I think [the reporter's] angle is that due to damage from [natural disaster], trains have been running slower." Such an "angle" informed how Ray consulted the reporter about how their recently acquired transit data and its aggregate perspective does not align with the data team's transit-feed data. In this case, such an incongruence with the data sets is largely contingent on the reporter's angle.

During my first observational interview with Ray about the Health Texting project, he noted how "I think what they're trying to do is be able to talk about the idea from a health standpoint: about taking in so much information. They probably have other angles of this story, and this is just one side of it." Here, Ray shared what he thinks is the team's overarching objectives, health, in relationship to the broader topic of the project: information consumption.[40]

During my second semi-structured interview with Ray, Ray walked through the City Bike project with me. At one point, he also used "angle" as a means to express potential alternative ways to approach the story and the data:

> **Ray**: Another angle of the story: City Bike tries to move these bikes around, so this stuff doesn't happen. For this, we could've done more with this map [referencing interactive map on his screen]. We could have added more data. I think a map was always going to happen, given that this is how people think about where the docks are and something like that. It could be done in different ways for sure . . .

In hindsight, Ray suggests that he and the team could have made a more interactive map based on a different angle: how the City Bike company attempts to shuttle and re-distribute bikes to different docks in need.

Even though Ray only mentioned this code explicitly three times during 3 different interviews, this concept represents another important rhetorical and communicative goal indicative of the context of Ray's coding work; namely, how Ray's coding is implicated in the purpose-driven and goal-directed nature of reporting work.

Indeed, *angle* highlights the link between the developing story and the data, which helps the story take shape. When Ray discussed other projects, such as the State Toxic Sites, City Restoration, and Recidivism, he described situations where the reporters and their angles helped him understand how he could support the project through his coding work.

In the case of the City Payroll project, where Ray was the initial producer of the project, he seemed to be trying to define angles for a story through his analysis coding

[40] Such an angle certainly shaped and was shaped by the process of the team to write a texting campaign script that was being revised up until the initial launch date.

work. In the following 2 sections, I report findings that indicate how Ray seems to follow, develop, and/or refine a story angle through his coding work.

Slicing

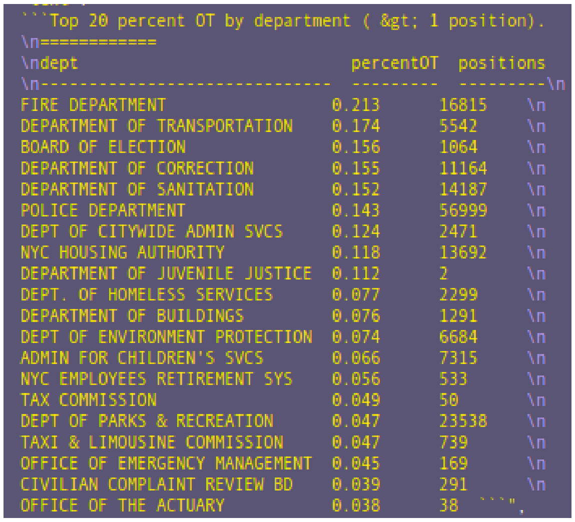
Code Definition: The act to dimensionalize and/or reduce the original data into a goal-directed set(s) for a particular analysis.

City Payroll: During this project, Vince referred to Ray's preliminary analysis work as "slicing," while also noting that Ray's slicing approach to the data was appropriate and the right direction to develop the story further. Such slices are evinced by Ray's coding work to analyze the City Payroll data that he initially queried from the open portal online. For example, Ray shares some of the following tables produced from the sliced data according to a potential angle that explored overtime (OT) across the data:

- "Top 20 percent OT by title (> 1 position)"
- "Top 20 percent OT by department (> 1 position)"
- "Total percents [of OT]"
- "Years (> 1) and percent positions with overtime"
- Summary of "Board of Elections" department positions

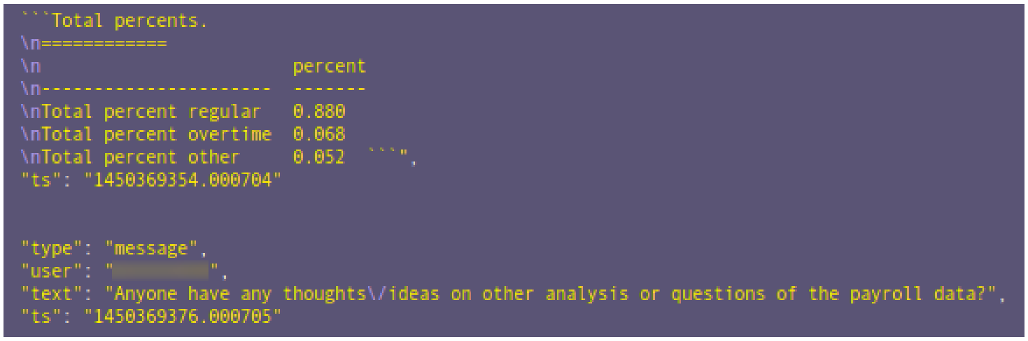
Figures 3.3 and 3.4[41] show the 2 initial tables of computed values that Ray created from particular slices of data, which stem from the germ of an idea to engage OT. Indeed, in these 2 tables, Ray presents top-level values of OT to his colleagues on Slack and he solicits feedback, asking them "Anyone have any thoughts/ideas on other analysis or questions of the payroll data?"

[41] This image is an excerpt of a JSON file from Ray's Slack logs on the team's general channel. I used the archive of his Slack logs, so I could capture data from project days and times that I could not observe.



~~~
```Top 20 percent OT by department ( &gt; 1 position).
\n===========
\ndept                             percentOT  positions
\n------------------------------  ---------  ---------\n
FIRE DEPARTMENT                 0.213      16815    \n
DEPARTMENT OF TRANSPORTATION    0.174      5542     \n
BOARD OF ELECTION               0.156      1064     \n
DEPARTMENT OF CORRECTION        0.155      11164    \n
DEPARTMENT OF SANITATION        0.152      14187    \n
POLICE DEPARTMENT               0.143      56999    \n
DEPT OF CITYWIDE ADMIN SVCS     0.124      2471     \n
NYC HOUSING AUTHORITY           0.118      13692    \n
DEPARTMENT OF JUVENILE JUSTICE  0.112      2        \n
DEPT. OF HOMELESS SERVICES      0.077      2299     \n
DEPARTMENT OF BUILDINGS         0.076      1291     \n
DEPT OF ENVIRONMENT PROTECTION  0.074      6684     \n
ADMIN FOR CHILDREN'S SVCS       0.066      7315     \n
NYC EMPLOYEES RETIREMENT SYS    0.056      533      \n
TAX COMMISSION                  0.049      50       \n
DEPT OF PARKS & RECREATION      0.047      23538    \n
TAXI & LIMOUSINE COMMISSION     0.047      739      \n
OFFICE OF EMERGENCY MANAGEMENT  0.045      169      \n
CIVILIAN COMPLAINT REVIEW BD    0.039      291      \n
OFFICE OF THE ACTUARY           0.038      38  ```",
~~~

Figure 3.3: Screenshot of one of Ray's initial products of "slicing" the original City Payroll data. Note that this representation from the Slack archived logs as I read them in my code editor.



~~~
```Total percents.
\n===========
\n                         percent
\n----------------------  -------
\nTotal percent regular   0.880
\nTotal percent overtime  0.068
\nTotal percent other     0.052  ```",
"ts": "1450369354.000704"


"type": "message",
"user": "[redacted]",
"text": "Anyone have any thoughts\/ideas on other analysis or questions of the payroll data?",
"ts": "1450369376.000705"
~~~

Figure 3.4: Screenshot of Ray's initial sharing of a "slice" on Slack, requesting for feedback from his team. Note that this representation from the Slack archived logs as I read them in my code editor.

After requesting feedback, Ray received responses from each member of the team at various times throughout the week. Phil was the first to respond, providing some new questions that could help refine the angle of the OT-related data:

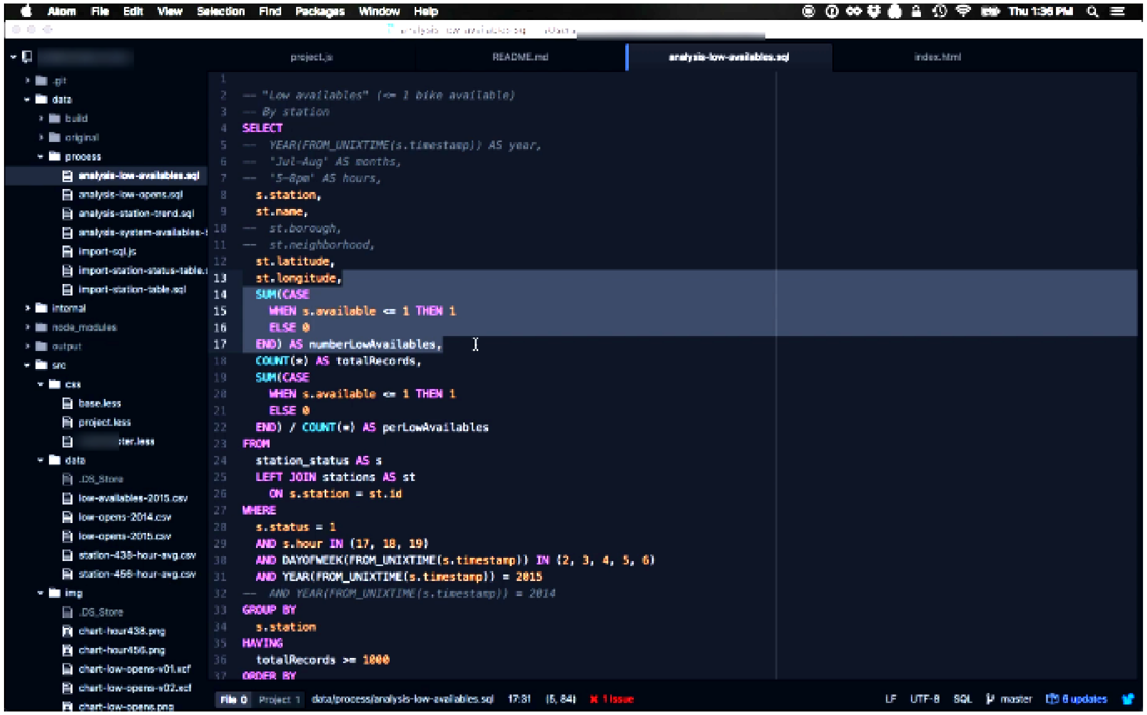
> **Phil**: I would be mostly curious about a) How does overtime relate to seniority, over the course of a career. Is overtime a thing that savvy senior

> people get, or spread more evenly within a department? I'm more curious about intradepartment variance than percentage of a department's total pay that is overtime. for example, for people who show up in the payroll data for many years in a row - do they get a lot of overtime every year? is it up and down?
>
> . . .
>
> b) Are positions that get a lot of overtime ones with more restrictive hiring requirements, e.g. exams, occupational licensing.
>
> . . .
>
> c) What positions is the city actively hiring for?
>
> **Ray**: That would be interesting. The data we have is start time at agency. I think that's an ok data point, but I am not too confident that it necessarily equates to seniority.

In their quick exchange, Ray's tables provided Phil with the means to refine the OT angle by suggesting inquiries related to potential notions of seniority. Ray finds this angle interesting, but questions the data and its capability to properly represent the concept of "seniority" with the start-time datapoint. In short, Ray's *perspective* of the data helps him interpret Phil's potential *angle*.

City Bike Share: During this project, recall how Ray noted how the complete City Bike database was far too big to understand on its own. He knew that it made more sense to reduce the 2-years worth of granular data, so he *sliced* the database by the following time constraints: between the summer months of July and August; and only during commuting hours in the morning and evening. Ray and his colleagues developed a simple *bikes per station* metric to understand potential issues about the both renting and parking *availability* during these slices. (See Fig. 3.5 for SQL code and Ray's description of it below.)

R: Once you get into a database and write some queries that looked at each station and ... so this is the low available bikes [highlights SQL code pictured below]. coming up with that idea that this is our metric.



```sql

-- "Low availables" (<= 1 bike available)
-- By station
SELECT
--  YEAR(FROM_UNIXTIME(s.timestamp)) AS year,
--  'Jul-Aug' AS months,
--  '5-8pm' AS hours,
  s.station,
  st.name,
--  st.borough,
--  st.neighborhood,
  st.latitude,
  st.longitude,
  SUM(CASE
    WHEN s.available <= 1 THEN 1
    ELSE 0
  END) AS numberLowAvailables,
  COUNT(*) AS totalRecords,
  SUM(CASE
    WHEN s.available <= 1 THEN 1
    ELSE 0
  END) / COUNT(*) AS perLowAvailables
FROM
  station_status AS s
  LEFT JOIN stations AS st
    ON s.station = st.id
WHERE
  s.status = 1
  AND s.hour IN (17, 18, 19)
  AND DAYOFWEEK(FROM_UNIXTIME(s.timestamp)) IN (2, 3, 4, 5, 6)
  AND YEAR(FROM_UNIXTIME(s.timestamp)) = 2015
--  AND YEAR(FROM_UNIXTIME(s.timestamp)) = 2014
GROUP BY
  s.station
HAVING
  totalRecords >= 1000
ORDER BY
```

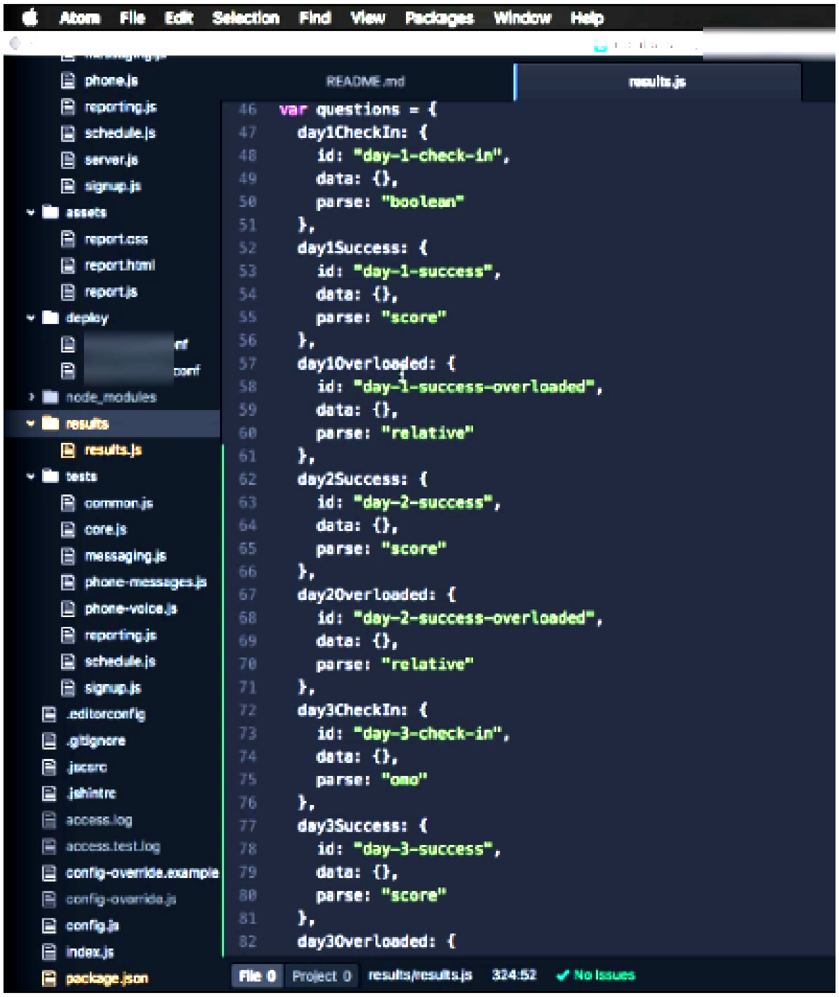
R: You can see from what's commented out that [highlights comments pictured below], that I'm pretty sure that I would have made these queries and then sent some of the results over to [reporter] and be like, "Hey, this is some of the data that I'm findings," before making a map.

Figure 3.5: Excerpt from transcription of semi-structured interview with Ray, discussing how he queried the database with SQL code to slice it.

In sum, this new perspective of the data enabled Ray and the team to refine an angle for an aggregate story about bike availability for commuters. Ray recalled how the SQL code represents and created the "low-availables" metric, wherein he produced new aggregate perspectives – *by neighborhood*, *by borough*, *by summer months*, and *by peak commuting times* – and combinations of them. His epistemic process aligns with the City Payroll slicing work, too, since he also notes how, "I'm pretty sure that I would have made these queries and then sent some of the results over to [the reporter] an be like, 'Hey, this is some of the data that I'm finding' before making the map."

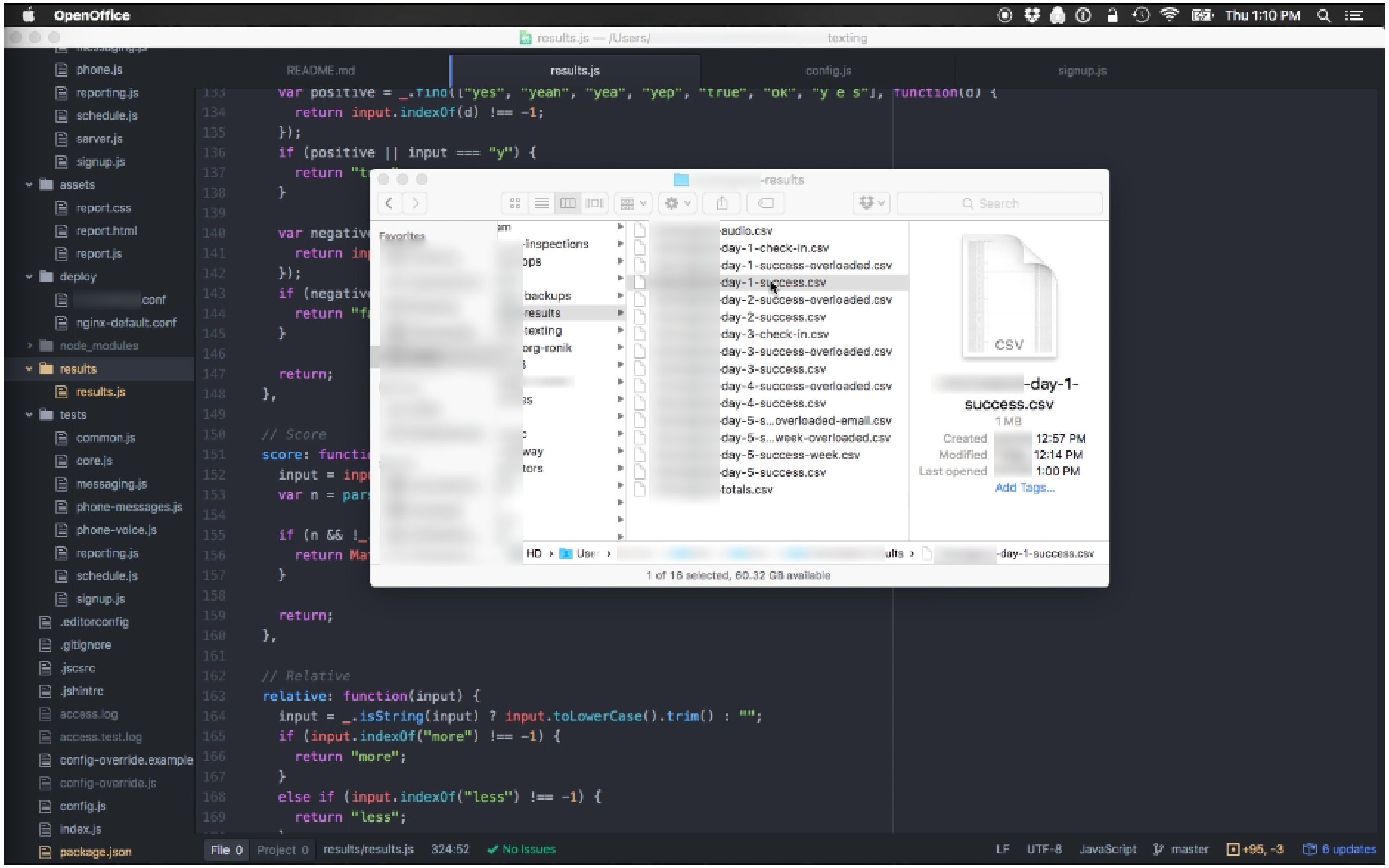
Health Texting: By the time that this project began, I had started to note a recurrent task emerging in Ray's coding work to slice sets of data. In an observational interview, Ray noted how he had been writing a new file, `results.js`, which he used to process the exported data from the production server. In reference to the code in Figure 3.6, he said that the code is "basically trying to go through each question that we have and then parsing out that information."

Ray's coding of the pictured multi-dimensional object array, `questions` (line 46), sliced the *per Person* document model from the database into *per Prompt* perspectives with a place to assign the day's data as a nested object array, `data: {}`. This data structure set up a place for him to write JS functions later in the file to code outputs for each of these day-based data into a collection of CSV files (see Fig. 3.7 below).



```
var questions = {
  day1CheckIn: {
    id: "day-1-check-in",
    data: {},
    parse: "boolean"
  },
  day1Success: {
    id: "day-1-success",
    data: {},
    parse: "score"
  },
  day1Overloaded: {
    id: "day-1-success-overloaded",
    data: {},
    parse: "relative"
  },
  day2Success: {
    id: "day-2-success",
    data: {},
    parse: "score"
  },
  day2Overloaded: {
    id: "day-2-success-overloaded",
    data: {},
    parse: "relative"
  },
  day3CheckIn: {
    id: "day-3-check-in",
    data: {},
    parse: "omo"
  },
  day3Success: {
    id: "day-3-success",
    data: {},
    parse: "score"
  },
  day3Overloaded: {
```

Figure 3.6: Screenshot of Ray's initial coding work to slice the *per Person* document model from the database into a *per Day* perspective with a place to assign the day's data as an object array: data: {}.



```
    var positive = _.find(["yes", "yeah", "yea", "yep", "true", "ok", "y e s"], function(d) {
      return input.indexOf(d) !== -1;
    });
    if (positive || input === "y") {
      return "t

    }

    var negativ
      return in
    });
    if (negativ
      return "f
    }

    return;
  },

  // Score
  score: functi
    input = inp
    var n = par

    if (n && !_
      return Ma
    }

    return;
  },

  // Relative
  relative: function(input) {
    input = _.isString(input) ? input.toLowerCase().trim() : "";
    if (input.indexOf("more") !== -1) {
      return "more";
    }
    else if (input.indexOf("less") !== -1) {
      return "less";
```

```
-audio.csv
-day-1-check-in.csv
-day-1-success-overloaded.csv
-day-1-success.csv
-day-2-success-overloaded.csv
-day-2-success.csv
-day-3-check-in.csv
-day-3-success-overloaded.csv
-day-3-success.csv
-day-4-success-overloaded.csv
-day-4-success.csv
-day-5-s...overloaded-email.csv
-day-5-s...week-overloaded.csv
-day-5-success-week.csv
-day-5-success.csv
-totals.csv
```

-day-1-success.csv
1 MB
Created 12:57 PM
Modified 12:14 PM
Last opened 1:00 PM

1 of 16 selected, 60.32 GB available

Figure 3.7: Screenshot of the product of Ray's slicing work, outputting a series of CSV files for potential analysis.

Ray created these *per Prompt* slices in preparation for a meeting with the podcasting team, which occurred 15 minutes after this coding session. He had yet to know exactly what the team wanted, but he knew that he would need to aggregate the *per Person* instance, i.e., perspective, of the database. Such a decision to code the data into new perspectives suggests that Ray had a sense about what potential angles aligned with particular kinds of aggregate perspectives afforded by the newly structured information seen in the figures above. During the TAP, Ray noted why he structured the data in this way:

> **Ray**: So the main reason we will be talking to the [podcasting team name redacted] people is to get those high-level questions that they wanted answered: 'Why did they [the participants] do this?' essentially – outside of general information and marketing, so that we can put those data analysis questions together. But [this code also gets] some more general type of things that we want to do.

Despite not having an initial meeting with the team, Ray had a sense about what general *angles* that the team might desire to pursue.

In sum, Ray's slicing work hinged on developing angles of a story as it was linked to the data set(s). Ray's coding activity played an integral role in testing and refining these angles through the process of writing, i.e., organizing values in such a way to produce, new perspectives made possible by the data sets. In Table 3.5, each project reported in this chapter are organized with their data sets, their original perspectives, the main angle derived from the data processing and analysis, as well as the project's output perspective.

Table 3.5: List of projects, their original data sets, perspective of original sets, overarching story angle, and output perspective.

Project	Original Data Sets	Original Perspective	Angle	Output Perspective
State Toxic Sites	DEP Site Status Data (7 files) and U.S. Census Tracts	DEP: *per Site* (location and status); Census: Average demographics per neighborhood within 1 mile radius of site	How many people live within a 1 mile radius of a particular toxic site? What toxic sites are near places that you dwell?	3 slices: *Average Census tract demographics within 1 mile radius of site*; *per Site* geocoded location data; and *per Site* toxic status info

Continued on next page

Table 3.5 – continued from previous page

Project	Original Data Sets	Original Perspective	Angle	Output Perspective
City Restoration Project	WWWC-charts.pdf file of computed values provided by reporter	*per Neighborhood* with Active Applicants, Construction Starts, Construction Complete, % Homeowners who received construction of reimbursement	What neighborhoods have benefited more or less from the state's restoration project after [natural disaster redacted]?	Slice: Same as original provided, but manually converted to CSV file format
City Payroll	Data retrieved from a database query from an online "Open Data Portal"	*per Employee* with job title, department, type of pay, total pay, etc.	What relationships might be of interest to readers from the city?	No final output, but produced: Top 20 OT, Top 20 Pay, Top 20 Pay per Department, etc.

Continued on next page

Table 3.5 – continued from previous page

Project	Original Data Sets	Original Perspective	Angle	Output Perspective
Health Texting	Data collected by the app that Ray developed for the podcasting team	Database of *per Person* with their questions and responses	How do subscribers adhere (or not) to the texting campaign goals?	No final output, but produced: *per Prompt*, Total subscribed per day, Total unsubscribed per day, etc.
Train-Time Perfor-mance	OTP data requested from the city by reporter; Team train headway database	OTP: *per Monthly % of train line OTP* (on-time performance) and yearly totals; Headway: *per Train* with every stop, every 5-minutes	Did the [natural disaster redacted] affect train performance around a particular region?	No slices: Incompatible perspectives.

Continued on next page

Table 3.5 – continued from previous page

Project	Original Data Sets	Original Perspective	Angle	Output Perspective
City Bike Rentals	Data collected by the team through a city-provided API (Application Programming Interface)	*Per station* with per 5-minute timestamp, location, dock, and bike data	Where and when are rental bikes available or not to rent across the city during commuting hours between July and August months?	5 slices: 3 with *% of Stations with 2 or fewer open docks across summer months during peak commuting times for last 2 years*; 2 with *Average bike availability at particular station*

Synthesis of analytic codes

Ray's coding acts to *slice* data sets incorporates the other analytic codes. (See Fig. 3.8 below.) His *slicing* creates new and different *perspectives* of the data. Slices differ from perspectives, since perspectives are conceptual products of a person's interpretation and understanding of the data slice, which is a textual product and mediational means. This relationship between a person's perspective of the data and its slices can affect or be affected by their tacit or explicit *angle* in finding stories. Put differently, slices have the potential to mediate the decisions and alterations made to angles, which may help Ray and the team develop new angles, shed unfounded or uninteresting ideas, or refine the claims derived from their ongoing interrogation of the data.

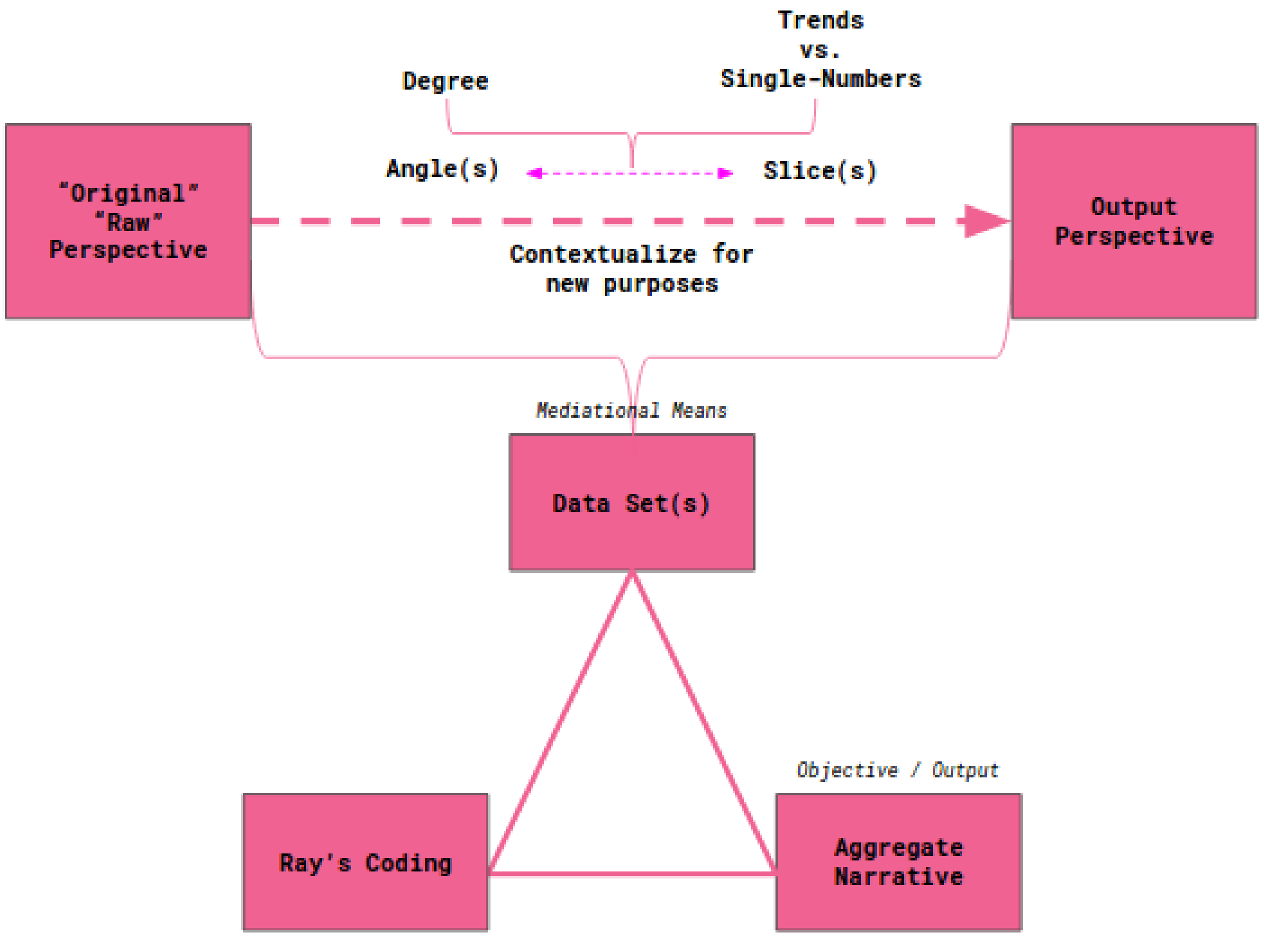


Figure 3.8: Tentative organization of category and codes surrounding the objective: aggregate narrative.

The above representation of Ray's coding activity, as mediated by data sets, is informed by the Vygotiskian epistemology that grounds many of the studies of the materialities of writing (Haas, 1996; Scribner & Cole, 1981; Witte, 1992). It signifies the importance of sets of data in Ray's workplace, and the figure represents how Ray's coding activity to find stories in the data involves particular strategies to contextualize the original data into the project objectives at-hand.

3.4 Discussion: Aggregate Narratives and Organizing Information through Coding

Aggregate narratives, or finding and developing stories with aggregate information, operates as one of the major objectives of Ray's team. In the prior sections, I have shown how I developed the concept of aggregate narratives from the collected data, and how the concept links Ray's work with aggregate information and the broader objective to find stories through the creation and use of various perspectives of data sets. Ray's coding work to process and analyze data sets begins to shed light on the malleable nature of such texts, wherein Ray and others see how aggregates can provide different kinds of stories through the creation and development of an angle, which have varying degrees of explanatory power (trends versus single-numbers) about social phenomena.

Ray's coding to slice data and create new perspectives seems to be shared amongst others who work with data sets. Data journalist Groskopf (2016) put together "The Quartz guide to bad data," which catalogs myriad, idiosyncratic issues that data journalists will experience and should note. In adherence with the concept of perspectives, Groskopf labels 2 sections of the field guide as *Data are too coarse* and *Data are too granular*. For instance, at the beginning of the "too coarse" section, Groskopf describes issues linked to desired perspectives afforded by aggregate data: "You've got states and you need counties. You've got employers and you need employees. They gave you years, but you want months. In many cases we get data that have been aggregated too much for our purposes.” Overall, sets of data offer perspectives for their users, and similar to Ray and his purposes for data, Groskopf also makes this link between the perspective of data to situated needs.

Ray and others on the team have developed an ability to read, make sense of, and invent ways to manipulate, combine, and create data to output it as a particular file for either further analysis, visualization, or other goals to help refine and tell the story. Indeed, findings start to show how data sets and Ray’s coding with such texts play a part in a dynamic process to develop aggregate narratives: that coding helps quickly slice data, which are then interpreted by many eyes, all of which alter the goals and angles of the story and thereby creating more slicing work. As a result of this coding activity to slice the data, new questions emerged from this analysis of Ray’s coding context: specifically, how does coding play a role in Ray’s epistemic work to slice data sets?

This recurrent context of coding activity seems to help Ray (and others) develop a rhetorical and interpretative sense about what and how to do with the data. Indeed, Ray’s coding activity included slicing the data and sharing those slices with colleagues. Based on these observations, it seemed that these slices facilitated epistemic work, wherein ray organized semiotic resources that represented some facet of reality. His slicing work demarcates moments when he needs to change, dimensionalize, or translate the information, which may have been collected and used for a variety of other purposes. Such findings gesture toward how Ray’s coding plays a role in shaping the representation of some reality (toxic site clean-up activities, city payrolls, or people attempting to consume less information), as well as gesturing toward how his coding mediates the objective to find stories in the data.

In the next chapter, I examine Ray’s *in situ* coding during processing and analysis tasks to explore the materialities of Ray’s coding to process data sets. Such data collection and analysis follows from the following specifying questions:

- What are the recurrent tools, semiotic modes and resources of the data sets?
- How do the tools and semiotic resources associated with data sets mediate Ray's coding of slices?

Chapter 4

Findings: Provisional texts

Structured data exists when information is clearly broken out into fields that have an explicit meaning and are highly categorical, ordinal or numeric.

–Booz, Allen, and Hamilton, 2015, p. 55

To study writing is, over and above all else, to study acts of making meaning that are mediated through 'texts.'

–Stephen P. Witte, 1992, p. 237

4.1 Introduction: What's in a List?

The quotes above exemplify a tension between data and meaning that I examine through Ray's coding to slice data. Data science practitioners Herman et al. (2015) wrote a field guide to working with data, and they describe meaning as something that can be rendered explicit and clear in and through the structuring of data sets. Witte (1992) charged writing researchers to recognize the active process to make sense of texts by studying the human endeavor to draw upon the organization and interpretation of semiotic resources. In this chapter, I fold data sets, and digital data more broadly, into the purview of studies of writing and its role in meaning-making.

I do so through Ray's coding with data; namely, his production of a relatively invisible type of text that I call *provisional texts*. *Provisional texts* represents anytime Ray codes, or slices data, to create distilled combinations of datapoints with the goal to refine an angle of the developing story. These texts are analogous to Geisler's (2001) notion of invisible texts and Witte's (1992) memorial texts (pp. 265-7). Both Geisler and Witte's studies of workplace writers show how they produce mundane and minor texts, which often are disregarded, despite their work to fulfill objectives and create finished products. Ray's coding work included writing source files, which would change or be repurposed in response to the team's rapidly developing ideas. Ray's coding acts to create the provisional texts included code and coding that was sometimes disregarded and left undocumented. Over time, I began to see how Ray's coding became the invisible part of the process to create particular slices, which provided himself and the team important knowledge to carry the story forward.

I name these texts, provisional, for 2 main reasons. Firstly, provisional reflects the malleable and mutable character of the information that people construct with data. The stories that Ray plays a role in developing with data are not self-evident, rather every coding act to slice the data represents new angles that people verified, explored, hedged, or ignored. Secondly, provisional links Ray's coding to the ways writing has always been an important and complex provision. Historian Schmandt-Besseret (1990) argues that the humble origins of writing point toward the material tokenization of land and livestock—a provision to account for one's property. Along the same very lines, Ray's work with sets of data involves the systematic, goal-oriented tokenization of some social-material phenomena too through his creation of of these complex lists.

Indeed, writing provides people the capability to render language in numerous forms to create richly different experiences. Writing and its materialities support and alter human activity, understanding, and the human capacity to reflect, analyze, and respond. It looks different and is produced differently based on where one reads and writes, what language(s) and technologies are used to create it, and what purposes people bring to the act to write or read texts. Literacy scholar Goody (1977/2001) historicizes the societal impact of lists and list-making on developing economies and polities. He argues that the material-semiotic nature of lists reconfigured human relationships with language by spatializing physical objects, people, and social activities into more elaborate culturally-informed, value-based orders. According to Goody, these pre-alphabetic, list-making practices defined a "'semantic field'" (p. 47) for their producers to make decisions; that is, a process of inclusion and exclusion driven by the desired ends of the elite for whom such lists are made. In our current time, networked and public-facing databases are becoming much more common. Digital server farms host lists that societies have only just begun to utilize at new scales, for new purposes, and new audiences. This case of Ray in a news room represents one such domain of activity taking up this new resource to tell data-driven stories.

In the previous chapter, I reported findings derived from an analysis of Ray's context, which included the recurrent objective to find stories with aggregate information. The core category, aggregate narratives, serves as an objective that Ray's coding often helps fulfill. In this chapter, I examine Ray's *in situ* coding activity to produce slices of data – complex translations of lists – during processing and analysis tasks, which provided integral information to produce aggregate narratives. Within the scope of understanding Ray's coding as writing, I examined Ray's decisions to represent information and make sense of it in relationship to the following questions:

- What are the recurrent tools, semiotic modes and resources of the data sets?
- How do the tools and semiotic resources associated with data sets mediate Ray's coding of provisional texts?

Through these questions, I investigate the properties of Ray's use of the JavaScript programming language as a means to use the computational figuring of digital data to compose provisional texts. These texts create Goody's goal-oriented, semantic fields by which Ray and his colleagues refine the angles that shape their stories. These questions, which I use as a means to examine how such meaningful lists are made, are also rooted in mediation theories discussed in section 2.2.1 (Vygotsky, 1934/2012; Scribner & Cole, 1981). Another branch of this epistemological lineage, distributed cognition, helps me examine Ray's coding of provisional texts more closely. Cognitive scientist Hutchins (1995) might describe Ray's provisional texts as epistemic objects, or things to think with. Like Haas (1996), Hutchins argues that technologies configure and reconfigure social relationships – interactions between people during particular activities – as much as their conceptual development. Hutchins does so through a cognitive-informed, ethnographic study of ship navigation teams. Through his study, he shows how such teams, and humans more generally, have developed the means to render utterances across a dynamic range of representational media, which have consequences contingent on the people conducting such interpretive practices. Overall, Hutchins' theory of cognition as distributed across people, tools, and language helps writing researchers study how people share information and develop conceptually-shared knowledge.

For instance, Hutchins argues that any meaning derived from any message is contingent on the modalities of the expression being uttered. He conveys this point through a case of a team member reporting a bearing to their destination of Hotel del Coronado as 003°:

> The bearing recorder simply recorded and relayed the reported bearing as a string of digits, but the plotter, without plotting it, responded: 'It better not be. If it is we're pulling into Tijuana right away!' In an interview, this same plotter, a quartermaster chief, once described the ability to feel bearings as directions in the local space defined by bodily orientation as being able to 'think like a compass' . . . This bearing meant something more to the plotter than it did to the bearing recorder because the plotter brought it into coordination with a structured representational medium . . . (p. 140)

Hutchins' example conveys how the simple utterance of a ship's bearing has dynamic materialities as derived from its hearer in relationship to their contexts of use. Even this sharing of simple inscription points to how signs are linked to cultural knowledge and the activity to wield technologies that create durable and recurrent situations of use for people to share common knowledge. Indeed, the more experienced plotter could "think like a compass," while the recorder simply passed the media along the chain of activity.

Technologies configure and reconfigure social relationships, interactions between people, as much as the knowledge the construct and share. Such is the same for writing. Pigg (2014b) observed participants learn *composing habits* with mobile technologies, which involve using the resources of such devices within their context of particular recurrent public spaces to complete their writing objectives. She concluded that "these places and materials promote a stability – or set of rhythms – that can be observed in routine movements both on-screen and in physical space" (p. 267). Pigg's research focuses on how students develop rhythms between space and tools as important factors for the writers observed to develop practices as habits. In this chapter, I report preliminary findings from an initial inquiry into how Ray developed a particular coding environment conducive to creating provisional texts, sharing such information across media and their modalities. I discuss the ways his coding of provisional texts integrated the use of language as coordinated across different writing technologies and people, and how his data processing and analysis was enacted by a particular habit of mind linked to the modalities of the data sets.

In what follows, I survey my method, report findings from an analysis of the data, and discuss how such findings begin to connect back to studies of the materialities of writing.

4.2 Method

After recognizing Ray's recurrent task to code slices of data sets, I conducted think-aloud protocols to gain some perspective about Ray's perception about his thinking during such coding tasks. I also selected data from past projects that would help refine insights about such coding work to dimensionalize data sets. In Table 4.1, I list out the reduced and selected data on which my findings about provisional texts are made.

Table 4.1: Data used to construct the core category of *provisional text*.

Core Category	**Data Types**	**Data Projects**
Provisional Texts	1/5 Think-Aloud Protocols 9/32 observational sessions ~8 hours of screen recordings ~100 pages of fieldnotes/memos 6 Observational interviews 3 Semi-structured interviews 2 Retrospective accounts Selected project code Selected project correspondence	State Toxic Sites City Payroll Health Texting

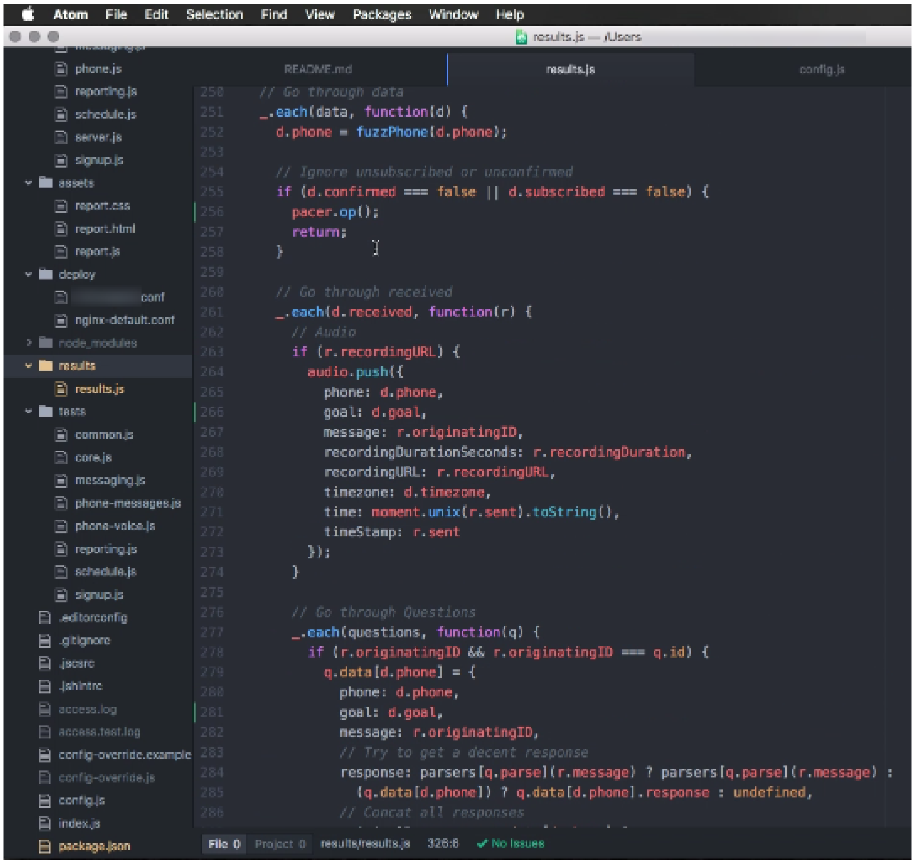
The TAP was conducted during the Health Texting project. Specifically, Ray had completed developing and testing the texting application that sent questions and challenges to subscribers about their attempts to consume less information for 1 week. I started to conduct the TAPs during this post-production phase, wherein the podcasting team wanted to learn more about subscribers' experiences. For more information about the TAPs and my general conduct, see Section 2.3.3. I reduced the number of TAPs to this singular protocol for 2 main reasons: At the time, I needed to generate preliminary findings to report for this dissertation, which would have been difficult due to the second main reason of time constraints. Accordingly, *the findings from the analysis in* *this chapter remain more preliminary in the grander scheme of the theorizing of this case-study.*

I also conducted another analytic pass through the other data listed in Table 4.1, which I selected from previous projects: State Toxic Sites and City Payroll. Specifically, the 2 retrospective accounts focus on the State Toxic Sites. All other types of data were used across all projects reported in this chapter: transcribed screen-recordings of Ray's laptop, Slack communication logs, selected code, and observational interviews.

4.3 Analysis and Findings

In this domain of coding work, data sets implicate Ray's coding in multiple forms of knowledge work in relationship to the digital and computational technical work of reading and writing code. His coding labor is linked to large sets of structured data, which he must process and analyze within the broader objective to find and create aggregate narratives. His coding practice includes producing multiple slices of the original data – *provisional texts* – which became the second core category of this embedded unit of analysis. In this chapter, I report findings drawn from Ray's coding practice to produce provisional texts.

Before I report particular findings generated from my grounded analysis, here is a representative coding situation, wherein Ray uses more explicit language to describe what he hopes to achieve through his slicing of the data. In the middle of the Health Texting project, Ray had already developed and released an interactive mobile texting campaign for a podcasting team. The theme of the campaign was information consumption in peoples' everyday lives, so the mobile texting application sent scripted prompts to subscribers – listeners of the podcast – about 3 times every day, asking them to reflect about their efforts to consume less information. The application saved all of the subscriber responses per person to a networked Amazon database. During the following moment, Ray had set up a new file, `results.js` (see Fig. 4.1 below), so he could create “aggregates” of the entire database and output them as CSV (Comma-Separated Values) files for analysis work. Ray’s code below serves as a way to dimensionalize the data by slicing it into different sets of relationships so they can then be organized and outputted as CSV files for further investigation.



```
250    // Go through data
251    _.each(data, function(d) {
252      d.phone = fuzzPhone(d.phone);
253
254      // Ignore unsubscribed or unconfirmed
255      if (d.confirmed === false || d.subscribed === false) {
256        pacer.op();
257        return;
258      }
259
260      // Go through received
261      _.each(d.received, function(r) {
262        // Audio
263        if (r.recordingURL) {
264          audio.push({
265            phone: d.phone,
266            goal: d.goal,
267            message: r.originatingID,
268            recordingDurationSeconds: r.recordingDuration,
269            recordingURL: r.recordingURL,
270            timezone: d.timezone,
271            time: moment.unix(r.sent).toString(),
272            timeStamp: r.sent
273          });
274        }
275
276        // Go through Questions
277        _.each(questions, function(q) {
278          if (r.originatingID && r.originatingID === q.id) {
279            q.data[d.phone] = {
280              phone: d.phone,
281              goal: d.goal,
282              message: r.originatingID,
283              // Try to get a decent response
284              response: parsers[q.parse](r.message) ? parsers[q.parse](r.message) :
285                (q.data[d.phone]) ? q.data[d.phone].response : undefined,
286              // Concat all responses
```

Figure 4.1: Ray’s code that slices up the data into different ‘aggregates’ during the Health Texting project.

During an observational interview, Ray provided a quick tour of his code, which “creates,” or writes, new aggregate perspectives based on each subscriber’s response:

> <*Ray Scrolls up the* `results.js` *and hovers his mouse over the* `// Go through data` *code block* (line 250).>

> **Ray**: So at this point [in the code], we've gone through the data [ref. `// Go through received` code block] and we've put it [the data] into each question [ref. `// Go through Questions` code block]. Essentially people can give us multiple responses for questions [ref. `response: parsers[q.parse]` and `originalResponses: q.data[d.phone]` code], but this is creating a row for each column number for that question.

These coding acts to slice data sets fall under the broader objective to create aggregate narratives through new insights about the different perspectives and angles Ray's coding helps define and refine. In what follows, I report findings about Ray's coding and use of such slices, which I call *provisional texts*.

4.3.1 Coding provisional texts: Semiotic resources and habits of mind

In this section, I report findings related to Ray's coding practice to create provisional texts; that is, slices of the data sets, which help the team find and refine their narrative surrounding the aggregate information. Firstly, I respond to the first specifying question: *What are the recurrent tools, semiotic modes and resources of the data sets?* I do so by providing a broader view of Ray's coding of provisional texts by showing the recurrent purposes and tools that make-up this coding work, as well as the semiotic modes of the data sets. Such a view of this coding activity will provide the definitional language necessary to respond to the second question: *How do such modalities mediate Ray's coding to process and analyze data sets?*

Tools, texts, and semiotic modes: Ray's writing environment

While observing Ray coding, he orchestrated a variety of tools with ease. With laptop at his dining room table, Ray nimbly traversed and used numerous tools on his screen. He used a variety of shortcuts and hotkeys to fluidly navigate between his Atom code editor, his terminal for the code's output, OpenOffice Calc to read CSV files, and the Chrome web browser to quickly search for necessary information to help him complete a coding task. Over time, I observed this pattern of tools, which enabled Ray to create a coding environment, wherein he could quickly make sense of the code and texts that he was producing, as well as making sure that his code was creating the desired outputs.

One way to capture a sense about Ray's writing environment is to see how he used these technologies together to create an environment that supported his reading, writing, and sharing of provisional texts. In Table 4.2, I tabulated Ray's recurrent tasks and tools to process and analyze data. By cataloging these tasks and tools, I was able to start identifying patterns and situations in which these tasks and tools occurred.

Table 4.2: Reading and coding data sets: Recurrent tasks and tools.

Activity	**Task**	**Tools**
Reading: Actively engaging a data set's semiotic modes during particular tasks	To find a new project	Open Office / Numbers spreadsheet, Atom code editor, Terminal console logs
	To understand datum and their relationships	Open Office / Numbers spreadsheet, Atom code editor, Plain text editor, Web-based data portal, Terminal console logs, PDF viewer

Continued on next page

Table 4.2 – continued from previous page

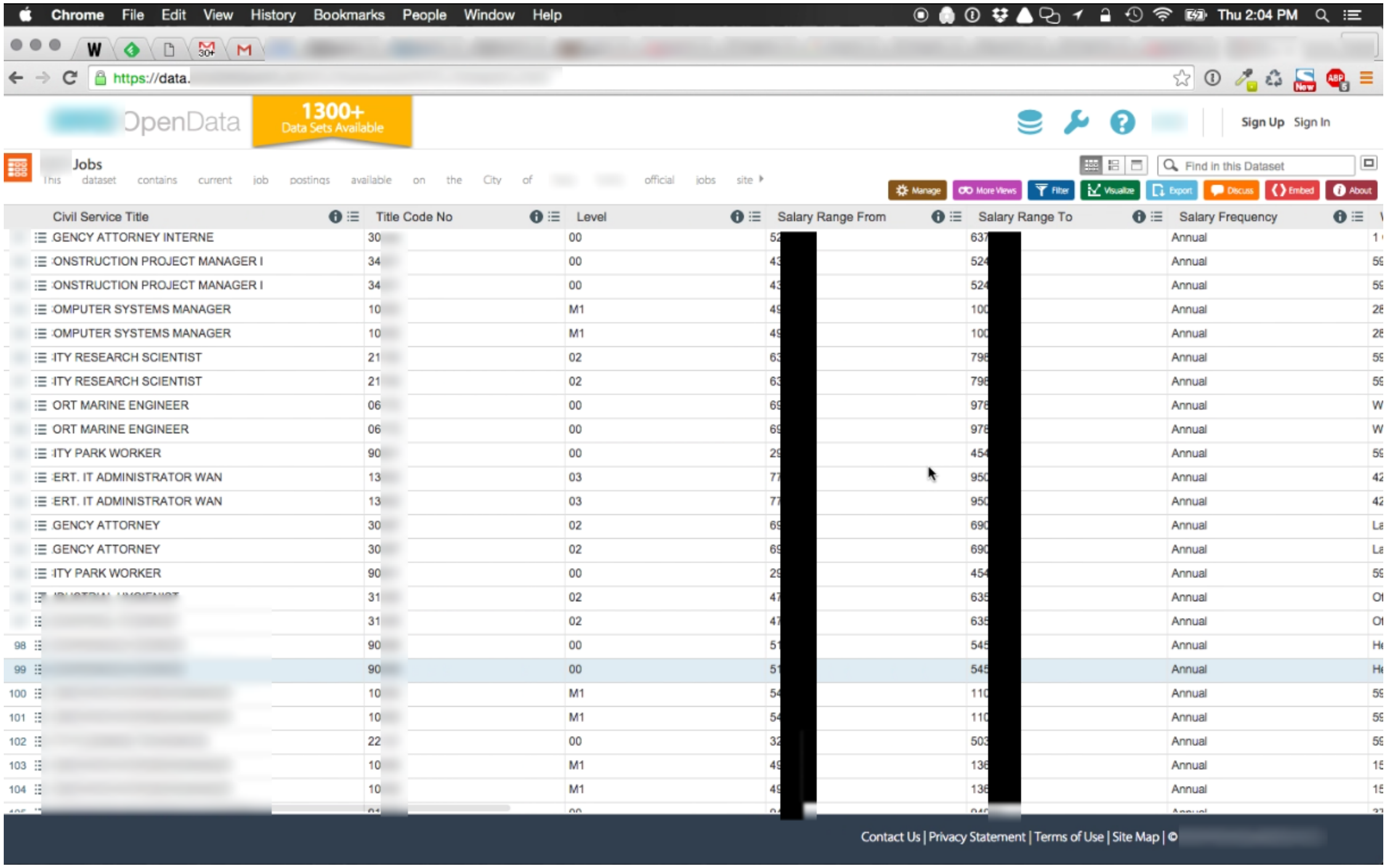
Activity	Task	Tools
	To consult team-member / assess potential story point	Open Office / Numbers spreadsheet, Atom code editor, Plain text editor, Terminal console logs, PDF viewer, E-mail, Slack
	To assess data integrity; hedge its boundaries of use	Open Office / Numbers spreadsheet, Atom code editor, Plain text editor, Terminal console logs, PDF viewer, E-mail, Slack
	To invent new combinations with datum and/or other data sets with other inscriptions	Slack, Atom code editor, Terminal console logs

Continued on next page

Table 4.2 – continued from previous page

Activity	Task	Tools
Writing/Coding: Actively writing code in relationship to data set(s) to process the information or test particular ideas about the relationship between datum and the developing story.	To combine 2 or more data sets	Atom code editor, Terminal
	To munge / clean data sets	Atom code editor, Terminal
	To analyze data sets	Atom code editor, Terminal

During much of Ray's data-work, he read and wrote outputs with console logs in his code editor, which rendered structured data in a readable format within a Terminal. Ray often coordinated this coding practice between these 2 technologies in tandem with any CSV (Comma-Separated Values) files within spreadsheet programs, such as OpenOffice Calc. If the data was in another common data format, JSON (JavaScript Object Notation), he often used console logs and the terminal to read the data. On rarer occasions, plain text editors.He often shared the provisional texts rendered in a Terminal within the Slack messaging application. For example, during the City Payroll project, he created 12 provisional texts by slicing his downloaded export of the city's open portal data (see Figure 4.2). After receiving some commentary from his colleagues, which often included refined angles to the story, Ray would go back, write some code to create new provisional texts (see Fig. 4.3) to then share in Slack (see 4.5) by directly copying and pasting the `console.table()` method (see Fig. 4.4).

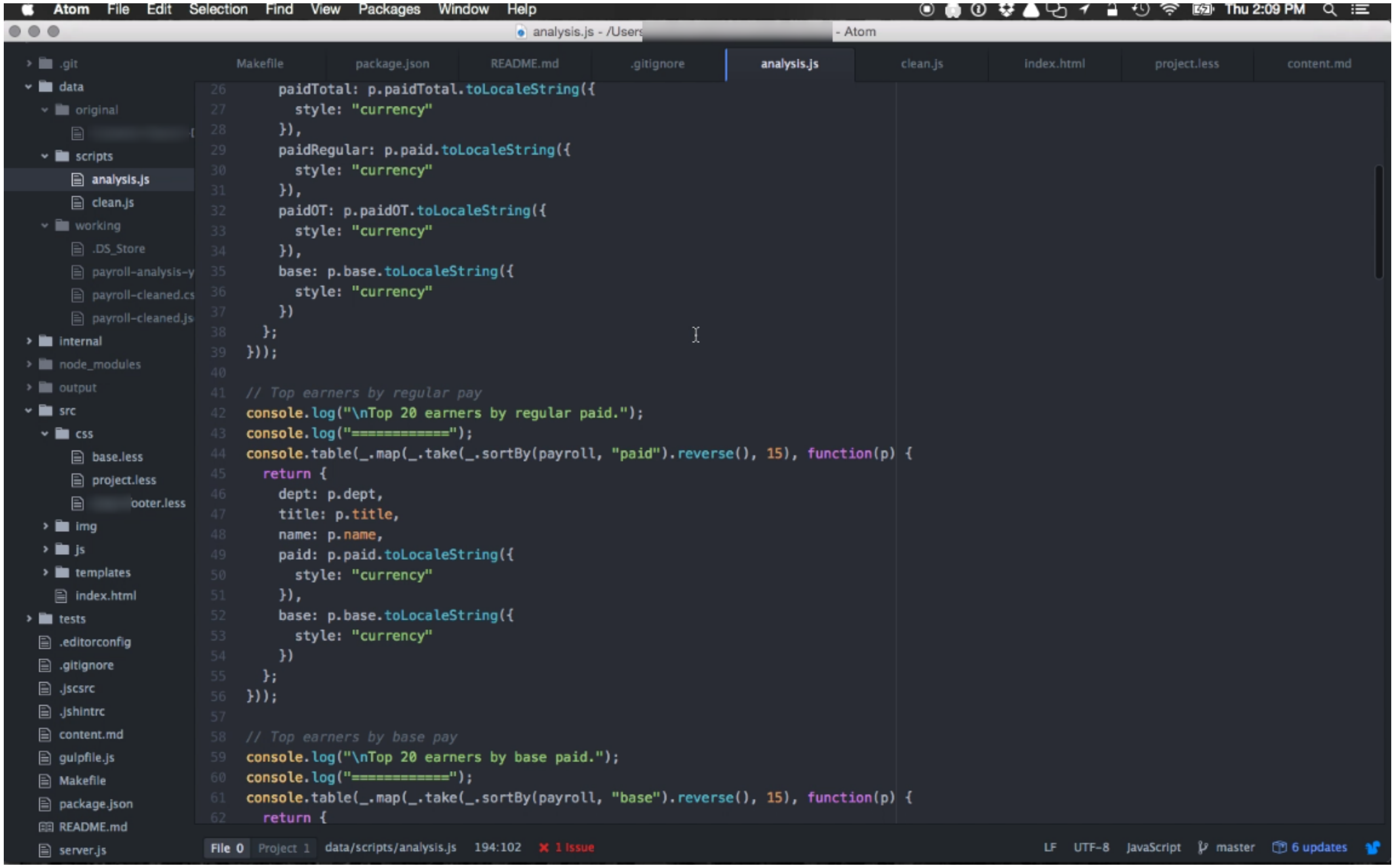


Jobs

This dataset contains current job postings available on the City of official jobs site

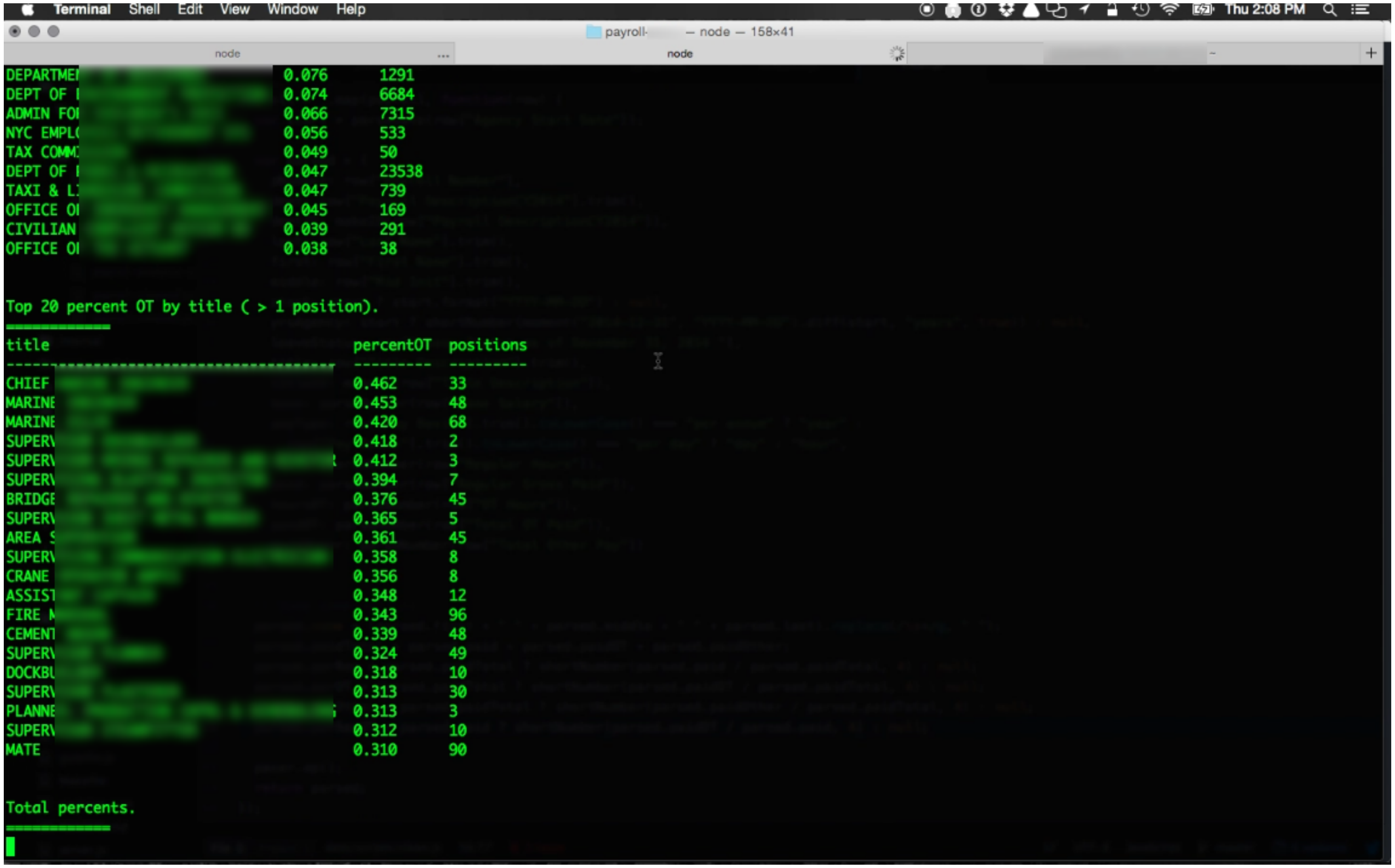
	Civil Service Title	Title Code No	Level	Salary Range From	Salary Range To	Salary Frequency	
	GENCY ATTORNEY INTERNE	30	00	[redacted]	[redacted]	Annual	1
	ONSTRUCTION PROJECT MANAGER I	34	00	[redacted]	[redacted]	Annual	59
	ONSTRUCTION PROJECT MANAGER I	34	00	[redacted]	[redacted]	Annual	59
	OMPUTER SYSTEMS MANAGER	10	M1	[redacted]	[redacted]	Annual	28
	OMPUTER SYSTEMS MANAGER	10	M1	[redacted]	[redacted]	Annual	28
	ITY RESEARCH SCIENTIST	21	02	[redacted]	[redacted]	Annual	59
	ITY RESEARCH SCIENTIST	21	02	[redacted]	[redacted]	Annual	59
	ORT MARINE ENGINEER	06	00	[redacted]	[redacted]	Annual	W
	ORT MARINE ENGINEER	06	00	[redacted]	[redacted]	Annual	W
	ITY PARK WORKER	90	00	[redacted]	[redacted]	Annual	59
	ERT. IT ADMINISTRATOR WAN	13	03	[redacted]	[redacted]	Annual	42
	ERT. IT ADMINISTRATOR WAN	13	03	[redacted]	[redacted]	Annual	42
	GENCY ATTORNEY	30	02	[redacted]	[redacted]	Annual	La
	GENCY ATTORNEY	30	02	[redacted]	[redacted]	Annual	La
	ITY PARK WORKER	90	00	[redacted]	[redacted]	Annual	59
	[illegible]	31	02	[redacted]	[redacted]	Annual	Ot
	[illegible]	31	02	[redacted]	[redacted]	Annual	Ot
98	[illegible]	90	00	[redacted]	[redacted]	Annual	He
99	[illegible]	90	00	[redacted]	[redacted]	Annual	He
100	[illegible]	10	M1	[redacted]	[redacted]	Annual	59
101	[illegible]	10	M1	[redacted]	[redacted]	Annual	59
102	[illegible]	22	00	[redacted]	[redacted]	Annual	59
103	[illegible]	10	M1	[redacted]	[redacted]	Annual	15
104	[illegible]	10	M1	[redacted]	[redacted]	Annual	15

Figure 4.2: Screen capture of where Ray retrieved the City Payroll data: an online open data portal.



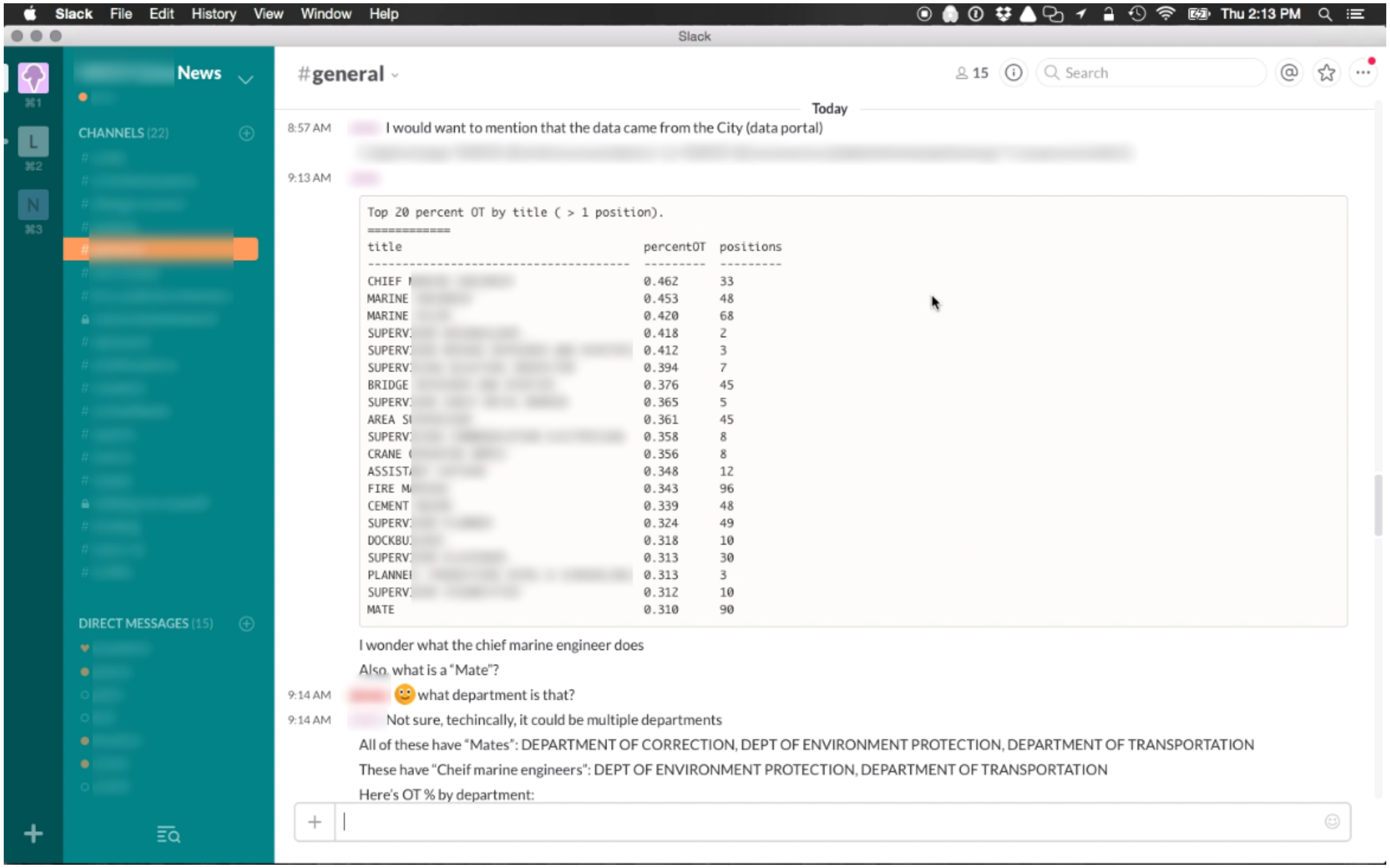
```
    paidTotal: p.paidTotal.toLocaleString({
      style: "currency"
    }),
    paidRegular: p.paid.toLocaleString({
      style: "currency"
    }),
    paidOT: p.paidOT.toLocaleString({
      style: "currency"
    }),
    base: p.base.toLocaleString({
      style: "currency"
    })
  };
}));

// Top earners by regular pay
console.log("\nTop 20 earners by regular paid.");
console.log("============");
console.table(_.map(_.take(_.sortBy(payroll, "paid").reverse(), 15), function(p) {
  return {
    dept: p.dept,
    title: p.title,
    name: p.name,
    paid: p.paid.toLocaleString({
      style: "currency"
    }),
    base: p.base.toLocaleString({
      style: "currency"
    })
  };
}));

// Top earners by base pay
console.log("\nTop 20 earners by base paid.");
console.log("============");
console.table(_.map(_.take(_.sortBy(payroll, "base").reverse(), 15), function(p) {
  return {
```

Figure 4.3: Screen capture of Ray's code that produced the provisional texts.



```
DEPARTME[redacted]   0.076     1291
DEPT OF [redacted]   0.074     6684
ADMIN FO[redacted]   0.066     7315
NYC EMPL[redacted]   0.056     533
TAX COMM[redacted]   0.049     50
DEPT OF [redacted]   0.047     23538
TAXI & L[redacted]   0.047     739
OFFICE O[redacted]   0.045     169
CIVILIAN[redacted]   0.039     291
OFFICE O[redacted]   0.038     38


Top 20 percent OT by title ( > 1 position).
===========
title                                percentOT  positions
------------------------------------ ---------  ---------
CHIEF [redacted]                     0.462      33
MARINE[redacted]                     0.453      48
MARINE[redacted]                     0.420      68
SUPERV[redacted]                     0.418      2
SUPERV[redacted]                     0.412      3
SUPERV[redacted]                     0.394      7
BRIDGE[redacted]                     0.376      45
SUPERV[redacted]                     0.365      5
AREA S[redacted]                     0.361      45
SUPERV[redacted]                     0.358      8
CRANE [redacted]                     0.356      8
ASSIST[redacted]                     0.348      12
FIRE M[redacted]                     0.343      96
CEMENT[redacted]                     0.339      48
SUPERV[redacted]                     0.324      49
DOCKBU[redacted]                     0.318      10
SUPERV[redacted]                     0.313      30
PLANNE[redacted]                     0.313      3
SUPERV[redacted]                     0.312      10
MATE                                 0.310      90


Total percents.
===========
█
```

Figure 4.4: Screen capture of the output for Ray's code that outputs the provisional texts in the Terminal.



8:57 AM [redacted] I would want to mention that the data came from the City (data portal)

[redacted]

9:13 AM [redacted]

```
Top 20 percent OT by title ( > 1 position).
============
title                                      percentOT  positions
-----------------------------------------  ---------  ---------
CHIEF [redacted]                           0.462      33
MARINE [redacted]                          0.453      48
MARINE [redacted]                          0.420      68
SUPERV[redacted]                           0.418      2
SUPERV[redacted]                           0.412      3
SUPERV[redacted]                           0.394      7
BRIDGE [redacted]                          0.376      45
SUPERV[redacted]                           0.365      5
AREA S[redacted]                           0.361      45
SUPERV[redacted]                           0.358      8
CRANE [redacted]                           0.356      8
ASSIST[redacted]                           0.348      12
FIRE M[redacted]                           0.343      96
CEMENT [redacted]                          0.339      48
SUPERV[redacted]                           0.324      49
DOCKBU[redacted]                           0.318      10
SUPERV[redacted]                           0.313      30
PLANNE[redacted]                           0.313      3
SUPERV[redacted]                           0.312      10
MATE                                       0.310      90
```

I wonder what the chief marine engineer does

Also, what is a "Mate"?

9:14 AM [redacted] 🙂 what department is that?

9:14 AM [redacted] Not sure, techincally, it could be multiple departments

All of these have "Mates": DEPARTMENT OF CORRECTION, DEPT OF ENVIRONMENT PROTECTION, DEPARTMENT OF TRANSPORTATION

These have "Cheif marine engineers": DEPT OF ENVIRONMENT PROTECTION, DEPARTMENT OF TRANSPORTATION

Here's OT % by department:

Figure 4.5: Screen capture of Ray's copy and pasted console table, as it renders in Slack, where he shares the provisional texts with his team.

Ray's ensemble of coding tools produced similar information flows during the State Toxic Sites and Health Texting projects. During the State Toxic project, Ray would either receive or construct a coding task, which included slicing the data to create a provisional text. He would use console logs to print these values in the Terminal, which he subsequently shared in Slack. Ray created a slightly more involved collection of aggregate perspectives during the Health Texting project by creating real-time slices of the database during the texting campaign. In this case, the podcasting team used the different slices of the data to develop particular talking points for the show. One of the assistants to the show would verify their interpretations of the data by sharing an excerpt of the day's script with Ray. (See evidence for this use of particular technologies in Appendix A.11 for the State Toxic project and Appendix A.12 for Health Texting.)

I briefly report some of these characteristics and uses of Ray's tools to process and analyze data, because of the tacitness that their capabilities afford him to read and write provisional texts. Each of these tools renders data and code in socially organized ways through various semiotic modes; that is, "culturally recognizable channels" (Wickman, 2010, p. 288, fn. 4) that the technology renders durable for him.[42]

In order to continue developing a language to discuss how Ray coded provisional texts, I provide a broader overview of Ray's tools and semiotic resources rendered by the above set of technologies. First, I report broader findings about particular file formats, data structures, and data types. Then, I discuss and define the recurrent computational methods. From there, I define the main style that Ray uses when coding provisional texts: *functional programming.*

***File Formats, Data Structures, and Data Types*:** The important item to recognize at this juncture is how the formats of the files define the set of constraints by which a person coding can express data structures. Indeed, digital file formats and the grammars of different data structures and data types provide the durable grounds for people

[42] Future analysis across all of the embedded units of analysis could help me develop a more robust picture about about how these various technologies support Ray's reading and writing activities.

to code information into meaningful texts. Such formatting constraints offer coders the ability to express meaningful relationships across the dimensions of a smaller data structure within a code file to the high-dimensional nature of some databases. Formats both reduce and open up the potential interpretations of any structured information. Table 4.3 provides a list of Ray's commonly used formats, structures, and types while he coded provisional texts.

Table 4.3: Ray's commonly used file formats, data structures, and data types, while coding provisional texts.

Data Types & Formats	Basic Definition
JS Data Types	
Object	Fundamentally, a complex data type with inherited properties and methods.
String	Express sequence, i.e., "string", of characters and symbols.
Int	Expresses Integers.
Float	Expresses floating-point numbers.
Boolean	Expresses some-thing or condition as either being `true` or `false`.
JS Data Structures	
Array	Organizes data types into lists.
Array Object	Organizes data types into lists with different capabilities, such as key-value pairs, as well as inherited properties and methods.
File Formats	
JSON	JavaScript Object Notation is a language-independent, standardized data-interchange format.

Continued on next page

Table 4.3 – continued from previous page

Data Types & Formats	Basic Definition
CSV	A language-independent file format that structures data into Comma-Separated Values.
JS	JavaScript programming language file, which adheres to the JS language interpreter engine's defined ways to structure data and data type relations.

I do not have the space to explain the nuances of each of the aforementioned data types and formats in this section. Instead, it is more important to explain 2 more important formats central to the findings discussed in the following section 4.3.1: CSV and JSON. These 2 particular file formats may seem somewhat foreign to traditional forms of writing and textual production. I also briefly define and discuss another digital format that Ray used during the Health Texting project: MongoDB and its document-model for databases. In what follows, I discuss these 3 in more detail, and show how the 3 formats carry forward certain legacies of print media.

For instance, CSV files operate the assumption of 2-dimensional, tabular value organization; namely, a basic spreadsheet. Ray, among others more broadly, dub CSV files as "flat files" due to its 2-dimensional nature. Developers like coding with CSV files for their simplicity, since parsers are natively found in any language. Additionally, they are relatively easy to read. The first row is reserved as a header for naming each column. Every new line with its comma-separated values thereafter denotes a row under that column schema. Ray too recognized these features of CSV files and mentioned how he thought they helped create sets of data that were "programmatically" richer than Excel spreadsheet and PDF files, for instance.[43] In other words, CSV files can more easily

[43] Ray voiced the opinion that file formats, such as Excel, were more acutely defined as application formats, and PDFs as document formats.

be parsed with programming languages than application files or static PDF files. Ray also recognized how CSV files were commonplace formats for database exports. In an interview, he noted how databases organize datum with table-based schemas, as defined by SQL code, and exporting such data as a CSV may "flatten" those hierarchical relationships made more explicit in the database itself. Indeed, recall section 3.3.1, where I noted how Ray encountered this issue attempting to understand the inter-relationships between the 2-dimensional table in a PDF file provided by a reporter, who downloaded it from an open portal database online. Such semiotic modes of these 2-dimensional CSV file formats mediated Ray's reading and writing with data sets.

JSON data formats also mediated Ray's coding and reading. Often, Ray needed to complete the output step by creating a JSON format of the data: State Toxic Sites, City Restoration, City Payroll, etc. JSON incorporates numerous data types and hierarchies that distinguishes it from CSV files.[44] For brevity, JSON can include multiple nested hierarchies, while CSV cannot. Additionally, JSON files can incorporate objects with member key-value pairs and arrays to create quite complex relationships across the data.

Other flavors of JSON exist and are used by Ray, since he works with digital maps. For example, during the City Restoration project, he integrated all of the different data outputs into one TopoJSON file. Topo stands for topology, and is used specifically for mapping-based data. It differs from generic JSON in that it contains a header with geometrical data important for the digital mapping environment to know how to interpret and render the coordinates within the file. In relationship to the areas defined in the TopoJSON file, the multi-dimensional nature of the file also enabled Ray to integrate the technical shape-file information with the pertinent data of interest to the story linked to the *applicants per neighborhood* perspective. (See Fig. **??** below for an excerpt of the topoJSON file used in the City Restoration project.)

The nested nature of JSON objects enable Ray to translate different types of relationships akin to SQL tables, subtables, or nested tables. JSON files can capture some

[44] For more technical information, see the standards page: `http://www.json.org/`

```
{
  "type":"Topology",
  "objects":{
    "neighborhoods-census-[redacted].geo":{
      "type":"GeometryCollection",
      "crs":{
        "type":"name",
        "properties":{
          "name":"urn:ogc:def:crs:[redacted]"
        }
      },
      "geometries":[{
        "type":"Polygon",
        "properties":{
          "neighborhood":"[redacted]",
          "borough":"[redacted]",
          "nonWhiteP":0.03242320819894011,
          "total":3515.9999991526615,
          "active":615,
          "startedConstructionP":0.2458233890214797
        },
      "bbox":[
    -11.12345678912345,11.12345678912345,-11.12345678912345,11.12345678912345
    ],
      "id":"[redacted]",
      "arcs":[[0]]
      },
      {
      ...
      },
    }]
    "transform":{
      "scale":[0.000053058873083l6862,0.000018988100012205926],
      "translate":[-11.12345678912345,11.12345678912345]
    }
  }
}
}
```

Figure 4.6: Excerpt from Ray’s `city-restoration-neighborhoods.topo.json` file, which Ray used to generate an interactive map with computed vales. Note that the excerpt has been formatted with white space to make evident the hierarchical relationships, since the original file was minified for better browser performance.

of the complexity that modern databases define explicitly amongst datum, which becomes useful when feeding such datapoints to become available across the web. JSON file formats, akin to code files more generally, draw from *a prior* expressions of logical systems from mathematics through the use of different syntaxes. As seen in the above Fig. 4.6, JSON uses curly brackets to denote groups of information related to each other in some meaningful (*cf.* Goody's "semantic fields" (1977/2001, p. 47)). Overall, Ray recognized how certain file formats help him conduct his data-processing and analysis work, putting aggregate information into programmatic, functional, and meaningful structures.

In Table 4.4 below, I provide selected coded outputs from the 3 projects discussed in this chapter, including selected modalities and inscriptions. During the Health Texting project, Ray used another digital format to organize information with computational modalities: MongoDB's non-relational, document database structure. According to MongoDB (2017b), the company that oversees this particular database structure and standard, any record of some thing or activity in MongoDB "is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents." In essence, the MongoDB document model applies the JSON object structure to database architecturing. MongoDB's documents share many of the modalities of JSON file formats, making it easier to export data into web-ready structures that are easy to manipulate with the JS programming language.

Table 4.4: Selected list of data set files for the 3 data team projects discussed in chapter 4, including their inscriptions and modalities.

Project	**(Selected) Modalities**	**(Selected) Inscriptions**
State Toxic Sites		

Continued on next page

Table 4.4 – continued from previous page

Project	(Selected) Modalities	(Selected) Inscriptions
sites-locations.csv, sites-details.csv	CSV: Comma-Separated Values (CSV), tabular, 2-dimensional: rows and columns / records and fields, 'flat file'	Latitude, Longitude, No LSRP (Licensed Service Remediation Professional), IEC (Immediate Environmental Concern)
City Payroll		
payroll-analysis.json	JSON: multi-dimensional array, nested hierarchies, objects, key-value pairs, camel-case labeling scheme	title, dept, payType, paidTotal, perReg, perOT
Health Texting		
db.js: Database model that structured the data collected from the texting campaign	MongoDB: NoSQL, document-oriented, multi-dimensional array, nested hierarchies, objects, key-value pairs, 'Person' model, 'person' schema	Name, phone number, email address, campaign goals, responses to campaign questions/prompts about information consumption

Overall, I discuss these 3 particular ways to structure information to emphasize the language work involved in not only naming things and observations to be stored on a server, but also how such inscriptions must be thoughtfully structured and put into meaningful relationships that will help Ray and the teams complete their data-driven objectives.

***Computational Methods*:** In addition to these constraints afforded by data types, structures, and formats, Ray's data processing and analysis activity included a wide variety of computational methods within the JS source-code files. Table 4.5 includes results from a search across 45 JS files linked to his data processing and analysis activities conducted in this embedded unit.

Table 4.5: JavaScript (JS) and JS code library computational methods across 45 data-processing and analysis files.

Computational Method	**Number of Instances**	**Basic Definition**
`.log()`	81 across 18 files	Prints out value to the console / terminal.
`.length`	57 across 18 files	JS method computes length of array.
`.map()`	49 across 16 files	JS and lodash method creates new array object with another array as its argument.

Continued on next page

Table 4.5 – continued from previous page

Computational Method	Number of Instances	Basic Definition
`.sum()`	36 across 4 files	lodash method that computes sum of values in arrays.
`.each()`	32 across 9 files	lodash method that iterates across a collection; plus invokes function to perform on each value.
`.filter()`	27 across 10 files	lodash method that returns a new filtered array, based on defined parameter.
`.pluck()`	24 across 4 files	Now deprecated lodash method; performs similarly to `.map()`
`.push()`	23 across 7 files	JS method that adds 1 or more elements to the end of an array.
`.sortBy()`	17 across 4 files	lodash method that creates an array of elements, which are sorted by an iteratee.

Continued on next page

Table 4.5 – continued from previous page

Computational Method	Number of Instances	Basic Definition
`.find()`	16 across 8 files	lodash method that iterates over a collection and returns first instance of defined argument.
`.reverse()`	15 across 3 files	lodash method that reverses the order of elements in an array.
`.take()`	12 across 2 files	lodash method that takes a defined number of elements from the front of an array
`.findWhere()`	12 across 7 files	Underscore method that iterates over a collection and returns first instance of defined argument.
`.groupBy()`	8 across 3 files	lodash method that creates an object composed of keys generated from the results of running each element of collection thru iteratee.
`.slice()`	8 across 4 files	lodash method that creates a slice of array from start up to, but not including, end.

Of particular note, the *lodash* code library was important to Ray's coding to process and analyze data sets. A code library is an extraneous set of computational methods that developers create to work with an existing programming language. Typically, developers create libraries in lieu of some recurrent need in their domains of practice. Libraries add goal-oriented expressiveness to the more general features of any programming language. The code library, lodash, includes more general methods to increase the expressive capabilities for developers working with complex arrays or objects, as well as finer-grained data-types, such as strings.[45]

It is not that Ray could not make do without lodash. However, he would often find himself repeating similar functions that take some similar kind of data inputs to produce some similar goal-oriented outputs. In short, code libraries are often the products of abstracted and recurrent developer goals and actions: ways to express and do things with data. The lodash library helps Ray keep his code files shorter and more expressive toward the goals of understanding and finding stories within data sets. However, the identification of these computational modes means very little without understanding how Ray puts them to use within his situated practice. Before reporting such findings, one more aspect of Ray's coding needs definition: JS *functions*.

***JavaScript Functions*:** JS functions played a big role in Ray's coding of provisional texts. Generally, functions take in some input as an argument, perform an operation as defined by the coder on that input, then return / output a new expression of that data for some other use. In JS, functions are also objects, since the standard defines their inheritance of particular native properties and methods. The main difference between standard JS objects and JS function objects is that JS functions can be called upon to perform a coder-defined operation.[46]

[45] Take another case, the svg.js code library, which helps developers who write JS with SVG files on the web, or the popular jQuery.js library for web developers who work with the Document Object Model of browsers.

[46] For more general information, reference the Mozilla Developer Network definition of JS functions: `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions`.

For example, Ray used mainly anonymous functions – functions with no names – to perform his provisional text work. The function in the below example in Fig. 4.7 enabled Ray to chain together numerous functions as arguments. Each of the anonymous functions in this code snippet performs an operation or returns some new form of data for the overall expression that includes the following chained methods: `.sortBy()`, `.map()`, `.groupBy()`, and `.filter()`. Each method uses the previous function's output as input to satisfy the respective argument. To read this code, I should note that it begins with the inner-most function, `.filter()`, which takes the larger aggregate data, `payroll` (line 4), as its initial argument. Curly braces – {. . . } – demarcate the scope of each function. Put another way, scope refers to the ways the JS language helps a person identify how, when, and where a computer will be able to use and share the particular digital data of concern. In JS, parantheses located after the function name – (. . .) – serve as a place for people to define what data the function accepts as its input, as well as what may be done to the data.

```
// Years and percent people with overtime
console.log("\nYears (> 1) and percent positions with overtime.");
console.log("============");
console.table(_.sortBy(_.map(_.groupBy(_.filter(payroll, function(p) {

  // Uses yrsAgency value to .filter() data
  return p.yrsAgency > 1;

}), function(p) {

  // Rounds yrsAgency to largest integer less than or equal to number
  // Returns that value to group the filtered data
  return Math.floor(p.yrsAgency);

}), function(group, gi) {

  // Uses filtered and grouped data to .filter() by paidOT datum
  var ot = _.filter(group, function(g) {
    return g.paidOT > 0;
  });

  // Uses all computed values from above to .map() the desired percentages
  return {
    Years: parseInt(gi),
    "% of positions with OT": (ot.length / group.length).toFixed(3),
    "% of OT pay": (_.sum(_.pluck(group, "paidOT")) / _.sum(_.pluck(group,
    "paidTotal"))).toFixed(3),
    "# of positions": group.length.toLocaleString()
  };

  // Takes the above desired, mapped data and sorts it by highest
  // # of years employed at the top of the list
}), "years").reverse());
```

Figure 4.7: Excerpt from Ray's `analysis.js` file, which he wrote during the City Payroll project. Comments added for readability.

There, of course, is much to unpack in the above code that produced a provisional text. However, I use this example as a way to acclimate readers, who may not have experience reading code or JS code. The important take-away from this short example is the recognition of Ray's creative use of language to manipulate the `payroll` data, which produces a provisional text about the percentage of people who received overtime based on how long they have been at an agency. This code example contains some of representative semiotic resources that Ray uses when he codes provisional texts: slicing the original exported data into a more refined selection of organized datum through coding with filter, grouping, sorting, and mapping functions. Through coding such as this, he hopes to further refine the angle of the germ of a story.

Semiotic modes and habits of mind: A finer perspective of Ray's coding

In this section, I report findings regarding the following research question: *How do the tools semiotic resources associated with data sets mediate Ray's coding of provisional texts?* After examining the finer-grained data of Ray coding provisional texts during 3 of the projects, I observed how the tools and technical semiotic modalities of Ray's code were intertwined with his epistemic work to understand the goals of the project and meaning of the values in the data sets. From my analysis of the data listed in Table 4.1, I constructed the following major findings about Ray's coding of provisional texts:

1. Ray's coding supported and was epistemic.

2. Ray's coding incorporated and coordinated semiotic resources and tools through a particular habit of mind: "What data type is this? And what does it mean?"

3. Ray's coding was shaped by historical knowledge of the modalities linked to data sets.

In what follows, I support these findings in respective subsections by using data from the State Toxic Sites, City Payroll, and Health Texting projects.

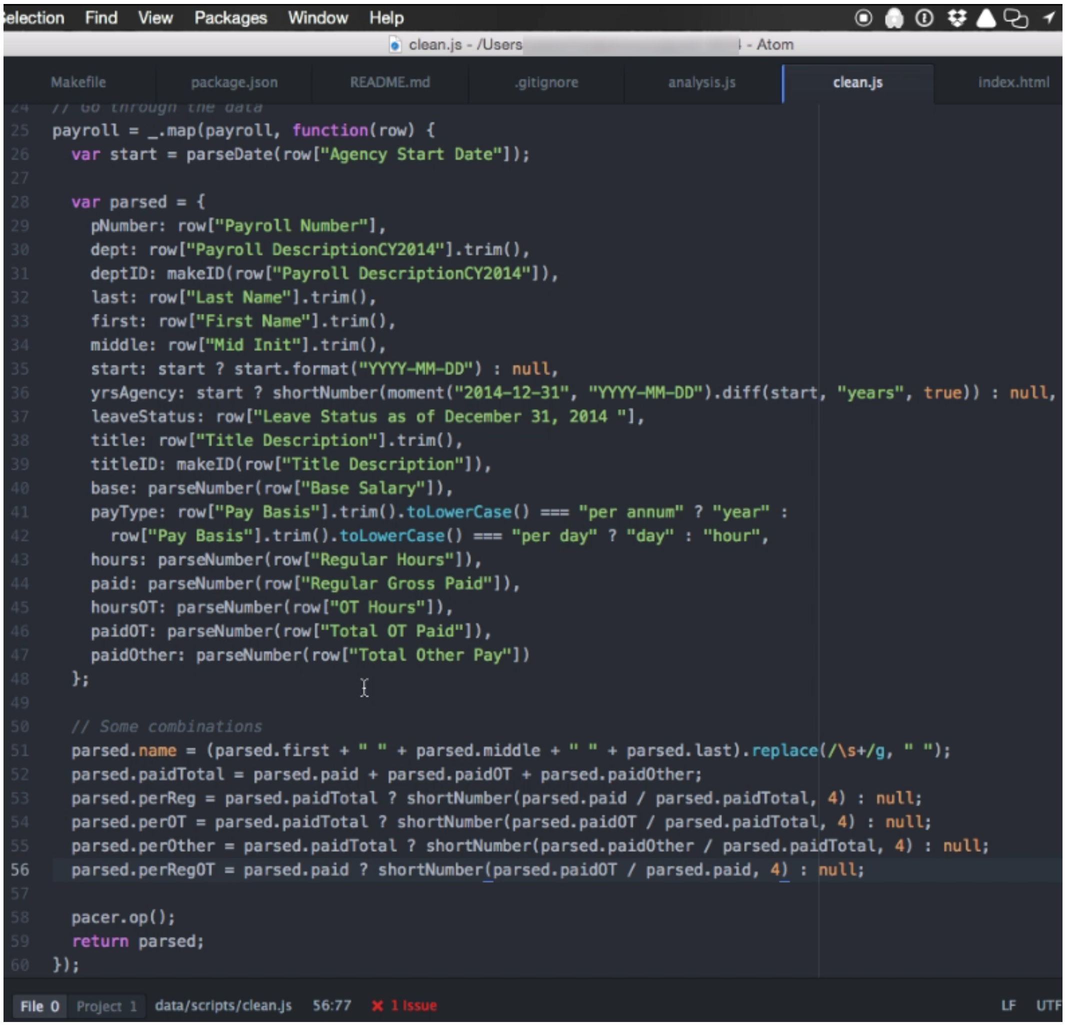
***Coding as epistemic*:** As I discussed in Chapter 3, Ray's coding work with data sets included acts to slice data into new perspectives that help develop or refine angles for finding stories for publication. In this section, I report instances showing how Ray's activity to produce and use these provisional texts served epistemic actions.

Epistemic work and the mediated nature of it can be seen in studies of both reading and writing. Haas and Flower (1988) and Haswell et al. (1999) both identify the ways reading is a constructive, meaning-making activity, which is shaped by subjective contexts brought to bear on the task. Heath (1982), too, and her study of parents reading to their children, shows how different families from different social backgrounds drew different outcomes and uses for the content of the book. Writing is epistemic in that studies of writing (Haas & Takayoshi, 2011; Pigg, 2014a; Solé, Miras, Castells, Espino, & Minguela, 2013; Wickman, 2010) show how writers develop an acumen for synthesizing the material-semiotic dimensions of texts with the social-material dimensions of the contexts in which the texts both represent and mediate. In this subsection, I report how Ray's coding shares these qualities with other forms of reading and writing.

In Ray's case, reading data sets in coordination with his correspondence with his colleagues shaped what and how he constructed and carried out his coding of provisional texts. During the City Payroll project, Ray shared 12 provisional texts with his colleagues on Slack, which he sliced from the original data that he pulled from the open data portal online. He coded them by first "cleaning" the data export of over 500,000 rows, which he did by translating the downloaded CSV file into JSON with the `clean.js` file. (See Fig. 4.8 below. For the complete code, see Appendix A.10.) Within this processing file, Ray first translated the original CSV export format into the array object with key-value pairs, as seen in lines 28-48 below. The variable name, `parsed`, connotes how the new format of the data as an object array for each CSV row maintains the original *per Employee* aggregate perspective.

Ray remarked how the original data set provided by the city was "not too detailed. . . . Pretty basic stuff: Department position, title, or name. They have this time that they started at the agency, which is interesting, but some of it is inaccurate, so I'm not sure what to do with it." Despite not knowing exactly what to do with the data quite yet, Ray coded new combinations to add to each row of the original data. Indeed, Ray used some of the original values to create new values, which he thought may prove useful for his future analysis work.

I later learned that his remark about the start-time at the agency and the lack



```
24  // Go through the data
25  payroll = _.map(payroll, function(row) {
26    var start = parseDate(row["Agency Start Date"]);
27
28    var parsed = {
29      pNumber: row["Payroll Number"],
30      dept: row["Payroll DescriptionCY2014"].trim(),
31      deptID: makeID(row["Payroll DescriptionCY2014"]),
32      last: row["Last Name"].trim(),
33      first: row["First Name"].trim(),
34      middle: row["Mid Init"].trim(),
35      start: start ? start.format("YYYY-MM-DD") : null,
36      yrsAgency: start ? shortNumber(moment("2014-12-31", "YYYY-MM-DD").diff(start, "years", true)) : null,
37      leaveStatus: row["Leave Status as of December 31, 2014 "],
38      title: row["Title Description"].trim(),
39      titleID: makeID(row["Title Description"]),
40      base: parseNumber(row["Base Salary"]),
41      payType: row["Pay Basis"].trim().toLowerCase() === "per annum" ? "year" :
42        row["Pay Basis"].trim().toLowerCase() === "per day" ? "day" : "hour",
43      hours: parseNumber(row["Regular Hours"]),
44      paid: parseNumber(row["Regular Gross Paid"]),
45      hoursOT: parseNumber(row["OT Hours"]),
46      paidOT: parseNumber(row["Total OT Paid"]),
47      paidOther: parseNumber(row["Total Other Pay"])
48    };
49
50    // Some combinations
51    parsed.name = (parsed.first + " " + parsed.middle + " " + parsed.last).replace(/\s+/g, " ");
52    parsed.paidTotal = parsed.paid + parsed.paidOT + parsed.paidOther;
53    parsed.perReg = parsed.paidTotal ? shortNumber(parsed.paid / parsed.paidTotal, 4) : null;
54    parsed.perOT = parsed.paidTotal ? shortNumber(parsed.paidOT / parsed.paidTotal, 4) : null;
55    parsed.perOther = parsed.paidTotal ? shortNumber(parsed.paidOther / parsed.paidTotal, 4) : null;
56    parsed.perRegOT = parsed.paid ? shortNumber(parsed.paidOT / parsed.paid, 4) : null;
57
58    pacer.op();
59    return parsed;
60  });
```

Figure 4.8: Screen capture of Ray's processing file with newly added computed values to the `parsed` array object (lines 51-56).

of "detail" became an important insight for him and the team. Recall how Ray's colleague, Phil, offered Ray 3 different angles, after Ray shared a series of provisional texts about overtime (see Section 3.3.2 – Slicing). Ray shared slices at essentially 2 main perspectives: *per department* and *top 20 earners*. Phil responded to such slices with a new angle to analyze the data with more specificity. Phil was curious if there were any interesting relationships between overtime pay and other dimensions of time at the agency to find out what Phil put as: "How does overtime relate to seniority?".

These findings indicate that the output, the provisional texts themselves, operated as epistemic objects (Hutchins, 1995; Kirsh, 2010) within this domain of activity with the objective to find stories in the data. Ray responded to Phil by saying that he is not "confident" that the available start-time datapoint "necessarily equates to seniority." In this moment, Ray reconciled his concept of seniority with his current understanding about how it fits with the available data. In so doing, he scoped out the boundaries of what the perspective of this data can indeed provide.

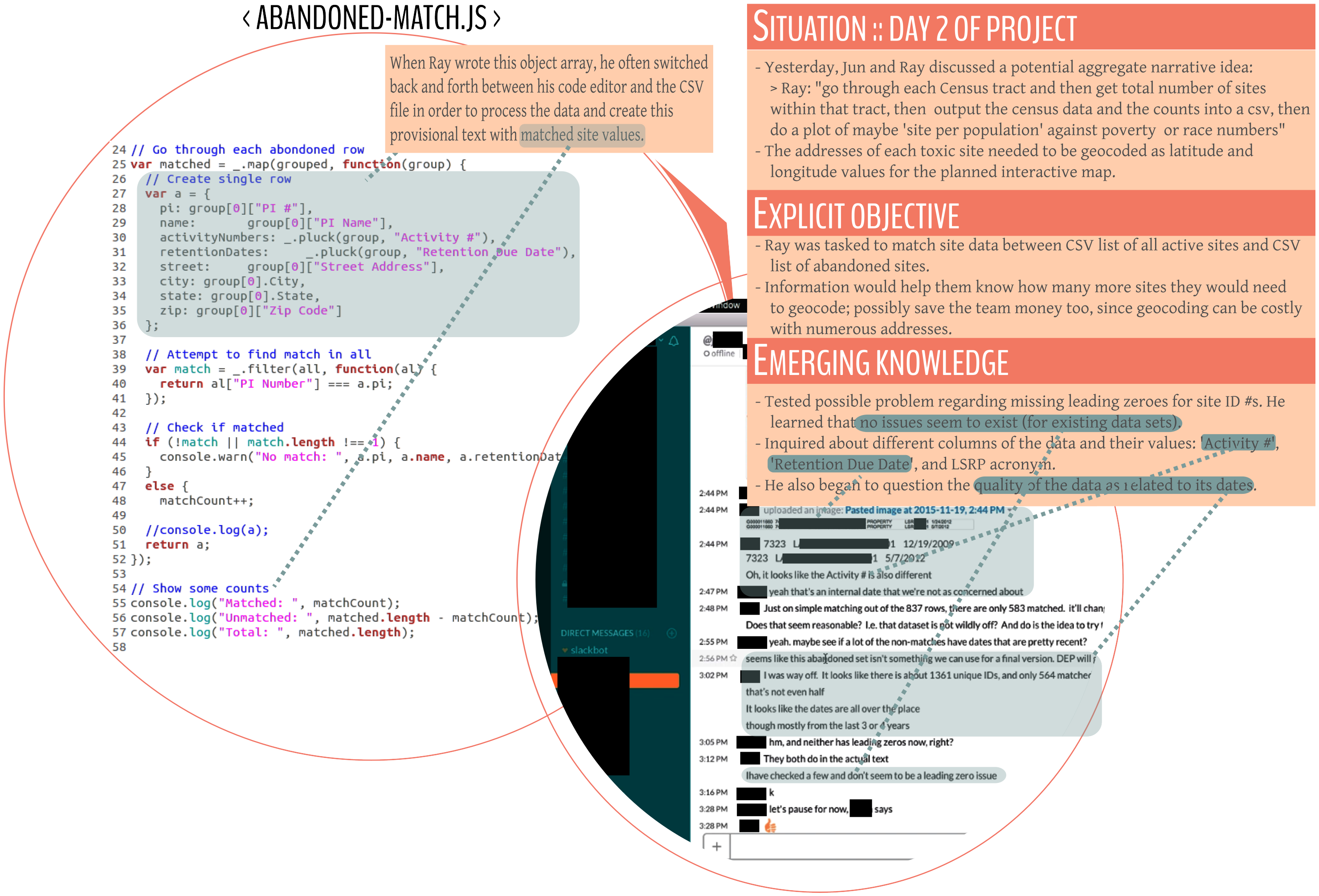
This limitation of the data became more apparent as the sharing of the provisional texts carried on over into the next day. So much so, that Vince, coming back into the discussion on Slack, seemed to sense this as well. Consequently, the project that Ray hoped would serve as a quick turnaround story died, when Vince said that the data needs more qualitative support with reporting work. In essence, the stories found with the perspectives offered by the data needed more contextual information to make them worth reporting and publishing. While these findings gesture toward the knowledge work mediated by provisional texts, how did Ray's coding of them serve as meaning-making activity?

As a way to answer such a research question, I began to piece together the finer moves made during particular situations in which Ray coded provisional texts. I traced some of the situations during the State Toxic Sites project by using observational data, interviews, retrospective accounts, and Slack logs. This tracing process led to insights surrounding the original reasons motivating the production of a particular texts, but also Ray's *emerging knowledge* about the data itself. Ray's emerging knowledge did not necessarily relate directly to the task itself, but always connected back to insight about the project at a broader level. Findings indicate that data-processing and analysis tasks provided Ray the time to engage with the sets of data and its textual properties.

In Figure 4.9, I gathered up all of the available, contextual details that shaped and were shaped by Ray's coding of the `abandoned-match.js` file; that is, information that never was chronicled within the code or other texts in the project directory. By doing so, I was able to discern some nuance about Ray's meaning-making when coding the seemingly mundane code to create provisional texts. Through this process of accounting for Ray's explicit coding exigence and situation to process or analyze data, I discovered how such coding with data-sets provide Ray opportunities to develop new, sometimes unexpected emergent knowledge about the information.

The code and contextual notes in Figure 4.9 represent Ray's first coding task on the Toxic Sites project. Prior to this coding task, Ray and Jun previously discussed the analysis goals and probable interactive-mapping objectives. In this case, probable conveys their plan to map the toxic site data in relationship with the 1-mile buffer radius created with the Census tract data. However, the team had yet to test the data for possible trending problems, so they did not know what kind of map experience could be created with the eventual output data. Since location data was important for their goals to map the data, Jun asks Ray to match the site IDs from the abandoned data set against the complete list of all sites, so they could learn how many more site addresses need to be geocoded.[47] From this explicit objective, Ray constructed the coding task to "Match the abondoned [sic] data with the list of all sites," as evinced by this comment written at the top of the file.

[47] Geocoding is the process to take location information and from that information calculate the approximate coordinates of that location. Typically, the coordinates are in latitude and longitude. Ray's team uses Google's geocoding services, which has a 2,500 requests per day rate-limit. Since the data has tens of thousands of rows, they are attempting to save money by finding existing sites with already provided coordinates.



‹ ABANDONED-MATCH.JS ›

```
// Go through each abondoned row
var matched = _.map(grouped, function(group) {
  // Create single row
  var a = {
    pi: group[0]["PI #"],
    name:       group[0]["PI Name"],
    activityNumbers: _.pluck(group, "Activity #"),
    retentionDates:     _.pluck(group, "Retention Due Date"),
    street:     group[0]["Street Address"],
    city: group[0].City,
    state: group[0].State,
    zip: group[0]["Zip Code"]
  };

  // Attempt to find match in all
  var match = _.filter(all, function(al) {
    return al["PI Number"] === a.pi;
  });

  // Check if matched
  if (!match || match.length !== 1) {
    console.warn("No match: ", a.pi, a.name, a.retentionDat
  }
  else {
    matchCount++;

  //console.log(a);
  return a;
});

// Show some counts
console.log("Matched: ", matchCount);
console.log("Unmatched: ", matched.length - matchCount);
console.log("Total: ", matched.length);

```

When Ray wrote this object array, he often switched back and forth between his code editor and the CSV file in order to process the data and create this provisional text with matched site values.

Situation :: Day 2 of Project

- Yesterday, Jun and Ray discussed a potential aggregate narrative idea:
  > Ray: "go through each Census tract and then get total number of sites within that tract, then output the census data and the counts into a csv, then do a plot of maybe 'site per population' against poverty or race numbers"
- The addresses of each toxic site needed to be geocoded as latitude and longitude values for the planned interactive map.

Explicit Objective

- Ray was tasked to match site data between CSV list of all active sites and CSV list of abandoned sites.
- Information would help them know how many more sites they would need to geocode; possibly save the team money too, since geocoding can be costly with numerous addresses.

Emerging Knowledge

- Tested possible problem regarding missing leading zeroes for site ID #s. He learned that no issues seem to exist (for existing data sets).
- Inquired about different columns of the data and their values: 'Activity #', 'Retention Due Date', and LSRP acronym.
- He also began to question the quality of the data as related to its dates.

2:44 PM [redacted]

2:44 PM [redacted] uploaded an image: Pasted image at 2015-11-19, 2:44 PM

G000011660 7[redacted] PROPERTY LSR[redacted] 1/24/2012
G000011660 7[redacted] PROPERTY LSR[redacted] 5/7/2012

2:44 PM [redacted] 7323 L[redacted]1 12/19/2009
7323 L[redacted]1 5/7/2012
Oh, it looks like the Activity # is also different

2:47 PM [redacted] yeah that's an internal date that we're not as concerned about

2:48 PM [redacted] Just on simple matching out of the 837 rows, there are only 583 matched. it'll chan
Does that seem reasonable? I.e. that dataset is not wildly off? And do is the idea to try t

2:55 PM [redacted] yeah. maybe see if a lot of the non-matches have dates that are pretty recent?

2:56 PM ☆ seems like this abandoned set isn't something we can use for a final version. DEP will

3:02 PM [redacted] I was way off. It looks like there is about 1361 unique IDs, and only 564 matche
that's not even half
It looks like the dates are all over the place
though mostly from the last 3 or 4 years

3:05 PM [redacted] hm, and neither has leading zeros now, right?

3:12 PM [redacted] They both do in the actual text
Ihave checked a few and don't seem to be a leading zero issue

3:16 PM [redacted] k

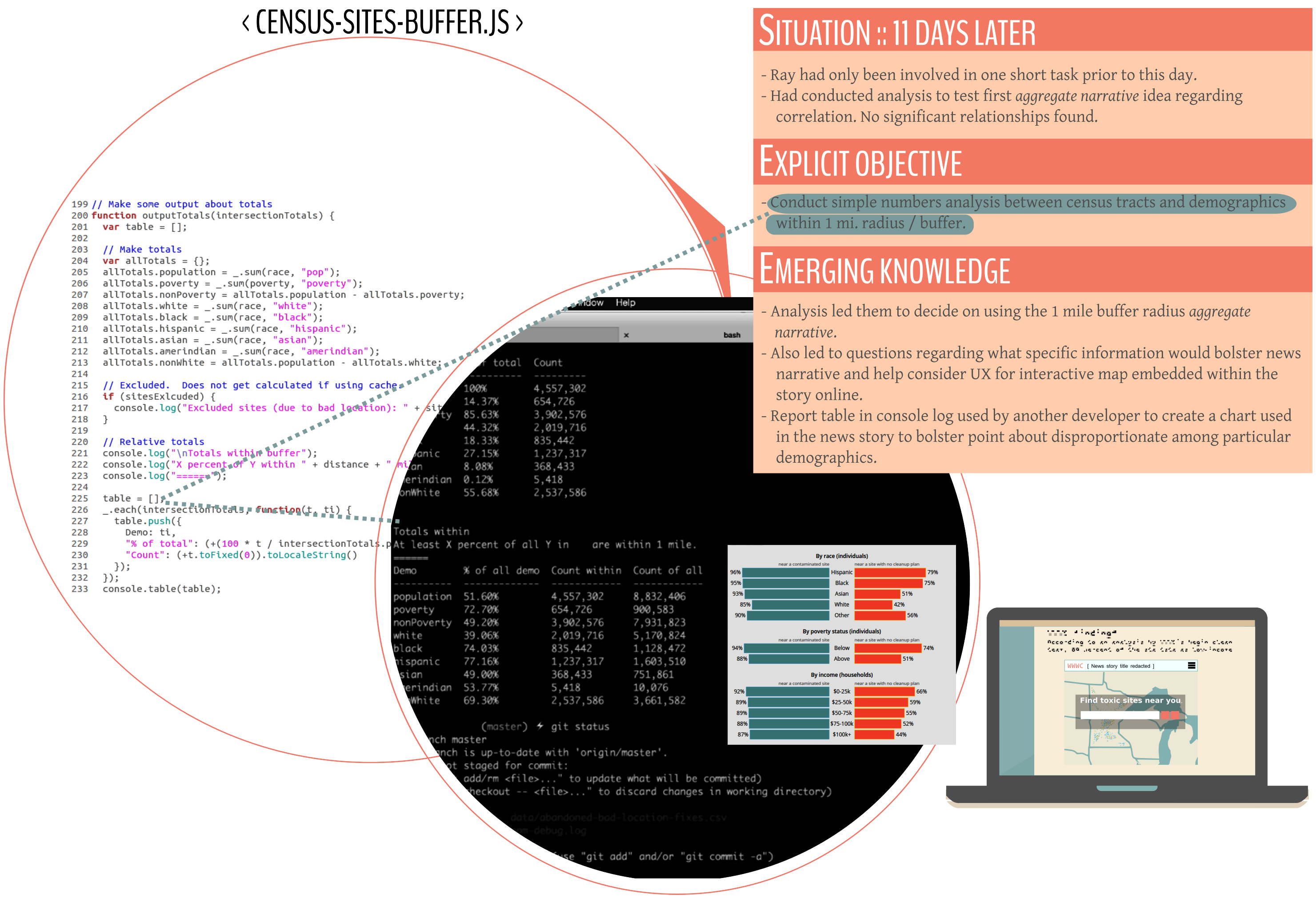
3:28 PM [redacted] let's pause for now, [redacted] says

3:28 PM [redacted] 👍

Figure 4.9: Representation of Ray's coding work to match up files in the list of abandoned toxic sites with the complete list of sites in preparation for their geocoding work.

As shown in Fig. 4.9, from Ray's explicit objective to match location data in the `abandoned-match.js` file, he began to question how the variance in "Program Interest" (PI) ID numbers, as well as how many of the IDs with leading zeros were missing those zeroes, might affect the programmability of the data set. In other words, how accurate can he match sites, or conduct future coding work, if the IDs constitute a problem? Ray also noticed how there may be duplicates within the files. He conferred with Jun about the issue by asking her about the difference in *Activity Number* and *Retention Due Dates*, which are 2 different columns found within the CSV file for each row. Finally, he also began to question the integrity of the data with regards to its age, since the years sometimes went back as far as 2009. All of this meaning-making occurred during a simple matching task.

After he completed this coding task, he reported the number to Rosa on Slack. Ray would not code on this project for another 11 days. In Figure 4.10, Ray had conducted a preliminary correlation test of the toxic site data by using Census tract data earlier that day. He found no significant correlation, so he and Jun hedged the degree to which the data could be tested by conducting a simple-numbers test between the DEP data and the 1-mile buffer radius composed with the Census tracts.



‹ CENSUS-SITES-BUFFER.JS ›

```
// Make some output about totals
function outputTotals(intersectionTotals) {
  var table = [];

  // Make totals
  var allTotals = {};
  allTotals.population = _.sum(race, "pop");
  allTotals.poverty = _.sum(poverty, "poverty");
  allTotals.nonPoverty = allTotals.population - allTotals.poverty;
  allTotals.white = _.sum(race, "white");
  allTotals.black = _.sum(race, "black");
  allTotals.hispanic = _.sum(race, "hispanic");
  allTotals.asian = _.sum(race, "asian");
  allTotals.amerindian = _.sum(race, "amerindian");
  allTotals.nonWhite = allTotals.population - allTotals.white;

  // Excluded.  Does not get calculated if using cache.
  if (sitesExlcuded) {
    console.log("Excluded sites (due to bad location): " + sit
  }

  // Relative totals
  console.log("\nTotals within buffer");
  console.log("X percent of Y within " + distance + " mi
  console.log("======");

  table = [];
  _.each(intersectionTotals, function(t, ti) {
    table.push({
      Demo: ti,
      "% of total": (+(100 * t / intersectionTotals.p
      "Count": (+t.toFixed(0)).toLocaleString()
    });
  });
  console.table(table);
```

```
              f total   Count
              ---------  ---------
              100%       4,557,302
              14.37%     654,726
          rty 85.63%     3,902,576
              44.32%     2,019,716
              18.33%     835,442
          panic 27.15%   1,237,317
          ian  8.08%     368,433
       erindian 0.12%    5,418
       onWhite  55.68%   2,537,586

Totals within
At least X percent of all Y in    are within 1 mile.
======
Demo        % of all demo  Count within  Count of all
----------  -------------  ------------  ------------
population  51.60%         4,557,302     8,832,406
poverty     72.70%         654,726       900,583
nonPoverty  49.20%         3,902,576     7,931,823
white       39.06%         2,019,716     5,170,824
black       74.03%         835,442       1,128,472
hispanic    77.16%         1,237,317     1,603,510
sian        49.00%         368,433       751,861
erindian    53.77%         5,418         10,076
 White      69.30%         2,537,586     3,661,582

          (master) ⚡ git status
 nch master
 nch is up-to-date with 'origin/master'.
 ot staged for commit:
   add/rm <file>..." to update what will be committed)
   heckout -- <file>..." to discard changes in working directory)

        data/abandoned-bad-location-fixes.csv
        pm-debug.log

      use "git add" and/or "git commit -a")
```

Situation :: 11 Days Later

- Ray had only been involved in one short task prior to this day.
- Had conducted analysis to test first *aggregate narrative* idea regarding correlation. No significant relationships found.

Explicit Objective

- Conduct simple numbers analysis between census tracts and demographics within 1 mi. radius / buffer.

Emerging Knowledge

- Analysis led them to decide on using the 1 mile buffer radius *aggregate narrative*.
- Also led to questions regarding what specific information would bolster news narrative and help consider UX for interactive map embedded within the story online.
- Report table in console log used by another developer to create a chart used in the news story to bolster point about disproportionate among particular demographics.

Figure 4.10: Representation of Ray's coding work to combine and create the 1-mile buffer radius defined with Census tracts, and subsequently conduct a single-numbers analysis. Note that the 2 visualizations represented are facsimiles of the originals.

By coding this provisional text, Ray helped the team decide what specific data and computed values to use to develop the story. In particular, the 2 `console.table` slices of the data (lines 220–233 and 235–249) offered a preliminary set of results for the team to continue their work. For example, Phil developed similar charts with the derived tabulated values, which was used in the published report. The preliminary findings also helped the team decide that they can not tell any stories about this issue as a wide-spread trend; particularly not as an interactive map. Instead, they used the single-numbers analysis approach with the 1-mile buffer radius. Namely, they took the average demographic slice within a mile of each toxic site as a means to develop the user experience for the interactive map. They decided to frame the map with a search bar that read, "Find a site near you," so they could personalize the data for their readers. Such a user experience aligns with the single-numbers approach to understand the impact as a current, here-and-now issue, rather than a broad trend closely associated with the minority groups experiencing this issue. In short, the available data could only push their claims about the significance of this issue as it relates to demographic populations so far.

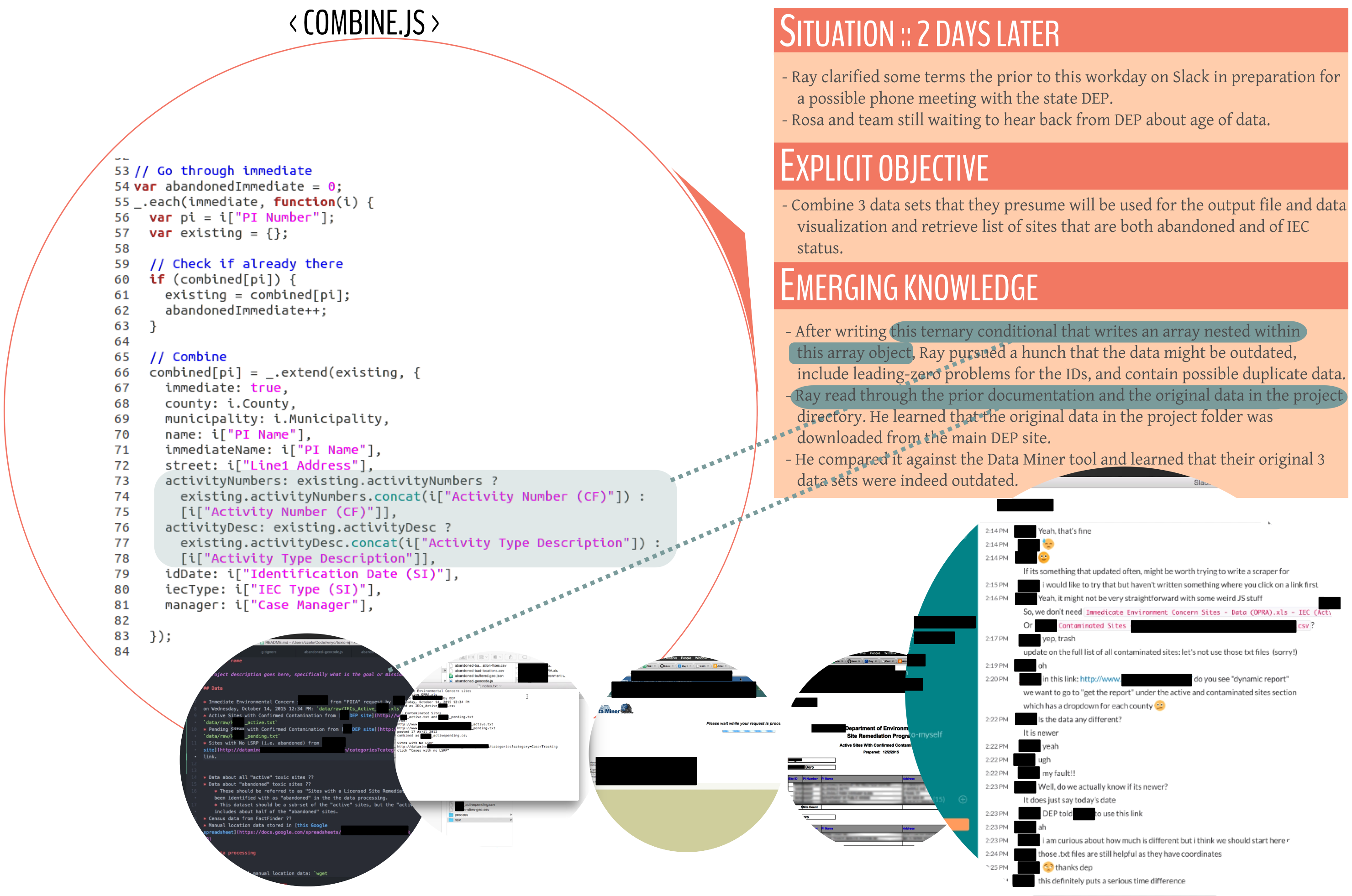
Two days later, Figure 4.11 shows some findings regarding Ray's coding to combine 3 data sets into 1 set, so he could start developing the interactive map. While Ray was coding the data structure within the `combined[pi] = _.extend(existing, {...})` method (lines 65–81), discrepancies began to surface about the dates related to the Activity Numbers and Activity Type inscriptions. As I observed Ray write this object array, I saw him code ternary conditional statements with a `.concat()` JS method (lines 73–78). When I asked him why he wrote those conditional statements, he navigated to the "Immediate Environmental Concerns" CSV file and said:

> **Ray**: So, there's definitely duplicates of sites in this data. For instance, there's, I think, I don't know if these are going to be a good example. But, at least, I know from the other data set, there's a difference in activity numbers, so I think what that means is that they would go out to the site to

> check it out or something like that. And then, so assuming that the activity is still the same, then the type description would go along with that activity number in theory. I don't know. I don't actually know.
>
> Then, there's also, in this specific case you can see that the "Source Status Date" is different. And then this "Source Control Status" is different as well. So we want to put those into arrays as well. Anything that we know that's going to be different should go into arrays. But then there's also this. <*sighs*> the "Source Name" is different, so I don't know if I should make a separate source object: an array of objects.
>
> **Chris**: What's the best relationship to represent what's going on in the original?
>
> **Ray**: Yeah. But also make it so that . . . it's a lot easier if we put it into a CSV. If it's flat, then we can look at it more easily. But that's not that important necessarily. And, I don't even know if this information is important, right? As far as I know, we haven't even looked at it, but I would assume this "Control Status" might be something we want to display.
>
> . . .
>
> If there's any bit of different data, then we have to assume its new data.

After coding this provisional text, Ray also decided to review all of the code, data sets, and available documentation prior to building the map, so he could better understand the project and log the processing and analysis steps. This coding activity with these particular values verified his initial suspicions about the outdated nature of the data. He responded to these varied unknowns about the meaning of the datum by referring to the README file that he had started to write earlier in the project, documenting the origins, processing, and analysis of the data. As he sifted through the original files and code that Jun had organized and created, he found the link to the state department's online database of toxic site information. He compared the data that

Rosa had downloaded straight from the initial DEP website against a query to the "Data Miner" networked database made available on a different page. After discussing Ray's questions about the data more with Jun on Slack, they decided that they needed to integrate this more recently updated data via the Data Miner tool.



‹ COMBINE.JS ›

```
// Go through immediate
var abandonedImmediate = 0;
_.each(immediate, function(i) {
  var pi = i["PI Number"];
  var existing = {};

  // Check if already there
  if (combined[pi]) {
    existing = combined[pi];
    abandonedImmediate++;
  }

  // Combine
  combined[pi] = _.extend(existing, {
    immediate: true,
    county: i.County,
    municipality: i.Municipality,
    name: i["PI Name"],
    immediateName: i["PI Name"],
    street: i["Line1 Address"],
    activityNumbers: existing.activityNumbers ?
      existing.activityNumbers.concat(i["Activity Number (CF)"]) :
      [i["Activity Number (CF)"]],
    activityDesc: existing.activityDesc ?
      existing.activityDesc.concat(i["Activity Type Description"]) :
      [i["Activity Type Description"]],
    idDate: i["Identification Date (SI)"],
    iecType: i["IEC Type (SI)"],
    manager: i["Case Manager"],

  });
```

```
README.md - /Users/zzolo/Code/wnyc/toxic-nj -
.gitignore    abandoned-geocode.js    abando

 name

oject description goes here, specifically what is the goal or missio

## Data

* Immediate Environmental Concern [redacted] from "FOIA" request by [redacted]
on Wednesday, October 14, 2015 12:34 PM: `data/raw/IECs_Active_[redacted].xls`
* Active Sites with Confirmed Contamination from [[redacted] DEP site](http://w
`data/raw/[redacted]_active.txt`
* Pending Sites with Confirmed Contamination from [[redacted] DEP site](http:/
`data/raw/[redacted]_pending.txt`
* Sites with No LSRP (i.e. abandoned) from [redacted]
site](http://datamine[redacted]n/categories?categ
link.


* Data about all "active" toxic sites ??
* Data about "abandoned" toxic sites ??
   * These should be referred to as "Sites with a Licensed Site Remedia
   been identified with as "abandoned" in the the data processing.
   * This dataset should be a sub-set of the "active" sites, but the "acti
   includes about half of the "abandoned" sites.
* Census data from FactFinder ??
* Manual location data stored in [this Google
spreadsheet](https://docs.google.com/spreadsheets/[redacted]
)

ta processing

 manual location data: `wget
```

SITUATION :: 2 DAYS LATER

- Ray clarified some terms the prior to this workday on Slack in preparation for a possible phone meeting with the state DEP.
- Rosa and team still waiting to hear back from DEP about age of data.

EXPLICIT OBJECTIVE

- Combine 3 data sets that they presume will be used for the output file and data visualization and retrieve list of sites that are both abandoned and of IEC status.

EMERGING KNOWLEDGE

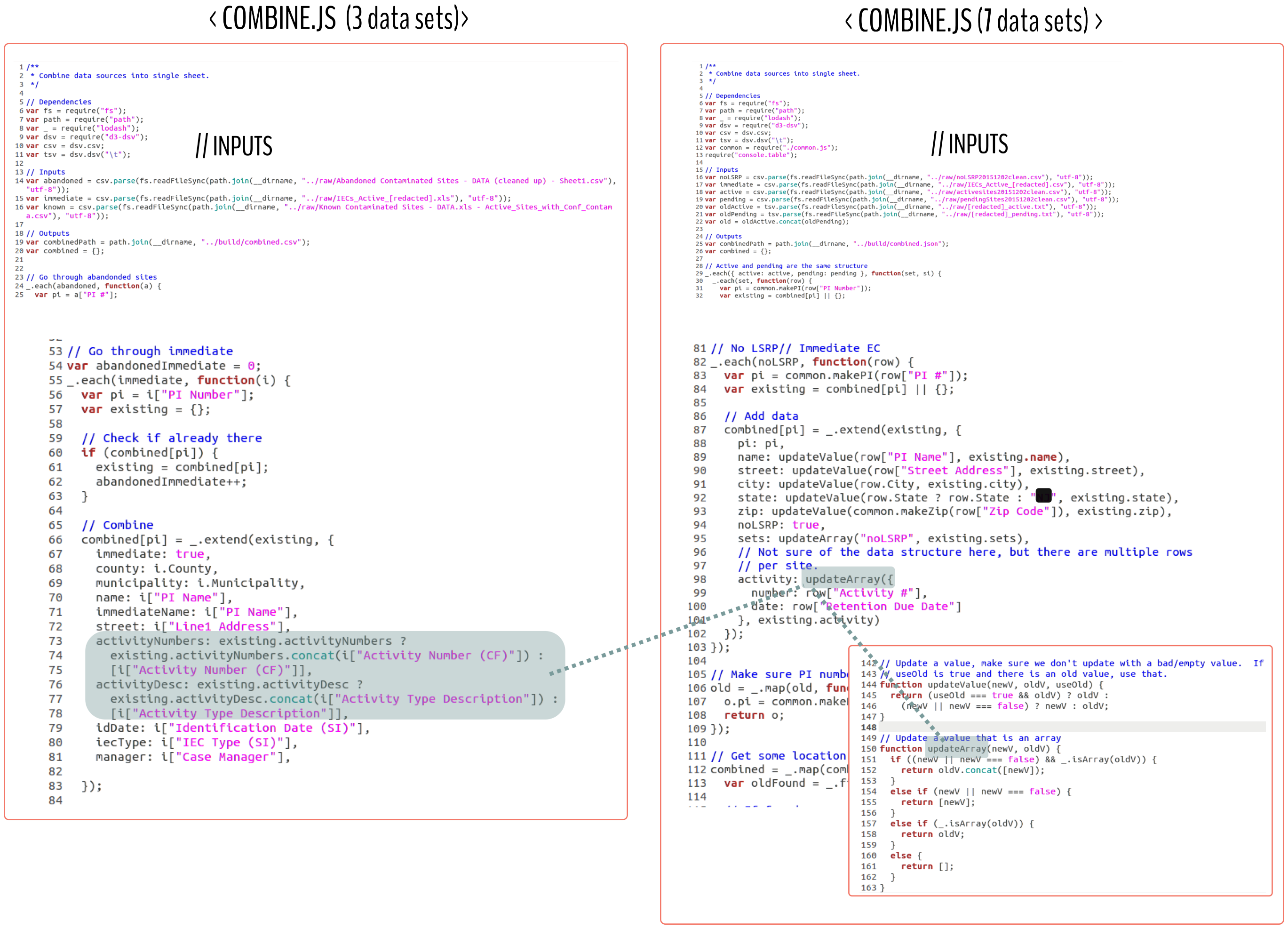
- After writing this ternary conditional that writes an array nested within this array object, Ray pursued a hunch that the data might be outdated, include leading-zero problems for the IDs, and contain possible duplicate data.
- Ray read through the prior documentation and the original data in the project directory. He learned that the original data in the project folder was downloaded from the main DEP site.
- He compared it against the Data Miner tool and learned that their original 3 data sets were indeed outdated.

[redacted]

2:14 PM [redacted] Yeah, that's fine
2:14 PM [redacted] 😓
2:14 PM [redacted] 🙂
If its something that updated often, might be worth trying to write a scraper for
2:15 PM [redacted] i would like to try that but haven't written something where you click on a link first
2:16 PM [redacted] Yeah, it might not be very straightforward with some weird JS stuff [redacted]
So, we don't need `Immedicate Environment Concern Sites - Data (OPRA).xls - IEC (Act`
Or [redacted] `Contaminated Sites [redacted] csv`?
2:17 PM [redacted] yep, trash
update on the full list of all contaminated sites: let's not use those txt files (sorry!)
2:19 PM [redacted] oh
2:20 PM [redacted] in this link: http://www.[redacted] do you see "dynamic report"
we want to go to "get the report" under the active and contaminated sites section
which has a dropdown for each county 😕
2:22 PM [redacted] Is the data any different?
It is newer
2:22 PM [redacted] yeah
2:22 PM [redacted] ugh
2:22 PM [redacted] my fault!!
2:23 PM [redacted] Well, do we actually know if its newer?
It does just say today's date
2:23 PM [redacted] DEP told [redacted] to use this link
2:23 PM [redacted] ah
2:23 PM [redacted] i am curious about how much is different but i think we should start here
2:24 PM [redacted] those .txt files are still helpful as they have coordinates
2:25 PM [redacted] 😔 thanks dep
[redacted] this definitely puts a serious time difference

Figure 4.11: Representation of Ray's coding work to combine the 3 available data sets, which Ray learned were outdated after his writing the ternary conditional statement.

Once the team received current data, Ray revised his code in the `combine.js` file to accommodate 7 total files (see Fig. 4.12). Ray took this revision opportunity with 4 more input files to revise how he handled the changing *Activity Numbers* and *Retention Due Dates* across all of the files. Instead of using a ternary conditional statement within the object array instance of each data point (see the original code, lines 73–78), he created the `updateArray()` function, which handles either datapoint to accept the most current value in the array. Note how he makes a similar revision for other data-types that differ from arrays with the `updateValue` function. New and more data brought about Ray's idea to make these changes to the code, but these changes also represent Ray's re-purposing of the data, since they are only concerned with the current values, which differs from the DEP's tracking the remediation specialists' activities at each site.



```
/**
 * Combine data sources into single sheet.
 */

// Dependencies
var fs = require("fs");
var path = require("path");
var _ = require("lodash");
var dsv = require("d3-dsv");
var csv = dsv.csv;
var tsv = dsv.dsv("\t");

// Inputs
var abandoned = csv.parse(fs.readFileSync(path.join(__dirname, "../raw/Abandoned Contaminated Sites - DATA (cleaned up) - Sheet1.csv"),
"utf-8"));
var immediate = csv.parse(fs.readFileSync(path.join(__dirname, "../raw/IECs_Active_[redacted].xls"), "utf-8"));
var known = csv.parse(fs.readFileSync(path.join(__dirname, "../raw/Known Contaminated Sites - DATA.xls - Active_Sites_with_Conf_Contam
a.csv"), "utf-8"));

// Outputs
var combinedPath = path.join(__dirname, "../build/combined.csv");
var combined = {};


// Go through abandonded sites
_.each(abandoned, function(a) {
  var pi = a["PI #"];
```

```
// Go through immediate
var abandonedImmediate = 0;
_.each(immediate, function(i) {
  var pi = i["PI Number"];
  var existing = {};

  // Check if already there
  if (combined[pi]) {
    existing = combined[pi];
    abandonedImmediate++;
  }

  // Combine
  combined[pi] = _.extend(existing, {
    immediate: true,
    county: i.County,
    municipality: i.Municipality,
    name: i["PI Name"],
    immediateName: i["PI Name"],
    street: i["Line1 Address"],
    activityNumbers: existing.activityNumbers ?
      existing.activityNumbers.concat(i["Activity Number (CF)"]) :
      [i["Activity Number (CF)"]],
    activityDesc: existing.activityDesc ?
      existing.activityDesc.concat(i["Activity Type Description"]) :
      [i["Activity Type Description"]],
    idDate: i["Identification Date (SI)"],
    iecType: i["IEC Type (SI)"],
    manager: i["Case Manager"],

  });

```

```
/**
 * Combine data sources into single sheet.
 */

// Dependencies
var fs = require("fs");
var path = require("path");
var _ = require("lodash");
var dsv = require("d3-dsv");
var csv = dsv.csv;
var tsv = dsv.dsv("\t");
var common = require("./common.js");
require("console.table");

// Inputs
var noLSRP = csv.parse(fs.readFileSync(path.join(__dirname, "../raw/noLSRP20151202clean.csv"), "utf-8"));
var immediate = csv.parse(fs.readFileSync(path.join(__dirname, "../raw/IECs_Active_[redacted].csv"), "utf-8"));
var active = csv.parse(fs.readFileSync(path.join(__dirname, "../raw/activesites20151202clean.csv"), "utf-8"));
var pending = csv.parse(fs.readFileSync(path.join(__dirname, "../raw/pendingSites20151202clean.csv"), "utf-8"));
var oldActive = tsv.parse(fs.readFileSync(path.join(__dirname, "../raw/[redacted]_active.txt"), "utf-8"));
var oldPending = tsv.parse(fs.readFileSync(path.join(__dirname, "../raw/[redacted]_pending.txt"), "utf-8"));
var old = oldActive.concat(oldPending);

// Outputs
var combinedPath = path.join(__dirname, "../build/combined.json");
var combined = {};

// Active and pending are the same structure
_.each({ active: active, pending: pending }, function(set, si) {
  _.each(set, function(row) {
    var pi = common.makePI(row["PI Number"]);
    var existing = combined[pi] || {};
```

```
// No LSRP// Immediate EC
_.each(noLSRP, function(row) {
  var pi = common.makePI(row["PI #"]);
  var existing = combined[pi] || {};

  // Add data
  combined[pi] = _.extend(existing, {
    pi: pi,
    name: updateValue(row["PI Name"], existing.name),
    street: updateValue(row["Street Address"], existing.street),
    city: updateValue(row.City, existing.city),
    state: updateValue(row.State ? row.State : "[redacted]", existing.state),
    zip: updateValue(common.makeZip(row["Zip Code"]), existing.zip),
    noLSRP: true,
    sets: updateArray("noLSRP", existing.sets),
    // Not sure of the data structure here, but there are multiple rows
    // per site.
    activity: updateArray({
      number: row["Activity #"],
      date: row["Retention Due Date"]
    }, existing.activity)
  });
});

// Make sure PI numbe
old = _.map(old, fun
  o.pi = common.make
  return o;
});

// Get some location
combined = _.map(com
  var oldFound = _.f

```

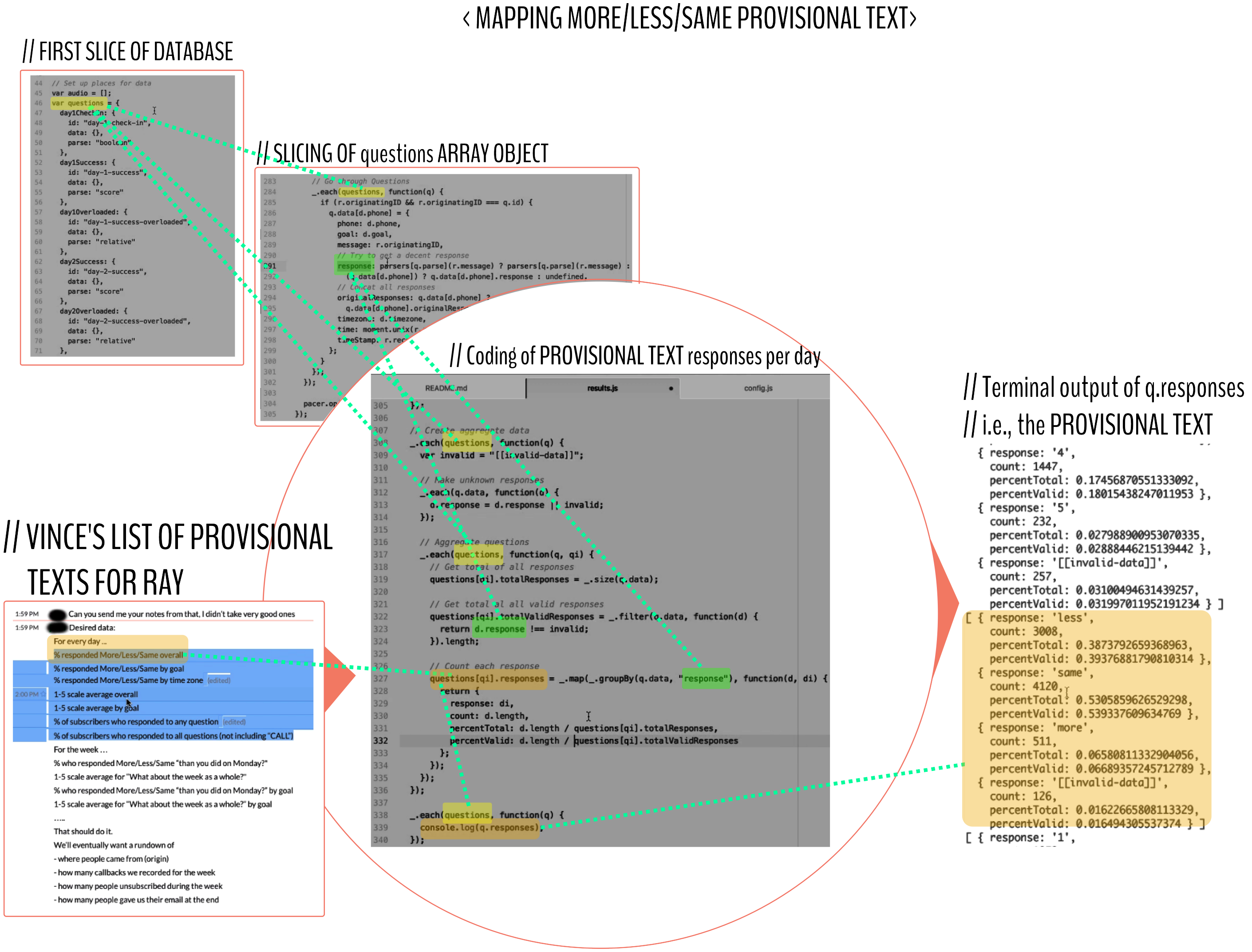
```
// Update a value, make sure we don't update with a bad/empty value.  If
// useOld is true and there is an old value, use that.
function updateValue(newV, oldV, useOld) {
  return (useOld === true && oldV) ? oldV :
    (newV || newV === false) ? newV : oldV;
}

// Update a value that is an array
function updateArray(newV, oldV) {
  if ((newV || newV === false) && _.isArray(oldV)) {
    return oldV.concat([newV]);
  }
  else if (newV || newV === false) {
    return [newV];
  }
  else if (_.isArray(oldV)) {
    return oldV;
  }
  else {
    return [];
  }
}
```

Figure 4.12: Two versions of Ray's `combine.js` code, emphasizing revisions made to his ternary conditional statements after the code now accommodated the combination of 7 data sets, instead of only 3.

Similar epistemic action occurred during the Health Texting project. After the texting campaign had been running for a few days, Ray had a meeting with the podcasting team. In the meeting, the podcast host wanted Ray to dig through the available data to find a possible talking point for the show that they were going to record later that afternoon. At one point during the meeting, the host told the team that they had a hunch about "one niblet of data" – a possible angle to pursue, remarking that "there's a story there."

This "niblet" had to do with one of the Likert Scale responses, where participants responded to a prompt about whether or not they consumed MORE, LESS, or the SAME amount of information as usual. From that discussion, Ray requested notes from the editor, Vince, over Slack. From that list of different slices of the data, Ray deemed the aforementioned "niblet" as priority.

In Figure 4.13 below, I linked the different textual artifacts and elements: list of provisional text ideas created from the meeting, different parts of the `results.js` code that slice up the Health Texting database export, and the `console.log(q.responses)` printed out in the Terminal. I trace the ways Ray slices up the data to produce the provisional text as a Terminal read out. Note how Ray uses the list of ideas to choose particular datapoints to help produce new perspectives of the data through his provisional texts in the Terminal.



‹ MAPPING MORE/LESS/SAME PROVISIONAL TEXT›

// FIRST SLICE OF DATABASE

```
// Set up places for data
var audio = [];
var questions = {
  day1CheckIn: {
    id: "day-1-check-in",
    data: {},
    parse: "boolean"
  },
  day1Success: {
    id: "day-1-success",
    data: {},
    parse: "score"
  },
  day1Overloaded: {
    id: "day-1-success-overloaded",
    data: {},
    parse: "relative"
  },
  day2Success: {
    id: "day-2-success",
    data: {},
    parse: "score"
  },
  day2Overloaded: {
    id: "day-2-success-overloaded",
    data: {},
    parse: "relative"
  },
```

// SLICING OF questions ARRAY OBJECT

```
      // Go through Questions
      _.each(questions, function(q) {
        if (r.originatingID && r.originatingID === q.id) {
          q.data[d.phone] = {
            phone: d.phone,
            goal: d.goal,
            message: r.originatingID,
            // Try to get a decent response
            response: parsers[q.parse](r.message) ? parsers[q.parse](r.message) :
              (q.data[d.phone]) ? q.data[d.phone].response : undefined,
            // Concat all responses
            originalResponses: q.data[d.phone] ?
              q.data[d.phone].originalResp
            timezone: d.timezone,
            time: moment.unix(r
            timeStamp: r.rec
          };
        }
      });
    });

    pacer.op
  });
```

// Coding of PROVISIONAL TEXT responses per day

```
  });

  // Create aggregate data
  _.each(questions, function(q) {
    var invalid = "[[invalid-data]]";

    // Make unknown responses
    _.each(q.data, function(d) {
      d.response = d.response || invalid;
    });

    // Aggregate questions
    _.each(questions, function(q, qi) {
      // Get total of all responses
      questions[qi].totalResponses = _.size(q.data);

      // Get total al all valid responses
      questions[qi].totalValidResponses = _.filter(q.data, function(d) {
        return d.response !== invalid;
      }).length;

      // Count each response
      questions[qi].responses = _.map(_.groupBy(q.data, "response"), function(d, di) {
        return {
          response: di,
          count: d.length,
          percentTotal: d.length / questions[qi].totalResponses,
          percentValid: d.length / questions[qi].totalValidResponses
        };
      });
    });
  });

  _.each(questions, function(q) {
    console.log(q.responses);
  });
```

// Terminal output of q.responses
// i.e., the PROVISIONAL TEXT

```
  { response: '4',
    count: 1447,
    percentTotal: 0.17456870551333092,
    percentValid: 0.18015438247011953 },
  { response: '5',
    count: 232,
    percentTotal: 0.027988900953070335,
    percentValid: 0.02888446215139442 },
  { response: '[[invalid-data]]',
    count: 257,
    percentTotal: 0.03100494631439257,
    percentValid: 0.031997011952191234 } ]
[ { response: 'less',
    count: 3008,
    percentTotal: 0.3873792659368963,
    percentValid: 0.39376881790810314 },
  { response: 'same',
    count: 4120,
    percentTotal: 0.5305859626529298,
    percentValid: 0.539337609634769 },
  { response: 'more',
    count: 511,
    percentTotal: 0.06580811332904056,
    percentValid: 0.06689357245712789 },
  { response: '[[invalid-data]]',
    count: 126,
    percentTotal: 0.01622665808113329,
    percentValid: 0.01649430553737 4 } ]
[ { response: '1',
```

// VINCE'S LIST OF PROVISIONAL TEXTS FOR RAY

1:59 PM [redacted] Can you send me your notes from that, I didn't take very good ones

1:59 PM [redacted] Desired data:

For every day ...
% responded More/Less/Same overall
% responded More/Less/Same by goal
% responded More/Less/Same by time zone (edited)
2:00 PM 1-5 scale average overall
1-5 scale average by goal
% of subscribers who responded to any question (edited)
% of subscribers who responded to all questions (not including "CALL")
For the week …
% who responded More/Less/Same "than you did on Monday?"
1-5 scale average for "What about the week as a whole?"
% who responded More/Less/Same "than you did on Monday?" by goal
1-5 scale average for "What about the week as a whole?" by goal
…..
That should do it.
We'll eventually want a rundown of
- where people came from (origin)
- how many callbacks we recorded for the week
- how many people unsubscribed during the week
- how many people gave us their email at the end

Figure 4.13: Mapping of Ray's coding of the Health Texting "niblet" of `responses` data (green and orange) as sliced from the `questions` slice (yellow).

After producing the logged output of the response data for each day, he said the following during the TAP:

> **Ray**: So `day–1–success` . . . < *hovers mouse over each data point* >
>
> Looking at the data, it looks like it should be right. It looks like 7–8,000 responses. These percentages look right. They're not very 'good' in that they're not necessarily interesting. It looks like 50% of people were the SAME, where we probably want more LESS.
>
> < *scrolls down a bit to* `day–2–success–overloaded` *data* >
>
> We can look at `day–2–success–overloaded` and we get a little higher up < *gesturing over* `less` *and* `more` *percentages*>. We get a little over 40% and a little under 50% for `same` and `less`.
>
> For day 3, we have . . . < *scrolls down to* `day–3–success–overloaded` > Ahhh, that's interesting < *hovers around* `less` *and* `same` *values* > . . . So, now, there's less people who responded, but we have more people who said `less` < *selecting* `less` >
>
> I'm just trying to think about whether or not this data is complete. I'm thinking about when I pulled it. Umm, which it should be. . . . Yeah, it's definitely 12 hours, if not more after the East coast one went out, so there's no reason to think [there's missing data].

In this excerpt, Ray begins to construct a sense for the integrity of his provisional text output. Specifically, he reads through the first 3 days of responses-data, trying to find potential talking points for the podcasting team. He also uses knowledge about how the texting application began at certain times and in certain timezones to ascertain the integrity of the data, i.e., its completeness.

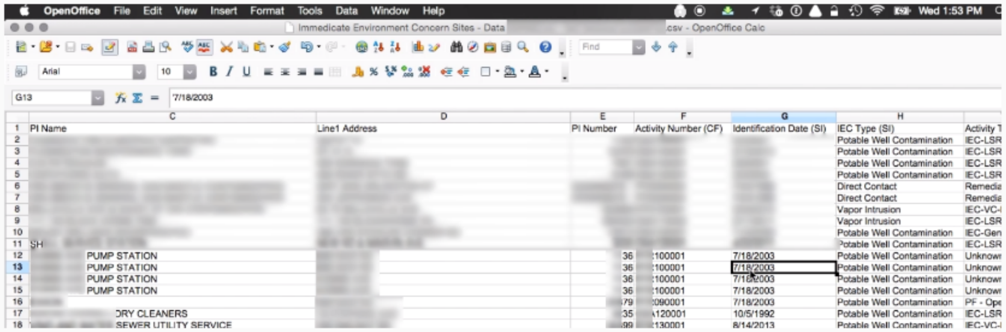
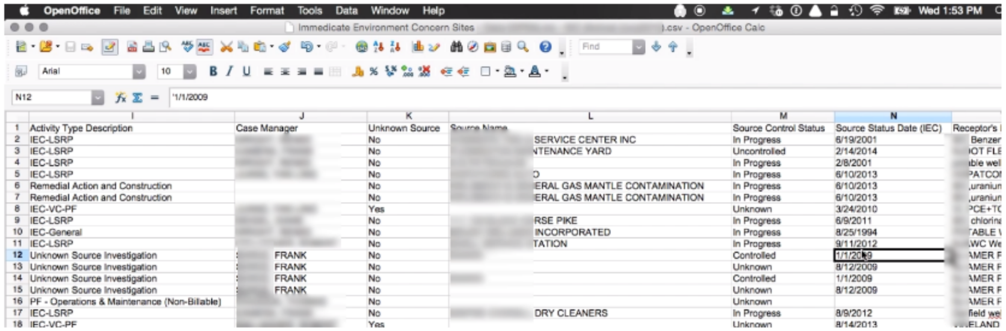
Overall, evidence collected from each of the aforementioned projects indicates 2 major findings. First, the provisional texts serve as texts to support the story-finding work of the team. Second, Ray's coding of the provisional texts involved meaning-making work about the data sets in relationship to the developing aggregate narrative. Evidence of such knowledge and knowledge work in relationship to Ray's coding led to questions about relationship between the tools and semiotic modes to produce such insights.

2. *Semiotic modes and habit of mind*: In this section, I provide a finer-grained description of Ray's coding of representative moments, both of which I reported on in the *Coding as epistemic* above. Specifically, I provide more granular details about Ray's coding of the `combine.js` during the State Toxic Sites project and the *Percentages of participants who responded More/Less/Same overall* during the Health Texting project. These thick descriptions build on the previous findings about the epistemic nature of Ray's coding by describing how Ray's coding incorporated and coordinated semiotic resources and tools through a particular habit of mind.

Ray put this habit of mind as follows during a retrospective account: "What kind of data is it?" In effect, Ray was communicating to me what he often thinks as he reads through data sets. For example, he opens a CSV data set in Calc, while he has his Atom code editor open with a goal-oriented file ready to do something with the CSV. As he reads through the data set in Calc, Ray said that he "More generally, I'm going through each column and saying: "'What kind of data is it?' Is it a boolean? Don't know if I handle dates [during this task]. Then there's the aspect of multiple values." After looking at Ray's coding tasks more closely, I recognized how this technical question about translating data-types from one digital file format to another was evidently tangled up in the meaning-making work to know what the values mean. In effect, Ray's explicit technical question was bound to another general question about understanding the meaning of the data; hence, the habit of mind: "What kind of data is it? [And what does it mean?]" In what follows, I present 2 fine-grained examples that provide evidence for this habit of mind, as Ray codes provisional texts.

During the State Toxic Sites project, Ray wrote a script, `combine.js`, which combined the IEC (Immediate Environmental Concern) data set with the abandoned data set. This script produced a provisional text that enabled him to subsequently perform a more thorough analysis with the Census tract data sets. In what follows, I provide evidence from one of the 2 retrospective accounts that illuminated some of the tacit habits of mind Ray practices while processing data. (See Appendix A.13 for the extended transcript.)

In this situation, Ray had already worked with the team and the project data a few times. I had Ray review a clip from the project, where he started to realize that what he originally thought were duplicate rows of site data were actually different, or changing, site information based on the state case manager and/or remediation specialist's update to the database for the site. Figure 4.14 shows Ray recalling this particular insight as he read through the data set in Calc. This insight about the data led Ray to express any column with changing values as an array within his code, which translated the structure of the CSV file into JS array objects. Interestingly, within the `combine.js` file, Ray translated this data structure differently with the abandoned data set than the IEC data set. In Figure 4.15, I juxtapose Ray's initial code that turns each row's 'flat', 2-dimensional, tabular CSV data structure of the abandoned site data into an array object (lines 27–49) with the code that "extends" the abandoned array object, `existing`, with the IEC CSV data (lines 66–75). Note how Ray wrote 2 distinct conditional statements to handle the changing status information for each site in the abandoned array object, while the he wrote inline ternary conditional statements to handle the same information for the IEC data.

0:50 – 1:12	• Video: Navigates to the IEC data set (CSV file), which is open in Calc spreadsheet program. • "So this is where I discover that there's multiple rows per site. I think that that's what I'd be trying to process there."	
1:12 – 1:25	• Video: Scrolls horizontally across the particular rows of interest to see the other columns. • "But the multiple rows doesn't mean new information; it just means there's different information in each row -- slightly different."	

Immedicate Environment Concern Sites - Data [redacted] .csv - OpenOffice Calc

G13 = 7/18/2003

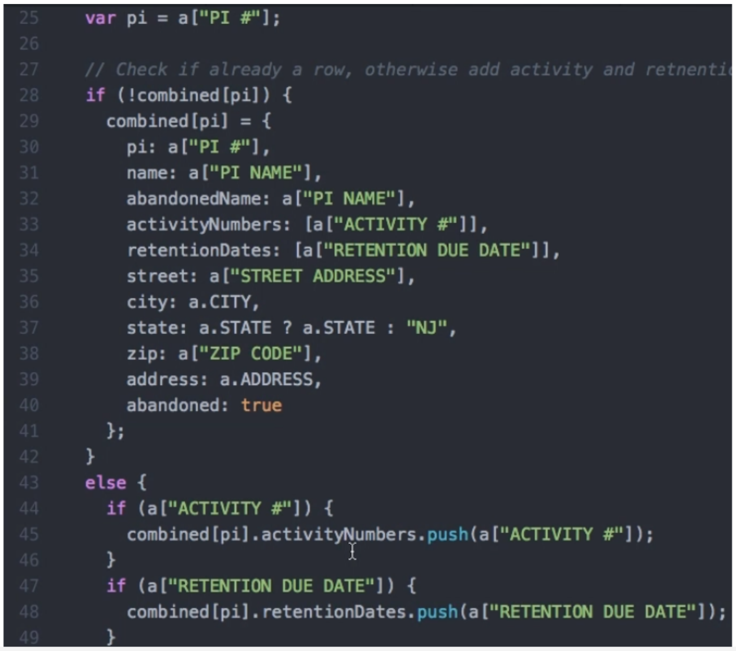
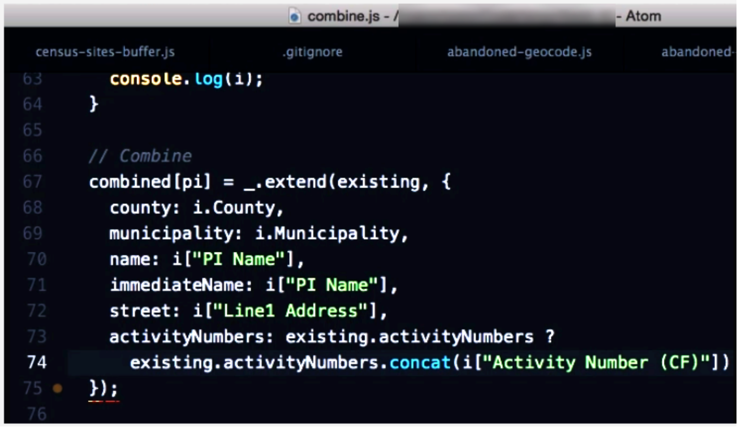
	C	D	E	F	G	H	
1	PI Name	Line1 Address	PI Number	Activity Number (CF)	Identification Date (SI)	IEC Type (SI)	Activity T
2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-LSR
3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-LSR
4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-LSR
5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-LSR
6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Direct Contact	Remedia
7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Direct Contact	Remedia
8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Vapor Intrusion	IEC-VC-I
9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Vapor Intrusion	IEC-LSR
10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-Gen
11	SH [redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-LSR
12	[redacted] PUMP STATION	[redacted]	[redacted]36	[redacted]100001	7/18/2003	Potable Well Contamination	Unknown
13	[redacted] PUMP STATION	[redacted]	[redacted]36	[redacted]100001	7/18/2003	Potable Well Contamination	Unknown
14	[redacted] PUMP STATION	[redacted]	[redacted]36	[redacted]100001	7/18/2003	Potable Well Contamination	Unknown
15	[redacted] PUMP STATION	[redacted]	[redacted]36	[redacted]100001	7/18/2003	Potable Well Contamination	Unknown
16	[redacted]	[redacted]	[redacted]79	[redacted]090001	7/18/2003	Potable Well Contamination	PF - Ope
17	[redacted] DRY CLEANERS	[redacted]	[redacted]35	[redacted]A120001	10/5/1992	Potable Well Contamination	IEC-LSR
18	[redacted] SEWER UTILITY SERVICE	[redacted]	[redacted]99	[redacted]130001	8/14/2013	Potable Well Contamination	IEC-VC-I

Immedicate Environment Concern Sites [redacted]).csv - OpenOffice Calc

N12 = '1/1/2009

	I	J	K	L	M	N	
1	Activity Type Description	Case Manager	Unknown Source	Source Name	Source Control Status	Source Status Date (IEC)	Receptor's I
2	IEC-LSRP	[redacted]	No	[redacted] SERVICE CENTER INC	In Progress	6/19/2001	[redacted] Benzer
3	IEC-LSRP	[redacted]	No	[redacted] NTENANCE YARD	Uncontrolled	2/14/2014	[redacted] OT FLE
4	IEC-LSRP	[redacted]	No	[redacted]	In Progress	2/8/2001	[redacted] ble wel
5	IEC-LSRP	[redacted]	No	[redacted] O	In Progress	6/10/2013	[redacted] PATCO
6	Remedial Action and Construction		No	[redacted] ERAL GAS MANTLE CONTAMINATION	In Progress	6/10/2013	[redacted] ,uraniun
7	Remedial Action and Construction		No	[redacted] ERAL GAS MANTLE CONTAMINATION	In Progress	6/10/2013	[redacted] ,uraniun
8	IEC-VC-PF	[redacted]	Yes		Unknown	3/24/2010	[redacted] PCE+TC
9	IEC-LSRP	[redacted]	No	[redacted] RSE PIKE	In Progress	6/9/2011	[redacted] chlorin
10	IEC-General	[redacted]	No	[redacted] INCORPORATED	In Progress	8/25/1994	[redacted] TABLE \
11	IEC-LSRP	[redacted]	No	[redacted] TATION	In Progress	9/11/2012	[redacted] WC We
12	Unknown Source Investigation	[redacted] FRANK	No	[redacted]	Controlled	1/1/2009	[redacted] AMER F
13	Unknown Source Investigation	[redacted] FRANK	No		Unknown	8/12/2009	[redacted] AMER F
14	Unknown Source Investigation	[redacted] FRANK	No	[redacted]	Controlled	1/1/2009	[redacted] AMER F
15	Unknown Source Investigation	[redacted] FRANK	No		Unknown	8/12/2009	[redacted] AMER F
16	PF - Operations & Maintenance (Non-Billable)	[redacted]	No		Unknown		[redacted] AMER F
17	IEC-LSRP	[redacted]	No	[redacted] DRY CLEANERS	In Progress	8/9/2012	[redacted] field we
18	IEC-VC-PF	[redacted]	Yes		Unknown	8/14/2013	[redacted] ELAND

Figure 4.14: Excerpt from transcription of Ray's retrospective account, reading the IEC data set, recognizing how the duplicate rows are actually site rows with changing or updating statuses.

6:26 – 8:00	• Video: Ray pauses the video. • Ray: I'm curious. [Navigates back to the abandoned data array object.] You know, in reality, it's (Immediate data set ternary conditional) actually the same code as this (abandoned data set conditionals) -- it does the same thing as these lines (43-49). But I don't know why I did it differently. Maybe I changed this later? Well, one of the reasons I probably -- the difference here is that we're creating one object. Here (the conditional) we're doing a little more, err, a little less, err, well we're not creating a big object like this, we're creating 2 smaller things. But either way, they're essentially doing the same thing. I'm not entirely sure why [Navigates back to the Immediate data begin combined with the abandoned.]	
8:00 – 8:30	• Video: Ray looks at the extension of the abandoned data with the Immediate data set. • Ray: I think the main reason I did it this way is because we're doing it inside the object itself. [hovers mouse over code of concern.] And so we can do an if statement around those things.	

```
25  var pi = a["PI #"];
26
27  // Check if already a row, otherwise add activity and retnenti
28  if (!combined[pi]) {
29    combined[pi] = {
30      pi: a["PI #"],
31      name: a["PI NAME"],
32      abandonedName: a["PI NAME"],
33      activityNumbers: [a["ACTIVITY #"]],
34      retentionDates: [a["RETENTION DUE DATE"]],
35      street: a["STREET ADDRESS"],
36      city: a.CITY,
37      state: a.STATE ? a.STATE : "NJ",
38      zip: a["ZIP CODE"],
39      address: a.ADDRESS,
40      abandoned: true
41    };
42  }
43  else {
44    if (a["ACTIVITY #"]) {
45      combined[pi].activityNumbers.push(a["ACTIVITY #"]);
46    }
47    if (a["RETENTION DUE DATE"]) {
48      combined[pi].retentionDates.push(a["RETENTION DUE DATE"]);
49    }
```

combine.js - /[redacted] - Atom

census-sites-buffer.js .gitignore abandoned-geocode.js abandoned-

```
63      console.log(i);
64    }
65
66    // Combine
67    combined[pi] = _.extend(existing, {
68      county: i.County,
69      municipality: i.Municipality,
70      name: i["PI Name"],
71      immediateName: i["PI Name"],
72      street: i["Line1 Address"],
73      activityNumbers: existing.activityNumbers ?
74        existing.activityNumbers.concat(i["Activity Number (CF)"])
75    });
76
```

Figure 4.15: Excerpt from transcription of Ray's retrospective account, reading the IEC data set, recognizing how the duplicate rows are actually site rows with changing or updating statuses.

After I asked Ray a brief prompt about how he came up with the idea to write a ternary conditional statement in this manner, he replied:

> **Ray**: You know, in reality, it's [IEC data set ternary conditional] actually the same code as this [abandoned data set conditionals]. It does the same

> thing as these lines (43-49). But I don't know why I did it differently. Maybe I changed this later?
>
> Well, one of the reasons I probably – the difference here [IEC data ternaries] is that we're creating one object. Here (the conditional) we're doing a little more, err, a little less, err, well we're not creating a big object like this, we're creating 2 smaller things [in the IEC object]. But either way, they're essentially doing the same thing.
>
> I'm not entirely sure why.
>
> <*navigates back to the IEC data being combined with the abandoned*>
>
> <*Ray looks at the extension of the abandoned data with the IEC data set.*>
>
> I think the main reason I did it this way [with the ternary conditionals] is because we're doing it inside the object itself. <*hovers mouse over code of concern*> And so we can do an if statement around those things.

As the retrospective account went on, Ray observed himself toggle between the data set and his Atom code editor. After some time, Ray narrates over the video clip saying:

> **Ray**: Again, I'm just looking for different information for the same site and what those look like. <*watches himself read the data set more in Calc*> Again, just trying to determine which fields [in the CSV data set] are multiple fields – have multiple values. So, I'm looking at the data to visually see it and then coding it to an array, essentially, if it needs to be.
>
> <*switches to Atom and completes the ternary conditional*>
>
> So, I'm looking at the data to visually see it and then coding it to an array, essentially, if it needs to be. More generally, I'm going through each column and saying, 'What kind of data is it?' Is this a boolean? Don't know if I handle dates. Then, there's the aspect of multiple values.
>
> . . .

> ...we're essentially reverse engineering the data.

Ray linked his coding through more technical work, perceiving this data-processing work as analagous to "reverse engineering". Elaborating on this metaphor, he started to speculate about how this data set, which came to them as CSV files, might have been structured originally in the DEP database:

> **Ray**: What's probable is that there's a [Toxic] Site Table and there's multiple activities per site, and that activity, the Activity Type, ... Maybe the Case Manager is for the Activity [Type], so each Activity has a Case Manager for it. It's hard to tell, because the Source [Name] information – I don't think it changes necessarily. But, that could be Source-based for each Activity. We don't know how this fits exactly. We're basically just trying to look and see what makes sense.

As Ray speculated about how the data might be structured in the DEP database, he also acknowledged how this structuring of information is important to 1) help him translate the structure into a more programmable state, while also 2) help him make sense of what the information might mean. Through this retrospective account of this seemingly mundane code to combine 2 data sets, Ray gestures toward the tangled nature of meaning-making with the multiple semiotic modalities of Ray's code in relationship to the data sets. Indeed, the flattened state of the CSV file – exported from what Ray speculates as SQL tables – rendered each site's multiple activities and other column information more difficult to discern and differentiate from being duplicate site information. In sum, *change the structure of the data, change the meaning-making processes.* Furthermore, it's worth highlighting how Ray neglected to refer to this work as reading and writing. Instead. coding and working with data sets was articulated as "visually seeing it" in Calc to then code-as-"reverse-engineer" it.

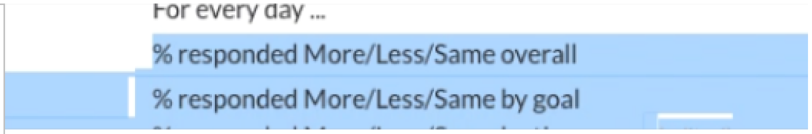
Further still into the retrospective account, Ray was confronted with another set of column data wherein he needed to decide what kind of data-type he needed to translate it into the developing array object within the `combine.js` code. He stated that:

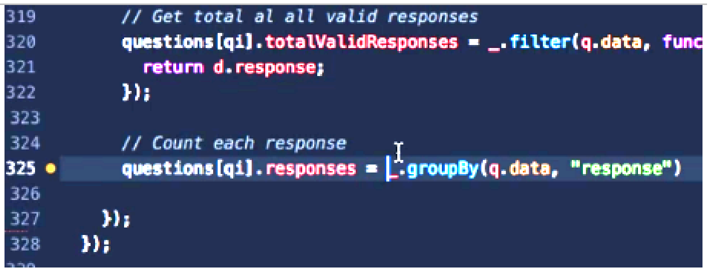
> **Ray**: Now I'm thinking about how I now have those 2 things [2 rows of the same site information, but with different values (see Fig. A.11 in Appendix A.14)] in 2 separate arrays. I was thinking about making an array of object and each object have base fields in it, which is a little more appropriate data structure.

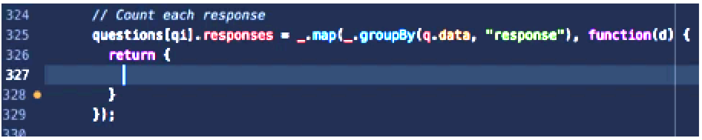
I asked Ray to explain to me what he meant by "appropriate," and he responded:

> **Ray**: Well it's the whole table structure. So if there's a table for a site and that means that each row is a site and that's got the 5 columns about that site. And then there's a column of activities and that's got an ID that goes to a site, and that site might have a row for each activity, but there may be multiple activities per site. But each row and activity can have like 5 fields to it, and so the way you do that with a JSON object is – you'd say your activity field is an array of objects.

Similarly, during the Health Texting project, recall how Ray was tasked to code the provisional text with a "niblet of data" that the podcasting host had a hunch would proffer something of interest to listeners. During this coding task, Ray used a similar coordination of tools, but instead of using Calc to read the data, he used `console.logs()` to render the current state of the data readable. In Figure 4.16, Ray begins his slicing work by figuring out how to take an instance of the `questions`

@6:20	- 6:33: Hovers mouse around the list and says that “Now, we want to count up each response, … ”	 For every day ... % responded More/Less/Same overall % responded More/Less/Same by goal

@6:50	- copy/pastes 'question[qi]
	- “So we can groupBy() the data [q.data] and we want 'response'. … So that will group all the rows into the response that we have.”
		
		`// Get total al all valid responses`
		`questions[qi].totalValidResponses = _.filter(q.data, func`
		`return d.response;`
		`});`
		
		`// Count each response`
		`questions[qi].responses = _.groupBy(q.data, "response")`
		
		`});`
		`});`

@7:20	- “And that [“response”] will be an object of arrays. And we want to change that into something more succinct.”	
		`// Count each response`
		`questions[qi].responses = _.map(_.groupBy(q.data, "response"), function(d) {`
		`return {`
		
		`}`
		`});`

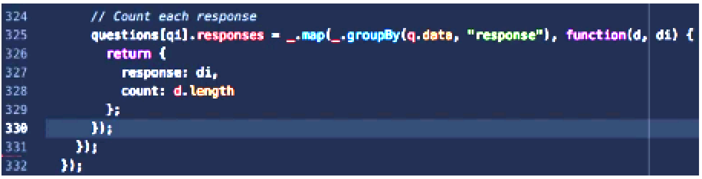
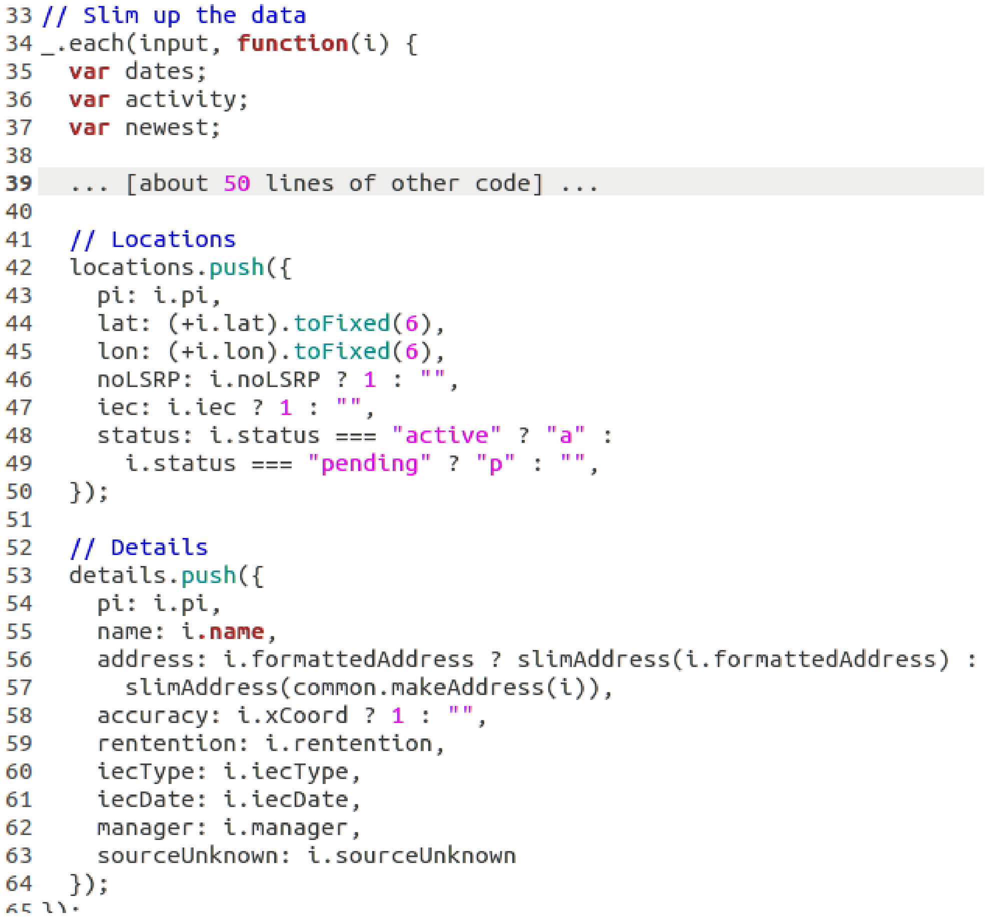
@7:40	- “So we want a “response,” which will end up being the key, and then we want the count, which will be a fact.”	
		`// Count each response`
		`questions[qi].responses = _.map(_.groupBy(q.data, "response"), function(d, di) {`
		`return {`
		`response: di,`
		`count: d.length`
		`};`
		`});`
		`});`
		`});`

Figure 4.16: Excerpt from transcription of Ray’s think-aloud, coding a provisional text for the podcasting team about the Same/Less/More responses per day.

In this case, coding this slice of the data contextualizes the data from its more granular aggregate export from the database. The podcasting team provided Ray with an angle of interest, which gave Ray the exigence and goal to code this provisional text. In short, Ray repurposed the data through his coding work, and it resulted in new data values to make sense of the aggregate narrative about participant responses to the prompts.

During the State Toxic Sites project, the results of similar narrative work can be seen when comparing the data that Ray's `combine.js` code maintained against the `output.js` code. While the combine.js code maintains all of the activities for each site within their respective arrays, the `output.js` code filters out such information save for the most current status. Based on the most current status of the site, Ray's code inscribes the site as either `p` for pending state-hired remediation specialist or `a` for active site with a state-hired remediation specialist. (See lines 48–49 in Fig. 4.17 below.)



```
// Slim up the data
_.each(input, function(i) {
  var dates;
  var activity;
  var newest;

  ... [about 50 lines of other code] ...

  // Locations
  locations.push({
    pi: i.pi,
    lat: (+i.lat).toFixed(6),
    lon: (+i.lon).toFixed(6),
    noLSRP: i.noLSRP ? 1 : "",
    iec: i.iec ? 1 : "",
    status: i.status === "active" ? "a" :
      i.status === "pending" ? "p" : "",
  });

  // Details
  details.push({
    pi: i.pi,
    name: i.name,
    address: i.formattedAddress ? slimAddress(i.formattedAddress) :
      slimAddress(common.makeAddress(i)),
    accuracy: i.xCoord ? 1 : "",
    rentention: i.rentention,
    iecType: i.iecType,
    iecDate: i.iecDate,
    manager: i.manager,
    sourceUnknown: i.sourceUnknown
  });
});
```

Figure 4.17: Excerpt from Ray's `output.js` code, which filters out much of the maintained status updates across all earlier code files, since the aggregate narrative only needs the most current value.

4.4 Discussion

Throughout this chapter, I have examined Ray's coding through an Writing Studies epistemology stemming from the materialities of writing. In this case, I implicated Ray's coding in a question shared by past writing researchers: *What's in a list?* By examining Ray's coding of provisional texts, I have pulled together more granular data to show

how Ray has developed a particular habit of mind in relationship to the semiotic modes of the different representations of data in this domain. Through this thick description, new findings were constructed about understanding Ray's coding as writing.

4.4.1 Coding as ways of figuring data: Habits of mind and semiotic resources

Ray's coding of provisional texts were saturated with semiotic resources: digital file formats, data structures and types, and computational methods and functions. These technical modalities seem to be more salient in Ray's mind, since he sees himself as the technical person on the team and even referred to his data processing and analysis work as "reverse engineering of the data." However, findings indicate that the textual semiotics of Ray's coding with the aforementioned modalities are tangled up with the complex organization of inscriptions about social-material phenomena in tandem with their original contexts of activity some times far removed from the data team.

This coding work with data sets and their structures seemed to be coordinated by Ray's habit of mind, which he initially linked to the technical question of "What kind of data is it?" and later noted this habit of mind was linked to making sense of the data in relationship to the reporting goals. Such a habit of mind shifts the focus away from coding as merely a technical endeavor. Findings show how Ray's coding included the creative epistemic work to figure ways with data, writing goal-oriented per instance slices of the original data sets. As a result, a `.each()`, `.map()`, or `.extend()` method is not simply part and parcel of a utility bag of programming language features. Instead, understanding coding as writing puts such modalities of any programming language in conversation with the conceptual work and objectives within one's domain. The structure of data sets and the code that Ray writes in JS become chained with the goals to find stories in the data. His coding acts often translate the structure of flat CSV files into multidimensional JS array objects, or reconfigure copious database exports into refined aggregate perspectives of the data, which yield potentially new or interesting stories or talking points.

4.4.2 Coding tools as durable writing environments

Ray's habit of mind and its connectedness to semiotic resources seems to be made possible through the coordination of particular technologies: his code editor, terminal, Calc, and Slack. The coordination of these technologies only emphasizes the highly textual nature of his coding work, since each environment makes it possible for him to write, read, or share information. Hutchins (1995) might call Ray's coordination of coding technologies and people as "the *propagation of representational state* across a series of *representational media*" (emphasis original, p. 117). Writing researchers concerned with the materialities of writing should identify how Ray's coding tools and the accompanying texts used and produced provided the durable grounds for him to express his ideas through material language. However, my findings show how Ray's use of the JS programming language and its material form on the screen renders digital information and what the code does with it in recognizable and readable textual formats; that is, computational utterances that can compute and organize large swathes of data.

4.4.3 Developing a data sense: Historical knowledge of computational modalities

Interestingly, when asked to reflect more about a particular mundane task to combine 2 data sets, Ray began to artfully speculate the potential structure of the Toxic Sites data in some estranged state government SQL database. Ray's creative speculation of data sets and their values illuminates how the meaning of such datum exceed their signs, as well as how the struggle for meaning of such values within these data-as-texts takes many forms with the available means of any developer. Such embedded, or even invisible, semiotics of data are akin to Witte's (1992) list-writer, who envisioned the spatial layout of the store, when writing her grocery list in the order by which she would gather the items.

4.4.4 Coding as contextualizing data

Findings from Ray's coding of provisional texts illuminated that such work supported an extended series of tasks to refine story angles through new slices of data. Such slices resulted in outputs that re-purposed the the data toward the stories reporters want to tell. Indeed, Ray's coding provided new ways of interacting with data sets about some everyday matter at aggregate perspectives, wherein the resulting aggregate narratives closely knitted together computation and narrative.

Chapter 5

Conclusions, Implications, and the Future of Writing Studies of Coding

5.1 Summary of Findings

In this dissertation, I set out to learn about coding as writing by exploring the consequences of making language material and computational in a digital medium. I have done so by providing a thick description of Ray's coding domain with the goals to learn more about coding as a textual, writing practice. This thick-description research enabled me to begin to distinguish the nuances between the ways Ray expressed aggregate information with a dynamic range of semiotic resources through his coding acts. Indeed, akin to Geertz's (1973) blinking versus winking, a closer examination of Ray's *in situ* coding helped me recognize, for instance, the subtle difference between Ray's reasoning to use a if/else conditional statement versus a ternary.

Ray's coding on a data team meant figuring and configuring data with the broader objective to find stories with and in the data. I call this pursuit to derive stories from large sets of aggregate information *aggregate narratives*, since the discourse and coding activity surrounding it meant inventing new *angles* surrounding the data, conceptually understanding how the data provides various *perspectives*, and seeing how those perspectives can be *sliced* up to eventually arrive at the boundaries and *degrees* of what the team can say about a particular social phenomena of interest.

Ray's coding and sharing of *provisional texts*, slices of the aggregate data, fueled much of this epistemic, meaning-making work to produce aggregate narratives. From a finer-grained analysis, findings indicated how Ray's coding renders digital information as textual media—a feat supported by his coordination of different writing environments: Atom coding editor, terminal, OpenOffice Calc, and Slack messaging. Ray, with these tools, conducted his coding of provisional texts through a particular *habit of mind* that blended questions about the type of data with the meaning of that aggregate information. Ray's habit of mind and coordination of his coding tools also was linked to his *data sense*, or how information may have been structured prior to its current format, so he could make the best decisions about how to translate that information into the data structures within his project code. Such coding acts to translate data from one context to another proved to help Ray develop knowledge about the meaning of the data in relationship to its technical structure. Over the course of the projects, it also enabled him to support the team work to find and tell stories with the data, repurposing information in lieu of the broader objective to produce aggregate narratives.

In what follows, I synthesize the above summarized findings with the general propositions regarding the materialities of writing.

5.1.1 Writing represents reality and mediates it

Previous studies of writing (Haas, 1996; Scribner & Cole, 1981; Witte, 1992, Wickman, 2010) have shown writing's dyadic function in our everyday lives. The texts that people produce represent reality, but also mediates the course a person or persons might take to accomplish, interpret, or decide what their next steps may be. Ray's provisional texts, and his coding of them, gave new and creative shape to aggregate information that represented a range of social phenomena: state toxic sites and their clean-up activities, the tracking of how much the city pays its employees, how podcast subscribers adhere or not to the activity to consume less information for a week, the status of the city's program to help people rebuild their homes after a natural disaster, and more.

5.1.2 Mediational means shape writing activity

Each of these representations of digital media are engaged with as texts and are also produced and sliced into more meaningful texts through coding. Such textual representations and their pliability is made possible through coding tools. In Ray's case of coding with data sets, he created a durable writing environment that supports the translation and interpretive work to quickly read and write data into usable states. Ray's use of coding tools and the JS programming language enabled him to draw upon a rich range of semiotic resources to organize particular perspectives of inscriptions in the form of textual media.

5.1.3 Writing exceeds the linguistic sign and is multimodal

As Vee (2017) argues, code exceeds writing as much as it is connected to it. The grammatization of computational modalities implicates people in expressions and interpretations of linguistic signs to perform digital operations. Findings from this study show how code, like other forms of writing, does not necessarily carry the whole of an author's signature, intent, or history of contexts and situations. Ray's coding mediated and was mediated by data sets and structured data, as well as the operations performed on such data. Ray's habit of mind yielded a richer picture about how semiotic modalities stem from a person's historical experiences with prior, similar, or connected literacy acts. In this case-study, Ray discussed how he has developed a conceptual set of semiotic modes in relationship to file formats and data structures, and their relationship with databases and programming languages. For instance, rows with seemingly duplicate toxic sites are actually status updates of the same site and the activity to remediate it. Here, the CSVs' flattened rows with different column information from the same site ID could have possibly been distinct tables with hierarchical relationships and rows to track particular case manager status updates. These insights, even if merely speculative on Ray's part, convey the invisible modalities that shaped Ray's new structure of the data as a JS array object in his own code and how to handle it, maintain it, and eventually repurpose it for the interactive map. Such findings invite more questions about how people internalize and combine these semiotic resources; namely, how these modalities become what Wickman (2010) refers to as "culturally recognizable channels" (p. 288, fn. 4) by which developer communicate their ways with data.

5.1.4 Writing is epistemic: Meaning-making is an active process

Previous studies of reading (Haas & Flower, 1988; Haswell et al., 1999) and writing (Pigg, 2014a; Wickman, 2010) indicate the active production of knowledge linked to both literacy activities. In Ray's domain, reading both the data sets and the sliced provisional texts facilitated the team's epistemic work to refine angles to stories. For example, during the City Payroll project, Ray produce numerous slices of the data, sharing them with his team on Slack, which instigated new lines of inquiry and continued coding work to produce provisional texts. Ray's own reading of data sets in relationship to his coding to process and analyze data incorporated contextual details not necessarily self-evident from the texts themselves.

5.1.5 Writing is an individual-social activity

This dissertation constitutes a case-study about Ray and his coding on a data team. Despite the fact that my method focuses on Ray's perspective of the projects discussed throughout, findings show how his coding work as an individual was socially organized by the people, tools, and objectives of the domain. Recall how Vince managed the team through the broader objective to find and tell compelling stories with and through data. Ray, who has a history in more traditional software development domains now found his coding shaped by these objectives and the people on his team. Whether through the sharing and feedback solicited through the provisional texts, or through his actions to navigate other peoples' project code, directory organization, and data-set files, Ray's coding work was embedded within the particular social activity to research and tell news stories.

5.1.6 Writers imbue texts with values, appeals, and knowledge about the world

Writers develop a sense of the social and accrue knowledge about what certain claims and combinations of them have appeal or not. (Brock, 2016) and (Brock & Kelly, 2015) code and how it appeals differently to audiences based on different situations. Ray's coding of provisional texts was tangled up in the angles of interest, i.e., inquiries of interest, that his colleagues sensed worth pursuing and telling to a wider audience. Provisional texts that Ray coded and either shared or read himself in earnest embodies the human endeavor to integrate knowledge of the social with the developing angles linked to data sets. Ray's coding and his production of provisional texts define the boundaries of claims that can be generated about the emerging narrative. These kinds of decisions use available information and semiotic resources to define the semantics and characteristics of a particular thing or activity in and of the world. For example, Ray's coding work included asking and taking action with regards to questions, such as the following: How can code help develop and represent a metric to define the performance of city bike stations?; How does the integrity of the data shape the kinds of analysis the team can perform?; Can 2 aggregates be processed and combined in some way to then test for trends about train performance? About a particular demographic's proximity to a toxic site?; etc. Overall, Ray's coding was implicated in adept opinions about what and how data can be used to tell stories.

5.1.7 Coding is a technical *and culturally situated* writing activity

C. V. Lopes (2014b) illuminates the human creativity and histories behind computational modalities in her treatment of code styles. Lopes' focus on technical constraints as the source of coding styles also conjures up the rich conceptual histories of code and its abstractions: stacks, heaps, flows of control, etc. Findings from my study of Ray's coding shows the depth of tacit knowledge that Ray draws upon as he figures data. As he codes with data sets, he must translate the current state of the data into a new structure that is more usable and programmable within his particular context.

I argue that studying the materialities of writing is to study the ways people blend the external and internal through their literacy acts. It's the research goal to find new ways to understand how and why our relationship with texts and written communication is so richly and frustratingly complex and expressive; how the process of abstraction, when writing, is always linked to a person's active process to draw from their past social-material experiences; or how the structure and content of texts mediate different people differently. In this case-study, I have started to develop findings that gesture toward how Ray, a developer in a newsroom, used the semiotic resources of digital media as texts, and how his coding facilitated the meaning-making of the social phenomena in question in tandem with his internalization of the sometimes very invisible and internalized semiotic modes of how digital information lives in computing technologies.

5.2 Limitations

This dissertation serves as a beginning, rather than an end. By the writing of these particular concluding thoughts, I had finally arrived at a better sense about how to carve out particular units of analysis, conduct particular analytical coding strategies, develop more nuanced ways to report my findings, as well as begin to integrate them into the a substantive theory of coding and its materialities as writing. However, this thick description herein does not amount to generalizations. I do not claim, for instance, that the reported modalities remained salient properties of the data sets and code in all of Ray's observed and reported coding acts. Additionally, the findings reported in Chapter 4 remain preliminary, since more passes through the data will enable me to test these initial conceptualizations of habits of mind against the other TAPs and observational data. Despite these limitations, such constraints offer researchers new questions about the knowledge work of coding. For example, what are the relationships between time and situations with the kinds of knowledge that coding with data sets affords people to potentially construct? Yet another way to examine Ray's habit of mind in relationship to the the coding of provisional texts is to define smaller units of analysis. In so doing, I can conduct a closer inspection of changes made over the course of the project. Overall, the theory of coding as writing that I report and integrate into Writing Studies is substantive and generative, rather than general and universal.

5.3 Pedagogical Applications

Understanding coding as a writing is to understand that it should also not be taught nor learned as some generalizable skill detached from its contexts of practice. If such a claim about coding is taken seriously, curricula should also begin to reflect and act upon this insight. What if coding was scaffolded across the university in similar ways as Writing Across the Curriculum or Writing-Enriched Curricula? What if university writing centers trained and hired different types of developers to support and enrich the kinds of coding already being practiced across domains?

More specific to the domain studied herein, insights about provisional texts could be used in a variety of ways to help educate reporters to work with data sets: To slice them, hone their angles, ask better questions of them, spot issues with the data structures and values, and also more quickly become attune to data and its potential perspectives and aggregate narratives. For example, teachers could assign students to use Open Data Portals, peruse it for data, develop database queries, and subsequently refine angles by slicing it and creating provisional texts for others to chime in on.

Many "learn to code" (K. Brooks & Lindgren, 2015) initiatives focus on teaching code and its logical patterns under the scheme of developing a generalizable notion of "computational thinking" (n.a., 2017a). Computational thinking movements in educational domains argue that computing practices involve particular sets of problem-solving, cognitive skills that can be applied across the disciplines. Literacy scholars might recognize such claims as akin to Great Leap theories of literacy for our current cultural moment.

This case-study of coding as writing shows how data and its dynamic materialities mediate and permeate much of Ray's coding work at WWWC. Ray's coding activity on a data-driven news team involves a vastly different set of knowledge, skills, and tools than his other prior professional work as, for instance, a database administrator for a university. Code, the language used, and its performative logics carry a history of ways of working with and manipulating data toward particular goals and objectives. Broader social histories connected to a languages method(s) for representing and using data in particular ways become linked to one's localized objectives and technologies used by people and their communities. This more culturally situated view of coding and its mediated relationship with data may help temper this brand of coding educational movements as diverse as the people and contexts practicing coding.

5.4 Implications

5.4.1 For Data Science domains

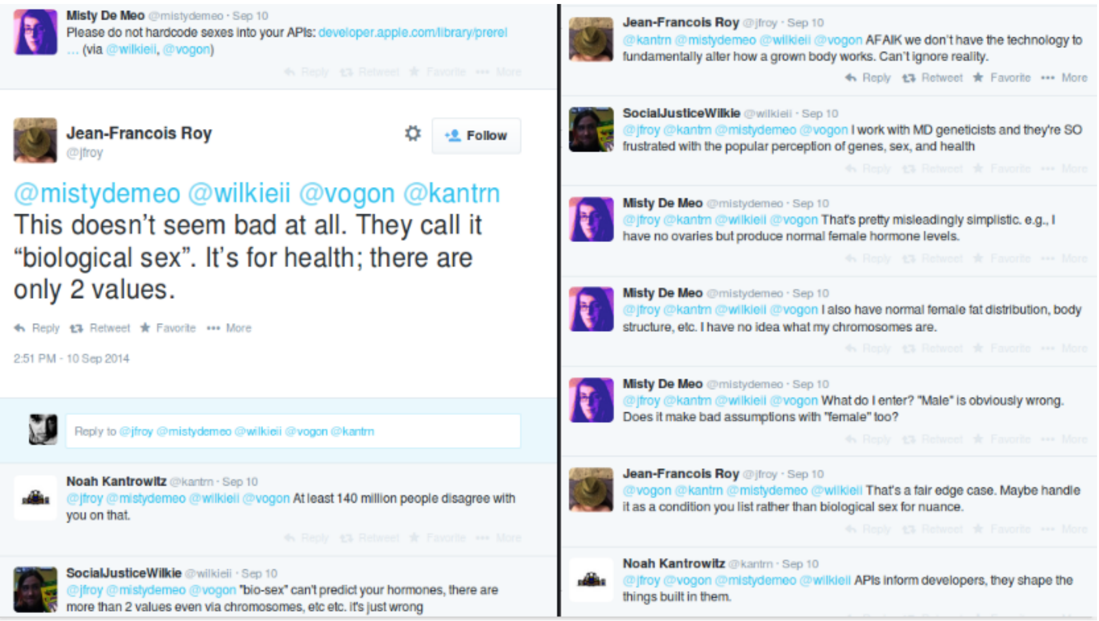
Historian Mazzotti (2017), social scientist Tufekci (2015), and mathematician turned data analyst (O'Neil, 2016) respectively discuss how the concept of algorithm has shifted from a set of instructions to solve a particular problem to become synonymous with whole of a program running on a machine and new forms of positivism. They all argue that numerous discourse communities unfortunately understand platforms as bringers of objectivity, reproducibility, and reliability to solve their complex problems. Each ask for closer examinations about the ways algorithms are opaque, riddled with human biases, and sometimes produce disastrous consequences. In this dissertation, I offer the basic tenets of a methodology within studies of the materiality of writing as a way to illuminate the often effaced rhetorical and cultural dimensions of database design and coding work. I contend that such a methodology can offer generative ways to study how the rhetorics of databases that make up algorithms-as-platforms can be understood as texts – a contestation of aims – which require the dynamic work of people to contextualize information for their goals and audiences. Such a field of research build on the broader research question introduced by this dissertation: What can be learned about platforms, if rhetoricians examine the consequences of making language material and computational in a digital medium?

Furthermore, when considering the consequences of the latest move by the FCC to deregulate and grant Internet Service Providers the ability to share and commodify customers' internet behavior (Finley, 2017; Shepardson, 2017), how does this omnibus of data about our everyday lives get coded into other kinds of data to produce new purposes, angles, and very private stories about our behaviors? How does casting a direct light on the human involvement and invention involved in coding with such information make these institutional power-plays and postures bound to ethical oversight, if not downright refusal to entertain such flippant policies.

5.4.2 For Software Engineering domains

If reading and writing code is not a purely technical skill, and if coding as understood as a writing practice involves the work to use particular uses of language and tools to develop a set of shared, communal knowledge, then how does understanding the materialities of coding as writing with data help software engineering professionals? Ray's habit of mind, which he coordinated in sync with his coding of provisional texts, may provide one avenue for writing and rhetoric researchers concerned with how code represents and performs in and across domains. For example, with the increase in networked and sensor-based computing devices, more data are being collected and used by businesses in charge of software development teams writing code to represent people: their gender, sex, sexuality, bodily health, speech patterns, and other everyday behaviors. If developers do not challenge their typified notions of representing people in relationship to some activity in more nuanced ways, we as a society risk doing unnecessary harm to each other.

Take for instance Apple's programming language, Swift. When Apple originally released its *HealthKit* Application Programming Interface (2014), which would enable developers to develop applications for the iPhone, it hardcoded biological sexes as a binary: Male, Female, or NotSet. In Fig. 5.1 below, some developers discuss the assumptions that Apple developers made about people and their bodies by defining sex in such a reductive way (De Meo, 2014).



Misty De Meo @mistydemeo · Sep 10
Please do not hardcode sexes into your APIs: developer.apple.com/library/prerel … (via @wilkieii, @vogon)

Jean-Francois Roy
@jfroy

@mistydemeo @wilkieii @vogon @kantrn This doesn't seem bad at all. They call it "biological sex". It's for health; there are only 2 values.

2:51 PM - 10 Sep 2014

Reply to @jfroy @mistydemeo @wilkieii @vogon @kantrn

Noah Kantrowitz @kantrn · Sep 10
@jfroy @mistydemeo @wilkieii @vogon At least 140 million people disagree with you on that.

SocialJusticeWilkie @wilkieii · Sep 10
@jfroy @mistydemeo @vogon "bio-sex" can't predict your hormones, there are more than 2 values even via chromosomes, etc etc. it's just wrong

Jean-Francois Roy @jfroy · Sep 10
@kantrn @mistydemeo @wilkieii @vogon AFAIK we don't have the technology to fundamentally alter how a grown body works. Can't ignore reality.

SocialJusticeWilkie @wilkieii · Sep 10
@jfroy @kantrn @mistydemeo @vogon I work with MD geneticists and they're SO frustrated with the popular perception of genes, sex, and health

Misty De Meo @mistydemeo · Sep 10
@jfroy @kantrn @wilkieii @vogon That's pretty misleadingly simplistic. e.g., I have no ovaries but produce normal female hormone levels.

Misty De Meo @mistydemeo · Sep 10
@jfroy @kantrn @wilkieii @vogon I also have normal female fat distribution, body structure, etc. I have no idea what my chromosomes are.

Misty De Meo @mistydemeo · Sep 10
@jfroy @kantrn @wilkieii @vogon What do I enter? "Male" is obviously wrong. Does it make bad assumptions with "female" too?

Jean-Francois Roy @jfroy · Sep 10
@vogon @kantrn @mistydemeo @wilkieii That's a fair edge case. Maybe handle it as a condition you list rather than biological sex for nuance.

Noah Kantrowitz @kantrn · Sep 10
@jfroy @vogon @mistydemeo @wilkieii APIs inform developers, they shape the things built in them.

Figure 5.1: A developer from Google (Roy) disagrees with another developer (De Meo) on Twitter about the consequences of "hardcoding sexes" in Apple's iOS API.

From this one example, I mean to simply convey how commonplace such representations of complex things across software development communities. Where sex and gender are easily conflated and otherwise reduced to produce drastic consequences for people who do not conform to heteronormative narratives (*cf.* Bivens' (2015) analysis of how Facebook represented gender in their database for its first 10 years). Such research about issues of representation gesture toward the exploration of questions, such as "What factors make up developers' habits of mind that create such representational blindspots?" or larger questions such as "What cultural knowledge do different domains privilege over others? And what are the consequences of such privilege?"

5.5 Next Steps

Materiality studies of coding helps illuminate how the texts and their grammars exceed the signs on the screen. By applying this epistemology, coding and its grammar, syntaxes, and compilers of such rules become more than rules to follow. Signs come to be understood as entangled amongst one's history and contexts of coding. Grammars, signs, and their referents become expressions of space, people, and their social understanding about the world. Due to this dynamic nature of the materiality of writing, this dissertation serves as a beginning, rather than an end.

Ray's work with aggregate information, which shapes stories that reporters seek to tell their audiences, allies with other kinds of coding work that Ray conducted during my observational period. As I discussed in chapter 2, this dissertation reports findings from one of 4 embedded units of analysis. Future analytic work will be necessary to tease out how Ray's coding is shaped by myriad forms of data, and how such data represents social-material activity enacted in and through code.

My own actions to write up this dissertation enabled me to carve out the 4 embedded units of analysis, conduct particular analytical coding strategies, develop more nuanced ways to report my findings, and begin to integrate them into the a substantive theory of coding and its materialities as writing. From here, I can take the findings reported and continue my analytic work by coding the other TAPs of Ray's coding of provisional texts. Additionally, I have plans to conduct more refined mappings of the particular semiotic dimensions, such as time and space, of the data sets and respective code. Once I can solidify the different findings rendered from this embedded unit, I plan to code the 3 other units and cross-compare the findings all along the way. Indeed, I might consider asking how these findings about data processing and analysis compare against Ray's more traditional coding work to tool particular web applications. For example, during one observational session, Ray was writing code to develop a mapping tool to help reporters make interactive maps to embed in their stories. During this session, Ray coded a "place," as he put it, for someone else to either contribute or modify the code toward their particular needs. How might the concept of provisional text change, when compared against Ray's code that serves as a "place" for future users to code different options within a feature of the tool? Or, how might coding to process data change, when findings from this embedded unit are compared against the coding work to process data to archive it?

Additionally, a future potential study might further explore the tacit knowledge brought to the task of reading data sets. Such a study could include *in situ* observations of practitioners at work during their projects. Along the way, TAPs could provide data about how participants read data in relationship to their coding tasks.

5.6 </Open at the Close>

I hope that findings from this dissertation offers writing researchers and coding practitioners new language to fold into discourse communities that typically conceptualize coding through engineering metaphors. Writing, which includes coding, is not some pristine object that can easily be pinned down and explained wholly. Accordingly, what theories, concepts, habits, and values do people put to work in and through this most recent instantiation of language? If more writing researchers provide conceptual moves toward coding as language use, I hope to shift the discourse into more multiple conceptions of coding as writing practices. Overall, I hope that my dissertation research conveys just how much more research needs to be done and refined to illustrate better still how expressive people and their coding is and can be.

References

Abelson, B., Keefe, J., Wei, S., & Wiggins, C. (2015, November). *How data is changing media companies.* Panel at The Daily News Innovation Lab, New York, NY. Retrieved 23 May 2016, from `https://www.youtube.com/watch?v=1eOVN21je4k`

Aristotle. (2007). *On rhetoric* (G. Kennedy, Trans.). Oxford, NY: Oxford UP.

Atwood, J. (2008). *Coding: It's just writing.* Retrieved 10 Feb. 2017, from `https://blog.codinghorror.com/coding-its-just-writing/`

Bakhtin, M. M. (1986). The problem of speech genres. In A. Lock & C. Peters (Eds.), *Speech genres and other late essays* (pp. 60–102). Austin, TX: University of Texas Press.

Banks, A. J. (2006). *Race, rhetoric, and technology: Searching for higher ground.* New York, NY: Routledge.

Barton, D. (1991). The social nature of writing. In D. Barton & R. Ivanic (Eds.), *Writing in the community* (pp. 1–13). Newbury Park, CA: Sage Publications, Inc.

Barton, D. (1994). *Literacy: An introduction to the ecology of written language.* Cambridge, MA: Blackwell Publishers.

Basili, V. R., & Reiter, R. W. (1979). An investigation of human factors in software development. *IEEE Computer*, *12*(12), 21–38.

Batchelder, N. (2002). *Deleting code.* Retrieved 10 Feb. 2017, from `http://nedbatchelder.com/text/deleting-code.html`

Bazerman, C. (1988). *Shaping written knowledge: The genre and activity of the experimental article in science.* Madison, WI: University of Wisconsin Press.

Berkenkotter, C. (1995). *Genre knowledge in disciplinary communication: Cognition / culture / power.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Bertini, E., & Stefaner, M. (2017). Retrieved 2017-03-05, from http://datastori.es/

Bijker, W. (1995). *Of bicycles, bakelite, and bulbs: Toward a theory of sociotechnical change.* Cambridge, MA: MIT Press.

Bivens, R. (2015). The gender binary will not be deprogrammed: Ten years of coding gender on facebook. *New Media & Society*, 1–19.

Bracewell, R. J., & Witte, S. P. (2003). Tasks, ensembles, and activity: Linkages between text production and situation of use in the workplace. *Written Communication*, *20*, 511–559.

Brock, K. (2016). The 'Fizz Buzz' programming test: A case-based exploration of rhetorical style in code. *Computational Culture: A Journal of Software Studies*, *5*, n.p. Retrieved 26 Aug. 2016, from http://computationalculture.net/article/the-fizzbuzz-programming-test-a-case-based-exploration-of-rhetorical-style-in-code

Brock, K., & Kelly, A. R. (2015, April). *Rhetorical genres in code.* Paper presented at Indiana Digital Rhetoric Symposium, Bloomington, IN. Retrieved 05 May 2016, from https://www.youtube.com/watch?v=H2_iSznR3hg

Brooks, K., & Lindgren, C. (2015). Responding to the coding crisis: From code year to computational literacy. In L. C. Lewis (Ed.), *Strategic discourse: The politics of (new) literacy crises.* Salt Lake City, UT: Computers and Composition Digital Press.

Brooks, R. (1977). Towards a theory of the cognitive processes in computer programming. *International Journal of Man-Machine Studies*, *9*, 737–751.

Brooks, R. (1980). Studying programmer behavior experimentally: The problems of proper methodology. *Communications of the ACM*, *23*(4), 207–213.

Burnett, R. E., & Haas, C. (2007). Explicating writing as an embodied practice: A conversation between rebecca e. burnett and christina haas. *Journal of Business and Technical Communication*, *21*(1), 27–36.

Burns, H. (1979). *Stimulating rhetorical invention in english composition through computer-assisted instruction* (Doctoral dissertation, University of Texas, Austin, TX). Retrieved 2015-06-10, from `http://files.eric.ed.gov/fulltext/ED247602.pdf`

Buse, R. P., & Weimer, W. R. (2010). Learning a metric for code readability. *IEEE Transactions on Software Engineering*, *36*(4), 546–558.

Carter, J. L. (2016). *Making, disrupting, innovating.* Chair's Address presented at the meeting of Conference on College Composition and Communication, Houston, TX. Houston, TX, United States. Retrieved 29 Jun. 2016, from `https://www.youtube.com/watch?v=86FHQMZJI54`

Cordes, D., & Brown, M. (1991). The literate-programming paradigm. *IEEE Computer*, *24*(6), 52–61.

Cummings, R. (2006). Coding with power: Toward a rhetoric of computer coding and composition. *Computers and Composition*, *23*, 430–443.

Cushman, E. (2011). The Cherokee syllabary: A writing system in its own right. *Written Communication*, *28*(3), 225–281.

De Meo, M. (2014). *Twitter status.* Retrieved 2014-18-09, from `https://twitter.com/mistydemeo/status/509786151264468992`

Derrida, J. (1977/1988). *Limited inc* (S. Weber & J. Mehlman, Trans.). Evanston, IL: Northwestern UP.

Derrida, J. (1981). Plato's pharmacy. In B. Johnson (Ed.), *Dissemination.* Chicago, IL: University of Chicago Press.

Derrida, J. (1998). *Of grammatology* (G. C. Spivak, Trans.). Baltimore, MD: Johns Hopkins UP.

Devlin, J. (2008). *The programming aphorisms of Strunk and White.* Retrieved

10 Feb. 2017, from http://web.archive.org/web/20111031092020/http://www.codingthewheel.com/archives/programming-aphorisms-of-strunk-and-white

Doheny-Farina, S. (1993). Research as rhetoric: Confronting the methodological and ethical problems of research on writing in nonacademic settings. In *Writing in the workplace: New research perspectives* (pp. 253–267). Southern Illinois University Press Carbondale, IL.

Dyson, A. H., & Genishi, C. (2005). *On the case: Approaches to language and literacy* (Vol. 76). New York, NY: Teachers College Press.

Farkas, K., & Haas, C. (2012). A grounded theory approach for studying writing and literacy. In K. Powell & P. Takayoshi (Eds.), *Practicing research in writing studies: Reflexive and ethically responsible research* (pp. 81–96). New York, NY: Hampton Press.

Finley, K. (2017, March). The FCC seems unlikely to stop internet providers from selling your data. *WIRED Online*. Retrieved 04-17-2017, from https://www.wired.com/2017/03/fcc-graciously-sets-internet-providers-free-sell-data/

Flyvbjerg, B. (2006). Five mis-understandings about case-study research. *ACM Transactions on Computer-Human Interaction*, *12*, 219–245. Retrieved from http://portal.acm.org/citation.cfm?doid=353485.353487

Geertz, C. (1973). Thick description: Towards and interpretive theory of culture. In C. Geertz (Ed.), *The interpretation of cultures* (pp. 3–30). New York, NY: Basic Books.

Geisler, C. (2001). Textual objects accounting for the role of texts in the everyday life of complex organizations. *Written communication*, *18*(3), 296–325.

Glaser, B. G., & Strauss, A. L. (1967/2009). *The discovery of grounded theory: Strategies for qualitative research.* Piscataway, NJ: Aldine Transaction.

Goody, J. (1977/2001). Whats in a list? In E. Cushman, E. Knutgen, B. Kroll, & M. Rose (Eds.), *Literacy: A critical sourcebook* (pp. 32–51). Boston, MA:

Bedford/St. Martins.

Goody, J., & Watt, I. (1968). The consequences of literacy. In J. Goody (Ed.), *Literacy in traditional societies.* New York, NY: Cambridge UP.

Grabill, J. T. (2001). *Community literacy programs and the politics of change.* SUNY Press.

Graff, R. (2005). Prose versus poetry in early Greek theories of style. *Rhetorica: A Journal of the History of Rhetoric*, *23*(4), 303–335.

Greene, S., & Higgins, L. (1994). "Once Upon a Time": The use of retrospective accounts in building theory in Composition. In P. Smagorinsky (Ed.), *Speaking about writing: Reflections on research methodology* (pp. 20–54). Thousand Oaks, CA: SAGE Publications.

Groskopf, C. (2016, December). *The Quartz guide to bad data.* Retrieved 2017-03-13, from `https://github.com/Quartz/bad-data-guide`

Gutiérrez, K. D., & Stone, L. D. (2000). Synchronic and diachronic dimensions of social practice: An emerging methodology for cultural-historical perspectives on literacy learning. In C. D. Lee & P. Smagorinsky (Eds.), *Vygotskian perspectives on literacy research: Constructing meaning through collaborative inquiry* (pp. 150–164). New York, NY: Cambridge University Press.

Haas, C. (1996). *Writing technology: Studies on the materiality of literacy.* Mahwah, NJ: Routledge.

Haas, C. (1999). On the relationship between old and new technologies. *Computers and Composition*, *16*, 209–228.

Haas, C., & Flower, L. (1988). Rhetorical reading strategies and the construction of meaning. *College Composition and Communication*, *39*, 167–183.

Haas, C., & Takayoshi, P. (2011). Building and maintaining contexts in interactive networked writing: An examination of deixis and intertextuality in instant messaging. *Research in the Teaching of English*, *25*, 276–298.

Haas, C., Takayoshi, P., Carr, B., Hudson, K., & Pollock, R. (2011). Young peoples everyday literacy: The language of instant messaging. *Research in the Teaching of English*, *45*, 378–414.

Haefner, J. (1999). The politics of code. *Computers and Composition*, *16*, 325–339.

Hart-Davidson, W. (2012). *Code? Not so much [blog].* Digital Rhetoric Collaborative. Retrieved 29 Jun. 2016, from `http://www.digitalrhetoriccollaborative.org/2012/10/17/code-not-so-much/`

Haswell, R. H., Briggs, T. L., Fay, J. A., Gillen, N. K., Harrill, R., Shupala, A. M., & Trevino, S. S. (1999). Context and rhetorical reading strategies: Haas and flower (1988) revisited. *Written Communication*, *16*, 3–27.

Havelock, E. A. (1976). *Origins of Western literacy.* Toronto: Ontario Institute in Education.

Hayles, N. K. (2005). *My mother was a computer: Digital subjects and literary texts.* Chicago, IL: University of Chicago Press.

Heath, S. B. (1982). What no bedtime story means: Narrative skills at home and school. *Language Society*, *2*, 49–76.

Heath, S. B., & Street, B. V. (2008). *On ethnography: Approaches to language and literacy research.* New York, NY: Teachers College Press.

Herman, M., Rivera, S., Mills, S., Sullivan, J., Guerra, P., Cosmas, A., & Kim, M. (2015). *The field guide to data science.* Booz Allen Hamilton [self-published by company]. Retrieved from `https://www.boozallen.com/content/dam/boozallen_site/sig/pdf/publications/2015-field-guide-to-data-science-160211215115.pdf`

Hutchins, E. (1995). *Cognition in the wild.* Cambridge, MA: MIT Press.

Katz, S. M. (2002). Ethnographic research. In L. Gurak & M. M. Lay (Eds.), *Research in technical communication* (pp. 23–46). Westport, CT: Praeger.

Kernighan, B. W., & Plauger, P. J. (1978). *The elements of programming style.* New York, NY: McGraw-Hill.

Kim, M., Bergman, L., Lau, T., & Notkin, D. (2004). An ethnographic study of copy and paste programming practices in OOPL. In (pp. 83–92). IEEE. Retrieved 2015-12-05, from `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1334896`

Kirsh, D. (2010). Thinking with external representations. *AI & Society*, *25*(4), 441–454.

Knuth, D. (1968/1992). The art of programming. In D. Knuth (Ed.), *Literate programming.* United States: Center for the Study of Language and Information Lecture Notes.

Knuth, D. (1972). An empirical study of FORTRAN programs. *Software: Practice and Experience*, *1*, 105–133.

Knuth, D. E. (1992). Literate programming. *CSLI Lecture Notes, Stanford, CA: Center for the Study of Language and Information*, *1*.

Ko, A. (2016). *The invisibility of prior knowledge.* Retrieved 27 Sep. 2016, from `http://blogs.uw.edu/ajko/2016/09/27/the-invisibility-of-prior-knowledge/`

Ko, A. J., & Chilana, P. K. (2011). Design, discussion, and dissent in open bug reports. In (pp. 106–113). ACM Press. Retrieved 2015-12-05, from `http://portal.acm.org/citation.cfm?doid=1940761.1940776`

Ko, A. J., LaToza, T. D., & Burnett, M. M. (2015). A practical guide to controlled experiments of software engineering tools with human participants. *Empirical Software Engineering*, *20*(1), 110–141.

Koehls, F. (2008). *Writing modular code – make it legible.* Retrieved 10 Feb. 2017, from `http://hotkoehls.com/2008/04/writing-modular-code-make-it-legible/`

Koenemann-Belliveau, J. (Ed.). (1991). *Empirical studies of programmers: Fourth workshop.* Intellect Books.

Latour, B. (1988). Drawing things together. In M. Lynch & S. Woolgar (Eds.), *Representation in scientific practice* (pp. 19–69). Cambridge, MA: MIT Press.

Latour, B., & Woolgar, S. (1979/1986). *Laboratory life: The construction of scientific facts.* Princeton, NJ: Princeton UP.

Leblanc, P. (1993). *Writing teachers writing software: Creating our place in the electronic age.* Urbana, IL: NCTE.

Lewis, S. C. (Ed.). (2014). Journalism in an era of big data: Cases, concepts, and critiques. *Digital Journalism*, *3*(3), 321-466. doi: http://dx.doi.org/10.1080/21670811.2014.976399

Lindgren, C., & Temkin, D. (2015). *Interview with new media artist Daniel Temkin on esolangs.* Published for the CodeWork Collaborative, Insitute of Advanced Study, University of Minnesota in June 2015. Retrieved 01 Feb. 2017, from `http://umncodework.github.io/past_events/codework-interview-temkin-esolangs/`

Lopes, C. (2014a, April). *Exercises in Programming Style | Tagide.* Retrieved 2016-05-05, from `http://tagide.com/blog/academia/exercises-in-programming-style/`

Lopes, C. V. (2014b). *Exercises in Programming Style.* Boca Raton, FL: CRC Press.

Lupi, G., & Posavec, S. (2016). *Dear data.* New York, NY: Princeton Architectural Press.

Luria, A. R. (1962/1966). *Higher cortical functions in man* (B. Haigh, Trans.). New York: Basic Books & Plenum Press.

Mackiewicz, J., & Thompson, I. (2014). Instruction, cognitive scaffolding, and motivational scaffolding. *Composition Studies*, *42*, 54–78.

Marino, M. (2014). Field report for Critical Code Studies, 2014. *Computational Culture*, *4*. Retrieved 02 Feb. 2017, from `http://computationalculture.net/article/field-report-for-critical-code-studies-2014`

Mateas, M., & Montfort, N. (2005). A box, darkly: Obfuscation, weird languages, and code aesthetics. In *Proceedings of the 6th Digital Arts and Culture Conference, IT University of Copenhagen, 1-3 Dec. 2005* (pp. 144–153).

Mazzotti, M. (2017). Algorithmic life. *Los Angeles Review of Books*. Retrieved 31 Jan. 2017, from `https://lareviewofbooks.org/article/algorithmic-life/`

McGee, T., & Ericsson, P. (2002). The politics of the program: MS Word as the invisible grammarian. *Computers and Composition*, *19*, 453–470.

McIlroy, M. D., Pinson, E., & Tague, B. (1978). Unix time-sharing system: Foreword. *Bell System Technical Journal*, *57*(6), 1899–1904.

Miller, C. (1984). Genre as social action. *Quarterly Journal of Speech*, *70*, 151–167.

Montgomery, B. (2017a, April). If you are black. *Tampa Bay Times*. Retrieved 05-11-2017, from `http://www.tampabay.com/projects/2017/investigations/florida-police-shootings/if-youre-black/`

Montgomery, B. (2017b, April). Why cops shoot. *Tampa Bay Times*. Retrieved 05-11-2017, from `http://www.tampabay.com/projects/2017/investigations/florida-police-shootings/why-cops-shoot/`

Montgomery, B. (2017c, April). Why cops shoot: How we did it. *Tampa Bay Times*. Retrieved 05-11-2017, from `http://www.tampabay.com/projects/2017/investigations/florida-police-shootings/about-this-project-methodology/`

n.a. (2014). *Apple healthkit constants reference.* Retrieved 2014-18-09, from `https://developer.apple.com/library/ios/documentation/HealthKit/Reference/HealthKit_Constants/index.html#//apple_ref/c/tdef/HKBiologicalSex`

n.a. (2017a). *Computational thinking overview.* Retrieved 2017-05-05, from `https://edu.google.com/resources/programs/exploring-computational-thinking/#!ct-overview`

n.a. (2017b). *Mongodb: Documentation.* Retrieved 2017-04-28, from `https://docs.mongodb.com/manual/introduction/`

Nickell, E., & Smith, I. (2003). Extreme programming and software clones. In *Workshop on software clones.*

O'Neil, C. (2016). *Weapons of math destruction: How big data increases inequality and threatens democracy.* New York, NY: Crown Publishing Group.

Ong, W. J. (1958). *Ramus, method and the decay of dialogue.* Cambridge, MA: Harvard

UP.

Perez, F. (2014). *Ipython: From interactive computing to computational narratives.* Presented at the CSE Symposium on Weathering the Data Storm in January 2014. Retrieved 23 Sep. 2016, from `https://www.youtube.com/watch?v=V-EDqvscVxk`

Pigg, S. (2014a). Coordinating constant invention: Social media's role in distributed work. *Technical Communication Quarterly*, *23*(2), 69–87.

Pigg, S. (2014b). Emplacing mobile composing habits: A study of academic writing in networked social spaces. *College Composition and Communication*, *66*(2), 250.

Queneau, R. (1947/1981). *Exercises in style* (B. Wright, Trans.). New York, NY: New Directions.

Reynolds, N. (2007). *Geographies of writing: Inhabiting places and encountering difference.* SIU Press.

Richardson, E. (2003). *African American Literacies.* New York, NY: Routledge.

Roy, C. K., Cordy, J. R., & Koschke, R. (2009). Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. *Science of Computer Programming*, *74*(7), 470–495.

Royal, C., & Blasingame, D. (2014). *Data journalism explication - Product.* Presented at the International Symposium on Online Journalism in April 2015. Retrieved 23 Sep. 2016, from `https://youtu.be/Qi-iXJV9iow?t=58`

Saldana, J. (2009). *The coding manual for researchers.* Thousand Oaks, CA: SAGE Publications.

Salinger, S., Plonka, L., & Prechelt, L. (2008). A coding scheme development methodology: Using grounded theory for qualitative analysis of pair programming. *Human Technology: An Interdisciplinary Journal on Humans in ICT Environments*, *4*, 9–25.

Schmandt-Besseret, D. (1990). *Before writing.* Austin, TX: University of Texas Press.

Schriver, K. A. (1991, February). *Plain language for expert or lay audiences: Designing* *text using protocol-aided revision* (Tech. Rep. No. 46). University of California–Berkely, Carnegie Mellon University. Retrieved 27 Feb. 2017, from `https://www.nwp.org/cs/public/download/nwp_file/85/TR46.pdf?x-r=pcfile_d`

Scribner, S., & Cole, M. (1981). *The psychology of literacy.* Cambridge, MA: Harvard UP.

Shepardson, D. (2017, March). Congress may overturn Obama-era internet privacy rules. *Time.com*. Retrieved 04-17-2017, from `http://time.com/4693187/congress-internet-privacy-rules/`

Smagorinsky, P. (1994). Think-aloud protocols: Beyond the black box. In P. Smagorinsky (Ed.), *Speaking about writing: Reflections on research methodology* (pp. 3–19). Thousand Oaks, CA: SAGE Publications.

Smagorinsky, P. (2008). The method section as conceptual epicenter in constructing social science research reports. *Written Communication*, *25*(3), 389–411.

Solé, I., Miras, M., Castells, N., Espino, S., & Minguela, M. (2013). Integrating information: An analysis of the processes involved and the products generated in a written synthesis task. *Written Communication*, *30*, 63–90.

Soloway, E. (Ed.). (1986). *Empirical studies of programmers: First workshop* (Vol. 1). Intellect Books.

Sorapure, M. (2006). Text, image, code, comment: Writing in Flash. *Computers and Composition*, *23*, 412–429.

Spinuzzi, C., & Zachry, M. (2000, August). Genre ecologies: An open-system approach to understanding and constructing documentation. *ACM J. Comput. Doc.*, *24*(3), 169–181.

Spradley, J. P. (1979). *The ethnographic interview.* Belmont, CA: Wadsworth, Cengage Learning.

Strauss, A., & Corbin, J. (1998). *Basics of qualitative research: Techniques and procedures for developing grounded theory* (2nd ed.). Thousand Oaks, CA: SAGE Publications.

Strunk, W. J., & White, E. B. (2000). *The elements of style* (4th ed.). New York, NY: Allyn and Bacon.

Swarts, H., Flower, L., & Hayes, J. (1984). Designing protocol studies of the writing process: An introduction. In R. Beach & L. S. Bridwell (Eds.), *New directions in composition research: Perspectives in writing research* (pp. 53–71). New York, NY: The Guilford Press.

Thompson, I. (2009). Scaffolding in the writing center: A microanalysis of an experienced tutor's verbal and nonverbal tutoring strategies. *Written Communication*, *26*, 417–453.

Tufekci, Z. (2015). Algorithmic harms beyond facebook and google: Emergent challenges of computational agency. *Colorado Technology Law Journal*, *13*, 203–218. Retrieved 15 Feb. 2016, from `http://ctlj.colorado.edu/?page_id=238`

Turing, A. (1936/2013). On computable numbers, with an application to the Entscheidungsproblem. In S. Cooper & J. Leeuwen (Eds.), *Alan turing: His work and impact* (pp. 16–41). Waltham, MA: Elsevier.

Vee, A. (2017). *Coding literacy: How computer programming is changing the terms of writing.* Cambridge, MA: MIT Press [manuscript in final review stage].

Vygotksy, L. S. (1930/1987). The instrumental method. In R. Rieber, A. Carton, & J. Wollock (Eds.), *The collected works of L.S. Vygotsky (Cognition and language)* (pp. 85–89). New York, NY: Plenum Press.

Vygotsky, L. (1934/2012). *Thought and language* (A. Kozulin, Trans.). Cambridge, MA: MIT Press.

Wheat, M. (2006). *The elements of composition.* Retrieved 10 Feb. 2017, from `https://mitch-wheat.blogspot.com/2006/09/elements-of-composition.html`

Wickman, C. (2009). *Displays of knowledge: Text production and media reproduction in scientific practice* (Doctoral dissertation, Kent State University, Kent, OH). Retrieved 2015-12-05, from `http://rave.ohiolink.edu/etdc/view?acc_num=kent1247068612`

Wickman, C. (2010). Writing material in chemical physics research: The laboratory notebook as locus of technical and textual integration. *Written Communication*, *27*, 259–292.

Wickman, C. (2015). Locating the semiotic power of writing in science. *Journal of Business and Technical Communication*, *29*, 61–92.

Witte, S. P. (1992). Context, text, intertext: Toward a constructivist semiotic of writing. *Written Communication*, *9*, 237–308.

Witte, S. P. (2005). Research in activity: An analysis of speed bumps as mediational means. *Written Communication*, *22*, 127–165.

Yin, R. K. (2014). *Case study research: Design and methods.* Los Angeles, CA: SAGE Publications.

Appendix A

Appendices

A.1 Case Selection: “Request to Conduct Research”

REQUEST TO CONDUCT RESEARCH

Project title: Understanding computer programming as a writing practice
Researcher: Chris Lindgren, Writing Studies, University of Minnesota
Contact: (701) 893-6574, lindg250@umn.edu

Research project synopsis

What new insights can be gained by investigating computer programming with Writing Studies research methodologies? How do programmers make sense of their projects, tasks, and multiple information sources to ultimately write usable software? And, how are programming practices loaded with particular social messages and relations? My research seeks to answer these questions by conducting an observational study of programmers that traces the ways computer programming is intertwined with written, verbal, and graphical forms of communication.

How you will benefit from this research

My research will provide you with an expert perspective on the communicative work and activities surrounding programming practices. My training in Writing Studies provides me with a synthesis of social science, rhetorical (persuasive), and semiotic (meaning-making) methods and methodologies to understand how tacit cultural norms shape writing communication within and across contexts. My study, then, will serve as a means to unpack what is typically taken by many within particular groups to be “self-evident” and how such communicative behavior shapes the programming decisions during the development of software.

After I have analyzed the data, I could give a presentation to your company about my findings, which could highlight the programming practices that seem to work well versus others that do not. Such information could prove quite useful to project managers and team members.

My qualifications

I have a working foundational knowledge of computing and programming languages, and I am building on research that I have already conducted prior to this point. Specifically, I have taken an introductory course in Computer Science, and I have worked within the Computer Science department at the University of Minnesota. I am also proficient in Python and JavaScript, and I also dabble in Java and PHP. My master's thesis examined the historical-cultural movements to teach programming as a literacy. I hope this information communicates to you the following 3 points:

1. I already have foundational concepts and working knowledge of programming practices. I do not claim expertise in this area, but my proficiencies coupled with my disciplinary training provide me with a uniquely new perspective on computer programming.

2. I will not need to interrupt any work activity, during the day-to-day operation. I am there to observe the development process as it unfolds and will conduct rounds of interviews

specific to particular parts of such developments.

3. I am dedicated to my research's potential to help programming professionals, managers, and educators learn more about the ways tools and technical knowledge are intertwined with culture and other modes of communication.

Research methods

During my fieldwork, I will observe the work of programmers, record my observations by hand in a field notebook, and collect relevant artifacts related to the programming tasks. The table below summarizes my proposed methods and their aims.

Please note that many aspects of these methods are negotiable (*if* and *how* they will be conducted) based on the arrangements made with you. I completely understand how time is valuable and information can be sensitive, so arrangements will be made to define the necessary professional boundaries and expectations for an IRB-approved research study.

Method	Aims	Output
Observation	Observe programmers at work. Document their practices. Gain insight into how programmers conduct, coordinate, communicate, and complete their writing work.	Field notes.
Visual and textual documentation and collection	Document and collect texts that programmers produce and use in their writing work. Examine how programmers develop their code to fulfill the project's needs and communicate their technical work to themselves and broader audiences.	Photographs of writing environments & tools; Collect iterations of source code files and related resource materials.
Interview	Collect participant's language about programming practices and writing decisions. Check validity of analysis.	Negotiable amount, but would be short, semistructured interview with each programmer on team.
Video	Take video of programming practices to capture *in vivo* writing decisions.	Video & audio of *in vivo* code-writing practices.

Thank you for taking the time to learn more about my research project. I appreciate your effort to help me locate a research site for such a new way to examine and come to understand computer programming. Please contact me with any questions using the provided mobile number and email address on the first page.

A.2 Case Selection: “Research Breakdown”

Breakdown of research methods

chris a. lindgren
214 nolte center
315 pillsbury drive se
minneapolis, mn 55455

m [redacted]
lindg250@umn.edu

Coding history interview

- Conducted once at the start of the study
- ~90 minutes in length
- ~10 questions about your background and past experiences with coding
- Audio-recorded

Ethnographic observations

- Negotiable time frame; commonly 4-6 months and thinking between 8-16 hours/week. The wide time frame enables us to have some flexibility while maintaining consistency.
- Take fieldnotes in my notebook by hand
- Record video of screen activity
- I will be inductively coding this data throughout the duration of the study, looking for particular writing-behavior patterns.

Three-fold, finer-grained, task-specific methods

1.) Think-aloud protocols:

- Ask you to “think aloud” while completing a task , which you were already planing on doing during that work period.
- Conducted ~2-3 times
- Time contingent on task being conducted
- Audio/Video-recorded

2.) Discourse-based interviews:

- Very short interviews mediated by a material artifact that I ask questions about.
- Usually conducted to verify interpretations and see what “thinking” you bring to the object(s) in question.
- Audio-recoded

3.) Retrospective narrations:

- I play a short selection of writing activity via video and ask you to narrate it.
- Conducted ~2-3 times
- At most, 10-15 minutes; as few as 5 minutes.
- Audio/Video-recorded

A.3 Case Selection: “Research Site Request”

UNIVERSITY OF MINNESOTA

Twin Cities Campus

Department of Writing Studies
College of Liberal Arts

214 Nolte Center
315 Pillsbury Drive SE
Minneapolis, MN 55455

Office: 612-624-3445
Fax: 612-624-3617
Email: writ@umn.edu
Website: http://writingstudies.umn.edu

September 11, 2015

[redacted]
[redacted]
[redacted]

Dear [redacted] team:

In this letter, I am submitting the following requested information about my research study:

1. Description of the research study
2. The data collection process
3. Potential examples of artifacts that I will collect
4. Data storage, identity and anonymity

1. Description of the research study

What kinds of relationships obtain between writing code and other forms of written communication? And, what are the recurrent *writing practices* of web developers? These questions are embedded within the larger compelling question that my research agenda addresses: How can programming be better understood when studied explicitly as writing?

I hope that I can begin my research agenda at [redacted], where I have primed my case study to investigate writing code within a *writing practice* framework.[1, 2] *Writing practices* – the material technology, cultural knowledge, and skills binding a writing task – are historically-shaped, socially-organized, yet manifest in diverse context-specific ways. I intend to study how *writing code* shares these *practice* properties with traditionally-held forms of writing. Such a conceptual move provides myself and future researchers with the opportunity to understand how writing code is enmeshed and better understood when bundled with other forms of written communication. *Overall, my research bundles written communication in such a way to help develop more robust understandings of what it really takes to write code.*

My study responds to much of the existing research about computer programming from a wide range of disciplines (social and computing sciences), which understand programming as a discrete, technical, and generalizable skill.[3, 4, 5, 6] Such studies are hypothesis-driven, conducted in artificial conditions, and bifurcate the relationship between programming and writing. My study departs from this type of research with a case-study approach to illuminate how such an understanding is quite reductive programming as writing and how it obtains relationships with other forms of written communication.

I ask the questions below to guide my investigation into the recurrent writing practices of a web developer. Such an inquiry will show how writing code incorporates multiple modes of communication—all of which coordinate and facilitate a developer's understanding about the development teams' appropriate means of communication and configuration of writing activities within this particular domain of practice.

The research questions are as follows:

1. What tools (IDEs, code libraries, terminals, etc.) and artifacts (source code, documentation, correspondence, etc.) does a web developer use during writing tasks?

2. What types of reasoning systems, i.e., semiotic frameworks, exist and are constructed by the participant to establish levels of appropriateness and effectiveness about development decisions during particular writing tasks?

3. When considering such tools, artifacts, and semiotic frameworks, what are the participant's recurrent skills that put these elements into action during particular writing tasks? For instance, how does the participant utilize other communications or other constituents to source code to shape his code-writing?

Such questions will be investigated by observing how the participant's writing acts are linked to particular material tools and artifacts (question #1), cultural and community knowledge (question #2), and skills (question #3). When investigated separately, not much insight can be gleaned about the particular writing activity. Yet, when bundled together and observed *in situ*, my study will provide a nuanced view into how the code and sign systems developers use are inherently cultural and linked to material tools.

2. The data collection process

I will utilize the following methods to collect data specific to each of my research questions: 1) coding-history interview, 2) ethnographic observations, and 3) a three-fold battery of methods to collect data of task-specific, *in vivo* writing.

1.) Coding history interview

- Conducted once at the start of the study
- ~90 minutes in length
- ~10 questions about participant's coding-related background
- Digitally audio-recorded[7]

2.) Ethnographic observations

- Commonly 4-6 months and between 6-12 hours/week. This time-frame enables us to have some flexibility while maintaining consistency
- Take fieldnotes in my notebook by hand[8]
- Collect relevant physical / digital artifacts: photos of workspace, source code, documents, and/or logs of relevant correspondence, chat logs, or visited web pages[7, 8]
- Digitally recorded video of screen activity[7]

3.) Three-fold, finer-grained, task-specific methods

i. Think-aloud protocols:

- Ask participant to “think aloud” while completing a task, which he was already planning on doing during that work period
- Conducted ~2-3 times
- Time contingent on task being conducted
- Digitally audio/video-recorded[7] and fieldnotes[8]

ii. Discourse-based interviews:

- Very short interviews mediated by a material artifact that I ask questions about
- Usually conducted to *verify interpretations* and see what “thinking” participant brought to the writing produced and in question
- Conducted periodically throughout the study
- Digitally audio-recorded[7] and hand-written notes[8]

iii. Retrospective accounts of video-recorded tasks:

- I play a short selection of writing activity via video and ask the participant to narrate his reasoning behind his (in)decisions
- Conducted ~2-3 times
- At most, 10-15 minutes, or as few as 5 minutes
- Digitally audio/video-recorded[7] and hand-written notes[8]

I will inductively code this data throughout the duration of the study, using qualitative models to develop reliable and valid chains of evidence (data triangulation, resource maps, memos, handoff chains, etc.) about particular writing-behavior patterns.[9, 10]

3. Potential examples of artifacts that I will collect

As noted in section 2, I will collect a variety of data by means of a variety of methods. Specific to artifacts, I will collect any relevant correspondence, documentation, web browser history used during the observational periods.

- Correspondence could be chat logs or emails with colleagues, as he works through particular coding tasks and problems.
- Documentation could be particular web pages specific to a code library that he is using that day.
- Web browser history will be of interest, if he utilizes a particular ensemble of searches and sites to overcome a problem by constructing a workable solution.

My unit of analysis also incorporates the aforementioned written and verbal communication with the participant's code-writing. I will be using screen-recording software to capture his *in vivo* coding with the variety of tools that the participant uses to write, test, refactor, etc. his code. Additionally, I will collect iterations of particular source-code files in relation to the above data.

Overall, all of these potential physical and digital artifacts are related to understanding web development as a writing practice. My case study draws upon multiple sources of data (digital/physical artifacts, interviews, observations, *in vivo* screen recordings) as a way to systematically develop a convergence of evidence through data triangulation related to the participant's writing behavior.[10]

4. Data storage, identity and anonymity

My study will be approved by the University of Minnesota's Institutional Review Board. Pseudonyms will be used and all identifying information will be scrubbed and redacted. For instance, all identifying information in any correspondence or communication that he has with his co-workers will be redacted, and video data will be used only for my own data analysis. Digital data will be encrypted on an external hard drive and stored securely with other physical data in a lock-box within my locked office.

My inductive approach to data collection and analysis includes *asking permission to use and how to use* particular artifacts within any scholarly materials. In short, I will not use any materials in my scholarly writing without first seeking consent from the participant.

I want to thank everyone at [redacted] involved in this negotiation process. I appreciate all of your time and effort to learn more about my project. I hope that I will have the opportunity to observe and understand the amazing web development work being done at [redacted]. I'm excited to bring a fresh disciplinary perspectives on such practices within this novel domain of journalism.

If you have any questions for me, my contact information is included below.

Sincerely,

Chris Lindgren
Ph.D. Candidate in Rhetoric and Science and Technical Communication
University of Minnesota-Twin Cities
Department of Writing Studies
lindg250@umn.edu
[redacted]

[1] Scribner, S., & Cole, M. (1981). *The psychology of literacy*. Cambridge, MA: Harvard UP.

[2] Haas, C. (1996). *Writing technology: Studies on the materiality of literacy*. Mahwah, N.J: Routledge.

[3] Brandt, J., Guo, P.J., Lewenstein, J., Dontcheva, M., & Klemmer, S.R. (2009). Two studies of opportunistic programming: Interleaving web foraging, learning, and writing code. Proceedings from *ACM Computer-Human Interaction, Boston, MA, 4-9 Apr. 2009*, pp. 1589-1598.

[4] Dabbish, L., Stuart, C., Tsay, J., & Herbsich, J. (2012). Social coding in GitHub: Transparency and collaboration in an open software repository. Proceedings from ACM *CSCW, Seattle, WA, 11-15 Feb. 2012* (pp. 1277-1286).

[5] Hollan, J., Hutchins, E., & Kirsh, D. (2000). Distributed cognition: Toward a new foundation for Human-Computer Interaction research. *ACM Transactions on Computer-Human Interaction, 7*(2), pp. 174-196

[6] Ko, A.J., LaToza, T., & Burnett, M. M. (2013). A practical guide to controlled experiments of software engineering tools with human participants. *Empirical Software Engineering, 20*, pp. 110-141.

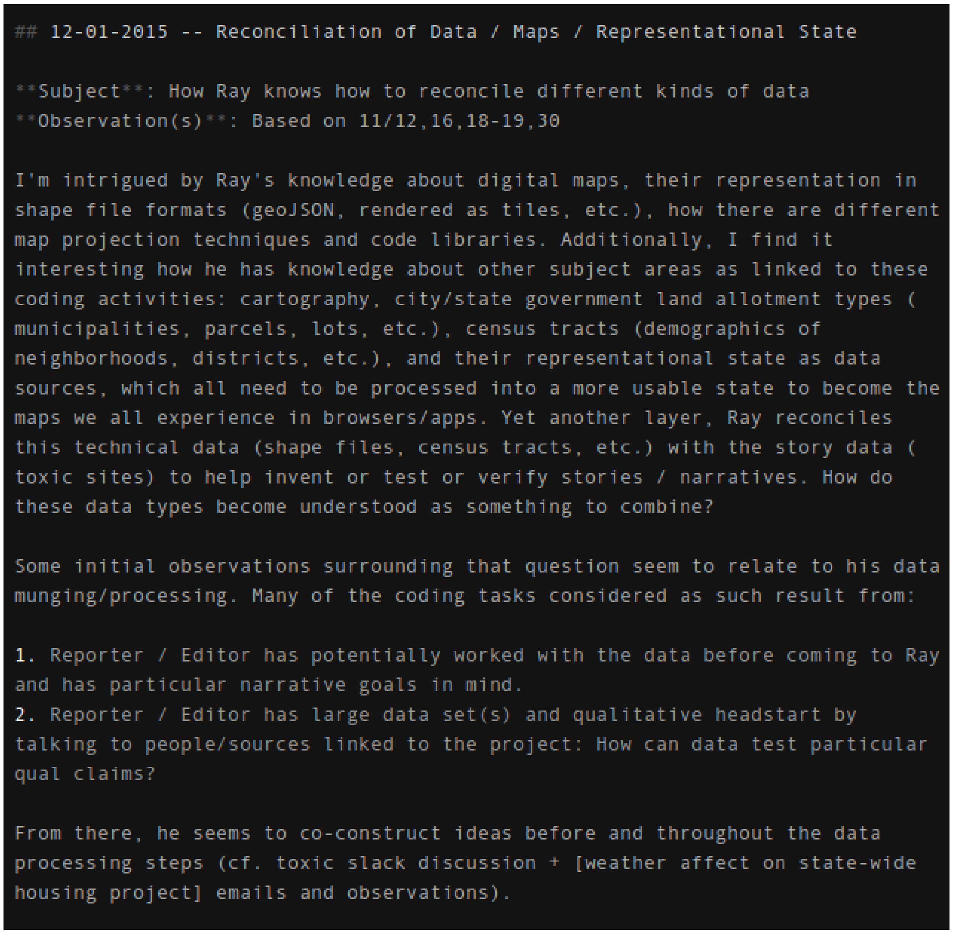
[7] Digital artifacts stored on encrypted hard drive.

[8] Physical artifacts stored in lock-box within secure office.

[9] See the following about modified grounded-theory methodologies specific to the discipline of Writing Studies: Farkas, K., & Haas, C. (2012). A grounded theory approach for studying writing and literacy. In K. Powell and P. Takayoshi (Eds.), *Socially responsible research methods for writing and literacy*. Hampton Press.

[10] Yin, R. (2014). *Case study research: Design and methods*. Thousand Oaks, CA: SAGE Publications, Inc.

A.4 Figure: Analytic memo about Ray's coding work with data during the open coding phase



```
## 12-01-2015 -- Reconciliation of Data / Maps / Representational State

**Subject**: How Ray knows how to reconcile different kinds of data
**Observation(s)**: Based on 11/12,16,18-19,30

I'm intrigued by Ray's knowledge about digital maps, their representation in
shape file formats (geoJSON, rendered as tiles, etc.), how there are different
map projection techniques and code libraries. Additionally, I find it
interesting how he has knowledge about other subject areas as linked to these
coding activities: cartography, city/state government land allotment types (
municipalities, parcels, lots, etc.), census tracts (demographics of
neighborhoods, districts, etc.), and their representational state as data
sources, which all need to be processed into a more usable state to become the
maps we all experience in browsers/apps. Yet another layer, Ray reconciles
this technical data (shape files, census tracts, etc.) with the story data (
toxic sites) to help invent or test or verify stories / narratives. How do
these data types become understood as something to combine?

Some initial observations surrounding that question seem to relate to his data
munging/processing. Many of the coding tasks considered as such result from:

1. Reporter / Editor has potentially worked with the data before coming to Ray
and has particular narrative goals in mind.
2. Reporter / Editor has large data set(s) and qualitative headstart by
talking to people/sources linked to the project: How can data test particular
qual claims?

From there, he seems to co-construct ideas before and throughout the data
processing steps (cf. toxic slack discussion + [weather affect on state-wide
housing project] emails and observations).
```

Figure A.1: Screen capture of early analytic memo about Ray's knowledge about particular types of data, how to work with it, and reconcile different kinds of data.

A.5 Tabulated List of Open Codes

Table A.1: List of initial open codes generated during the first phase of the first movement of research.

Open Codes
processing data / munging data / combining data / reading data / writing data / analyzing data / visualizing data / projects / project-based / toxic sites / train times / natural disasters / state government building project / assessments / testing / verifying / tallies / totals / averages / correlation / significance / consistency / regularity / integrity / story / reporting / aggregate / ‘create aggregate’ / ‘aggregate view’ / feature / property / patterns / data bias / ‘incomplete’ data / ‘raw’ data / original data / data source / our data / data points / data structures / reading in / file paths / input/output / parsing / file formats / CSV / JSON / geoJSON / TSV / shape files / PDF tables / spreadsheets / OpenOffice Calc / Mac Numbers / text files / HTML tables / data miner / scraping / new data / old data / mapping / projection / reprojection / state projections / coordinates / municipalities / parcels / lots / districts / census tracts / demographics / CartoDB / FactFinder / Census Reporter / Google Maps / regex / rendering / tiles / addresses / standardizing / web cache / browser / inspector tools / project architecture / folders / files / organization / ‘junk’ / README / documentation / identifier / Google search / Github / git / repositories / copy-pasting / translating code / dependencies / package.json / Makefile / shell script / Terminal / console logs / geocoding / API / rate limits / expenses / code modules / code libraries / proj4 / lodash / CRON / server / archiving / Ractive.js / Node.js / Javascript / CSS / HTML / templating / templates / project exigencies / data + goal / data + storyfinding / data + goal and storyfinding / Ray + data + storyfinding / data consultation / Slack / email / Google Hangout / standup meetings / Slack messaging / Slack channels / direct channels / team channels / project channels / data questions / ‘describe the data’ / code questions / project questions / context / tooling / linting / gulp / code styles / open source / UX / whiteboarding / mockups / task delegation / end-user / end-user programming / option trees / ‘place to code’ / canvas / markers / aspect ratio / map styles / Mapbox / cartography / search locations / labels / tooltips / prior coded feature / zoom / builds / generate / scripting / functions / conditionals / drag-drop / menus / icons / collapse / responsive / mobile / clone / parameters / arguments / this / return / stackoverflow.com / comments

A.6 Think-Aloud Protocol Review Form

Think-Aloud Protocols

What are TAPs?

- Verbalize moment-to-moment thoughts while performing task

- Building a chain of thoughts as you do something

- Important to keep verbalizing, talking through what would usually be a “pregnant pause” during a task

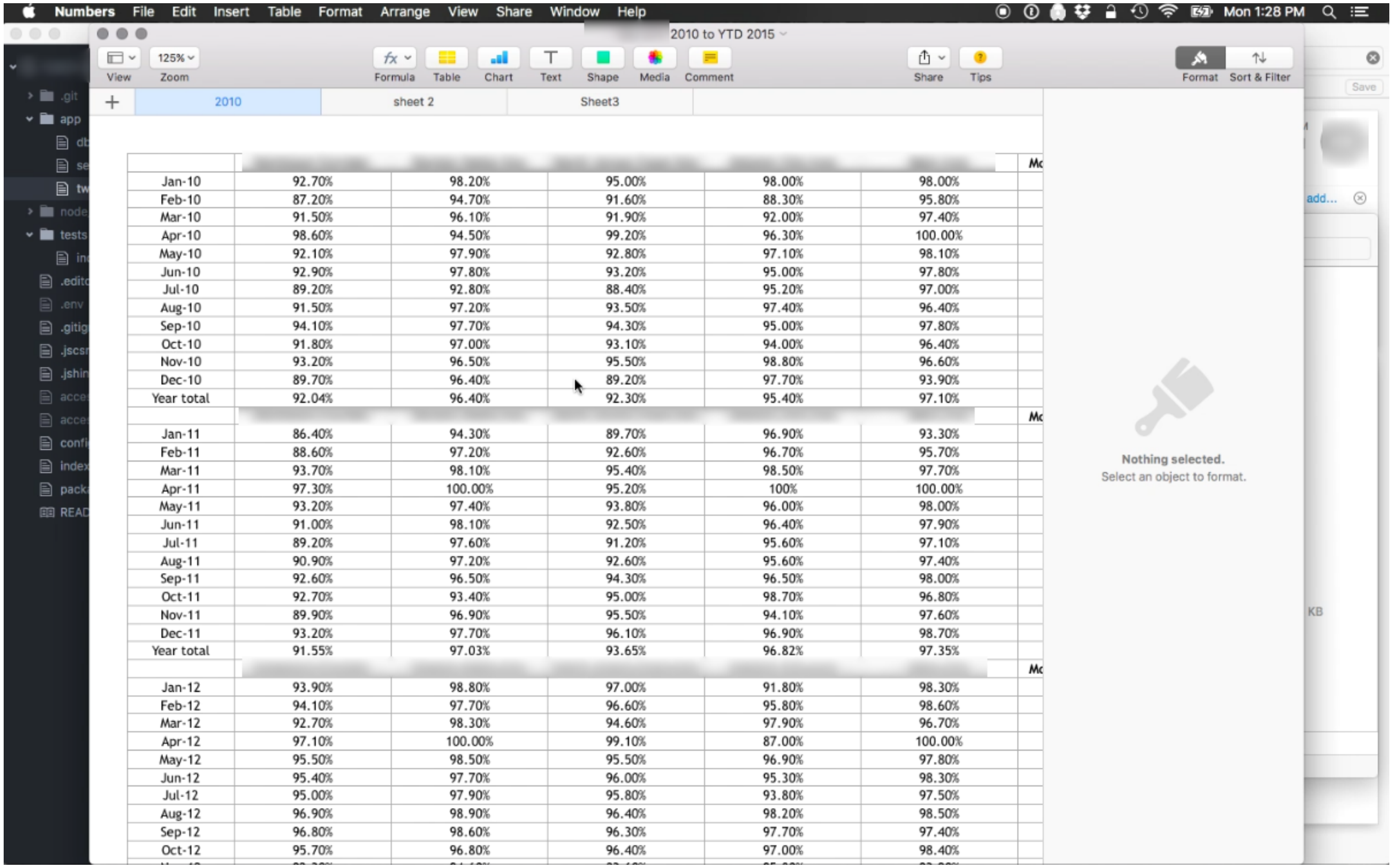
Example statements

- I'm writing var X to use in Y function. I'm naming it A, because it ...
- I'm copying this code to make it easier to write this data structure in this file.
- I'm naming this function to do X.
- I'm thinking about the best way place to write function A for different reasons: 1) … and 2) … I am deciding to write it here, because …
- I'm reviewing X & Y columns and rows in this CSV to understand how they relate to one another, so I can respond to {{person}} request

Basic Process

- Spot a task, then prep
- During the task, if you stop talking, I will say, “Keep talking”
- Conduct TAP until task is completed – perhaps a “git commit” moment.

A.7 Figure: Public Train, On-Time, Performance Data Set



	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Mc
Jan-10	92.70%	98.20%	95.00%	98.00%	98.00%	
Feb-10	87.20%	94.70%	91.60%	88.30%	95.80%	
Mar-10	91.50%	96.10%	91.90%	92.00%	97.40%	
Apr-10	98.60%	94.50%	99.20%	96.30%	100.00%	
May-10	92.10%	97.90%	92.80%	97.10%	98.10%	
Jun-10	92.90%	97.80%	93.20%	95.00%	97.80%	
Jul-10	89.20%	92.80%	88.40%	95.20%	97.00%	
Aug-10	91.50%	97.20%	93.50%	97.40%	96.40%	
Sep-10	94.10%	97.70%	94.30%	95.00%	97.80%	
Oct-10	91.80%	97.00%	93.10%	94.00%	96.40%	
Nov-10	93.20%	96.50%	95.50%	98.80%	96.60%	
Dec-10	89.70%	96.40%	89.20%	97.70%	93.90%	
Year total	92.04%	96.40%	92.30%	95.40%	97.10%	
	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Mc
Jan-11	86.40%	94.30%	89.70%	96.90%	93.30%	
Feb-11	88.60%	97.20%	92.60%	96.70%	95.70%	
Mar-11	93.70%	98.10%	95.40%	98.50%	97.70%	
Apr-11	97.30%	100.00%	95.20%	100%	100.00%	
May-11	93.20%	97.40%	93.80%	96.00%	98.00%	
Jun-11	91.00%	98.10%	92.50%	96.40%	97.90%	
Jul-11	89.20%	97.60%	91.20%	95.60%	97.10%	
Aug-11	90.90%	97.20%	92.60%	95.60%	97.40%	
Sep-11	92.60%	96.50%	94.30%	96.50%	98.00%	
Oct-11	92.70%	93.40%	95.00%	98.70%	96.80%	
Nov-11	89.90%	96.90%	95.50%	94.10%	97.60%	
Dec-11	93.20%	97.70%	96.10%	96.90%	98.70%	
Year total	91.55%	97.03%	93.65%	96.82%	97.35%	
	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Mc
Jan-12	93.90%	98.80%	97.00%	91.80%	98.30%	
Feb-12	94.10%	97.70%	96.60%	95.80%	98.60%	
Mar-12	92.70%	98.30%	94.60%	97.90%	96.70%	
Apr-12	97.10%	100.00%	99.10%	87.00%	100.00%	
May-12	95.50%	98.50%	95.50%	96.90%	97.80%	
Jun-12	95.40%	97.70%	96.00%	95.30%	98.30%	
Jul-12	95.00%	97.90%	95.80%	93.80%	97.50%	
Aug-12	96.90%	98.90%	96.40%	98.20%	98.50%	
Sep-12	96.80%	98.60%	96.30%	97.70%	97.40%	
Oct-12	95.70%	96.80%	96.40%	97.00%	98.40%	

Nothing selected.
Select an object to format.

Figure A.2: Screen capture of the OTP data set, which mediated Ray's discussion with a reporter, Sharon, who inquired about possible ways to use the data.

A.8 Figure: News team's storyboard matrix

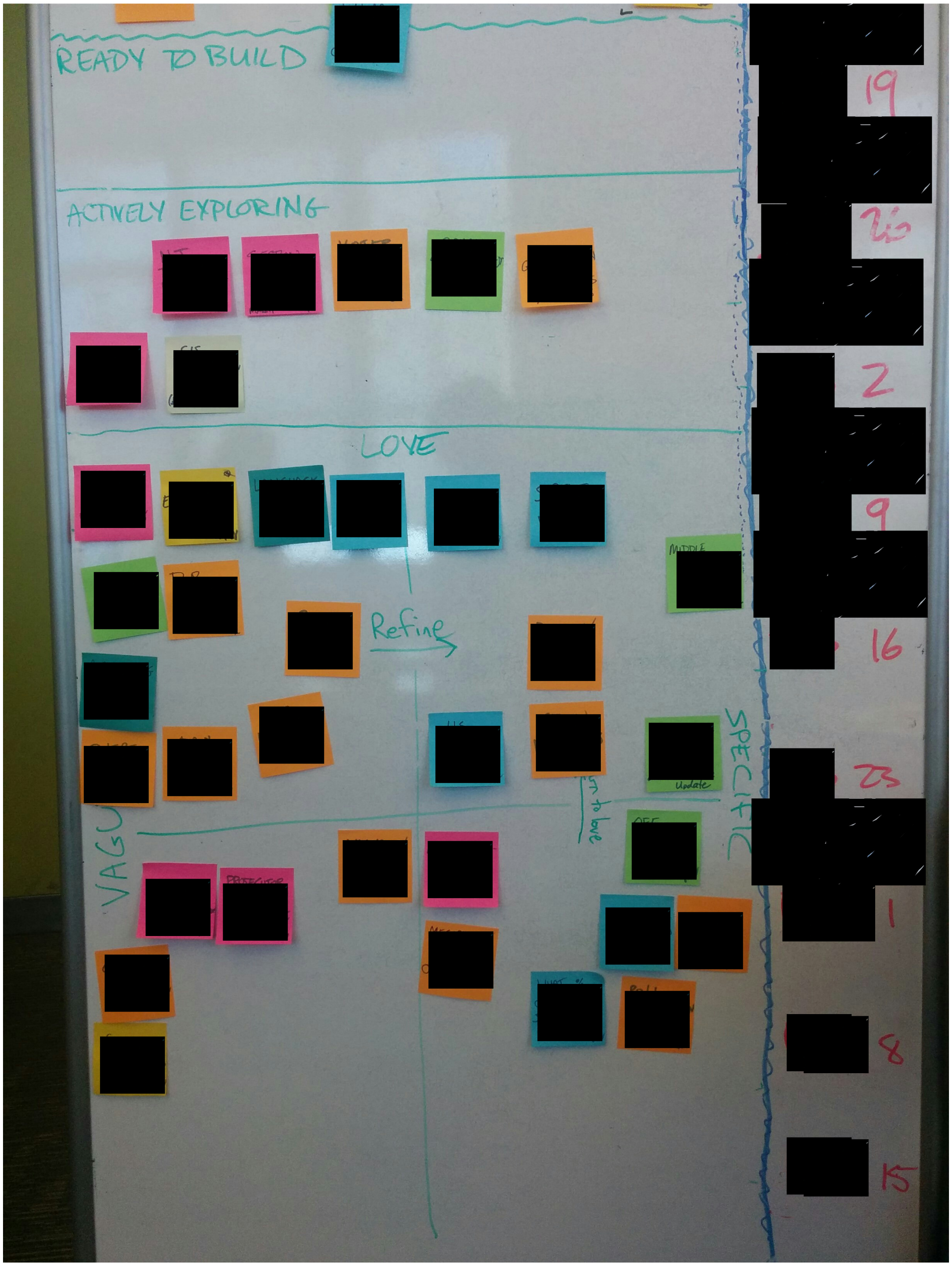
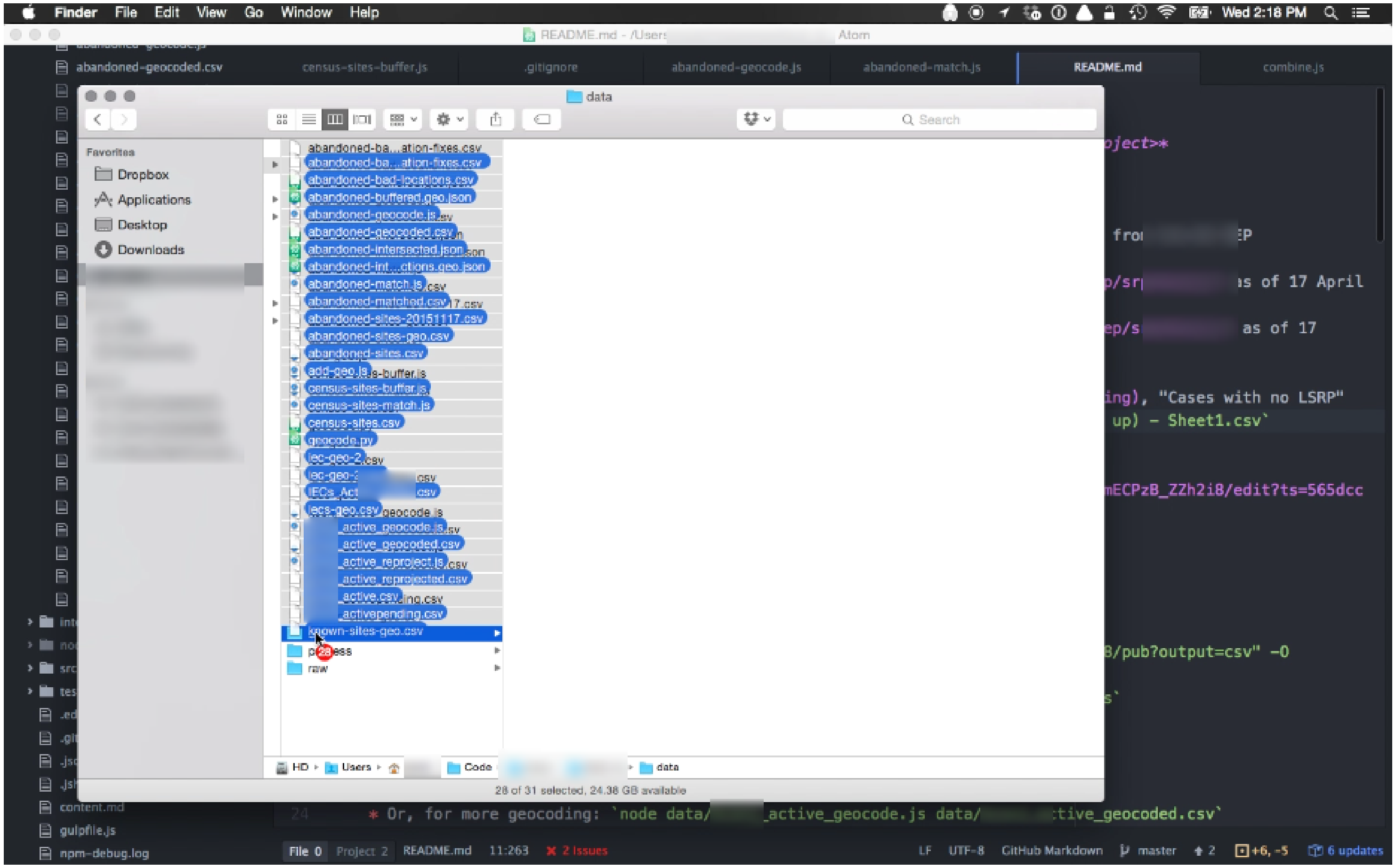


Figure A.3: News team storyboard matrix to denote publishing status. (All story titles and other identifiers redacted.)

A.9 Figure: Ray dumping the original data sets into the ‘raw’ directory



Finder File Edit View Go Window Help
Wed 2:18 PM
README.md - /Users[redacted] Atom
abandoned-geocoded.csv
census-sites-buffer.js
.gitignore
abandoned-geocode.js
abandoned-match.js
README.md
combine.js
data
Search
Favorites
Dropbox
Applications
Desktop
Downloads
HD ▸ Users ▸ [redacted] Code [redacted] ▸ data
28 of 31 selected, 24.38 GB available
content.md
gulpfile.js
npm-debug.log
File 0 Project 2 README.md 11:263 ✖ 2 Issues LF UTF-8 GitHub Markdown master ⬆ 2 +6, -5 6 updates

```
abandoned-ba...ation-fixes.csv
abandoned-ba...ation-fixes.csv
abandoned-bad-locations.csv
abandoned-buffered.geo.json
abandoned-geocode.js
abandoned-geocoded.csv
abandoned-intersected.json
abandoned-int...ctions.geo.json
abandoned-match.js
abandoned-matched.csv
abandoned-sites-20151117.csv
abandoned-sites-geo.csv
abandoned-sites.csv
add-geo.js
census-sites-buffer.js
census-sites-match.js
census-sites.csv
geocode.py
iec-geo-2.csv
iec-geo-[redacted].csv
IECs_Act[redacted].csv
iecs-geo.csv
[redacted]_active_geocode.js
[redacted]_active_geocoded.csv
[redacted]_active_reproject.js
[redacted]_active_reprojected.csv
[redacted]_active.csv
[redacted]_activepending.csv
known-sites-geo.csv
process
raw
```

```
oject>*
 fro[redacted]EP
p/sr[redacted]as of 17 April
ep/s[redacted] as of 17
ing), "Cases with no LSRP"
 up) - Sheet1.csv`
mECPzB_ZZh2i8/edit?ts=565dcc
8/pub?output=csv" -O
s`
* Or, for more geocoding: `node data/[redacted]_active_geocode.js data/[redacted]tive_geocoded.csv`
```

Figure A.4: Ray dumping all of the original data sets and scripts into the ‘data/raw’ directory.

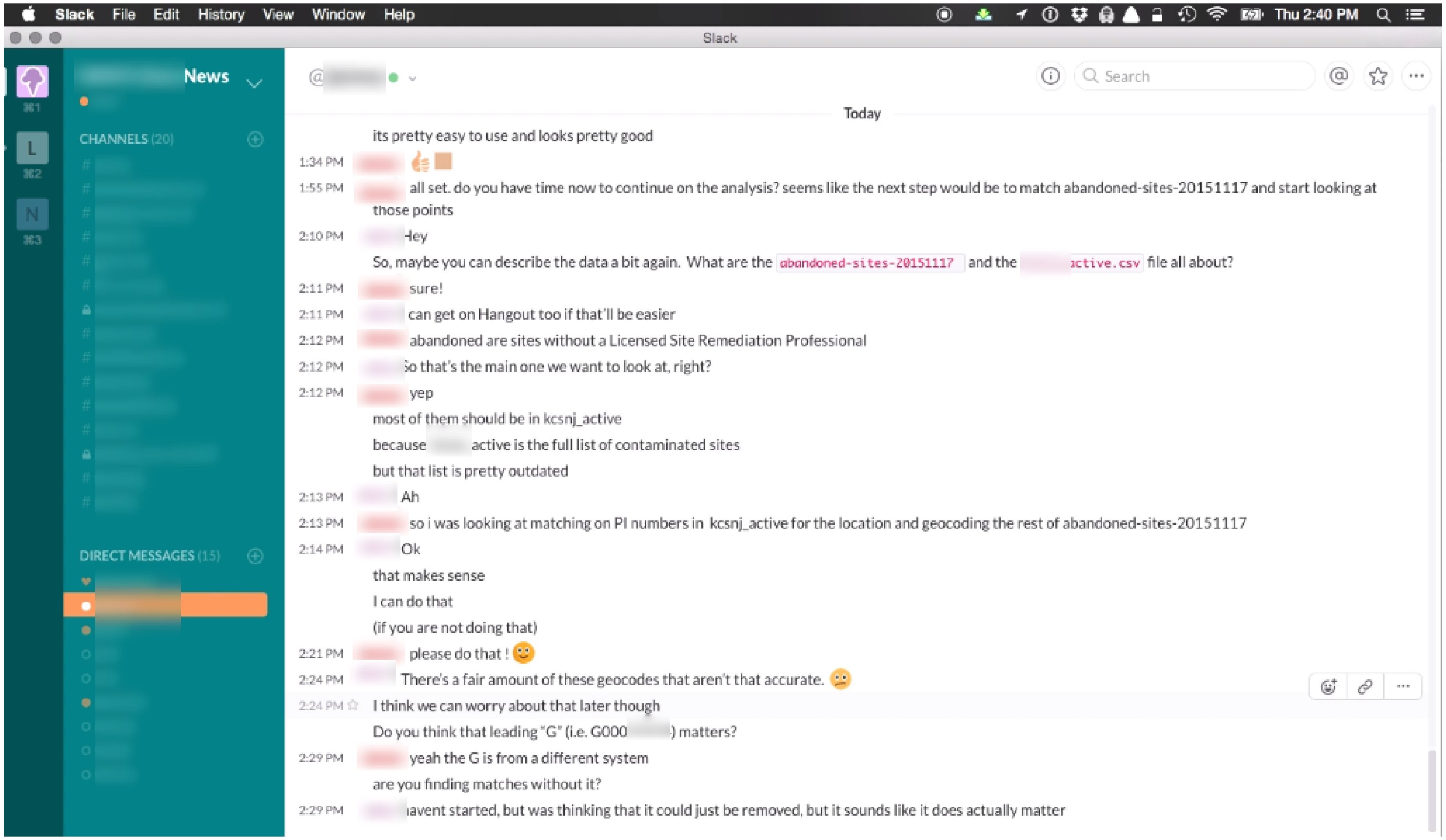
A.10 Complete clean.js file for City Payroll project

```
/**
 * Takes original CSV data and cleans into JSON so that we can
 * do analysis easier.
 */

// Dependencies
var path = require("path");
var fs = require("fs");
var io = require("indian-ocean");
var _ = require("lodash");
var moment = require("moment");
var pace = require("pace");


// Inputs
var payroll = io.readCsvSync(path.join(__dirname,
    "../original/[redacted]-Payroll-Data-CY2014.csv"));

// Outputs
var outputPath = path.join(__dirname, "../working/payroll-cleaned.json");

// See how things are going
var pacer = pace(payroll.length);

// Go through the data
payroll = _.map(payroll, function(row) {
  var start = parseDate(row["Agency Start Date"]);

  var parsed = {
    pNumber: row["Payroll Number"],
    dept: row["Payroll DescriptionCY2014"].trim(),
  deptID: makeID(row["Payroll DescriptionCY2014"]),
  last: row["Last Name"].trim(),
  first: row["First Name"].trim(),
  middle: row["Mid Init"].trim(),
  start: start ? start.format("YYYY-MM-DD") : null,
  yrsAgency: start ? shortNumber(moment("2014-12-31", "YYYY-MM-DD").diff(start, "years", true)) : null,
  leaveStatus: row["Leave Status as of December 31, 2014 "],
  title: row["Title Description"].trim(),
  titleID: makeID(row["Title Description"]),
  base: parseNumber(row["Base Salary"]),
  payType: row["Pay Basis"].trim().toLowerCase() === "per annum" ? "year" :
    row["Pay Basis"].trim().toLowerCase() === "per day" ? "day" : "hour",
  hours: parseNumber(row["Regular Hours"]),
  paid: parseNumber(row["Regular Gross Paid"]),
  hoursOT: parseNumber(row["OT Hours"]),
  paidOT: parseNumber(row["Total OT Paid"]),
  paidOther: parseNumber(row["Total Other Pay"])
};

// Some combinations
parsed.name = (parsed.first + " " + parsed.middle + " " + parsed.last).replace(/\s+/g, " ");
parsed.paidTotal = parsed.paid + parsed.paidOT + parsed.paidOther;
parsed.perReg = parsed.paidTotal ? shortNumber(parsed.paid / parsed.paidTotal, 4) : null;
parsed.perOT = parsed.paidTotal ? shortNumber(parsed.paidOT / parsed.paidTotal, 4) : null;
parsed.perOther = parsed.paidTotal ? shortNumber(parsed.paidOther / parsed.paidTotal, 4) : null;
parsed.perRegOT = parsed.paid ? shortNumber(parsed.paidOT / parsed.paid, 4 ) : null;

pacer.op();
```

```
  return parsed;
});

// Parse number
function parseNumber(input) {
  var parsed = parseFloat(input.toString().replace("$", ""));
  return _.isNaN(parsed) ? null : parsed;
}

// Parse date
function parseDate(input) {
  input = input.toString().trim();
  var inputFormat = "MM/DD/YYYY HH:mm:ss AA";
  return input && moment(input, inputFormat).isValid() &&
    moment(input, inputFormat).year() > 1901 &&
    moment(input, inputFormat).year() < 2016 ?
    moment(input, inputFormat) : null;
}

// Short number
function shortNumber(input, places) {
  places = places || 2;
  return Math.round(input * (10 ^ places)) / (10 ^ places);
}

// Make an more standard identifier for a string
function makeID(input) {
  return input.toString().toLowerCase().trim()
    .replace(/\W/g, " ")
    .replace(/\s+/g, "-");
}


// Saving as JSON might give a memory error, use CSV
```

```
io.writeDataSync(outputPath, payroll);
```

A.11 Example of Ray's provisional text work across technologies during the State Toxic Sites project.



its pretty easy to use and looks pretty good

1:34 PM [redacted] 👍

1:55 PM [redacted] all set. do you have time now to continue on the analysis? seems like the next step would be to match abandoned-sites-20151117 and start looking at those points

2:10 PM [redacted] Hey

So, maybe you can describe the data a bit again. What are the `abandoned-sites-20151117` and the [redacted]`active.csv` file all about?

2:11 PM [redacted] sure!

2:11 PM [redacted] can get on Hangout too if that'll be easier

2:12 PM [redacted] abandoned are sites without a Licensed Site Remediation Professional

2:12 PM [redacted] So that's the main one we want to look at, right?

2:12 PM [redacted] yep

most of them should be in kcsnj_active

because [redacted] active is the full list of contaminated sites

but that list is pretty outdated

2:13 PM [redacted] Ah

2:13 PM [redacted] so i was looking at matching on PI numbers in kcsnj_active for the location and geocoding the rest of abandoned-sites-20151117

2:14 PM [redacted] Ok

that makes sense

I can do that

(if you are not doing that)

2:21 PM [redacted] please do that ! 🙂

2:24 PM [redacted] There's a fair amount of these geocodes that aren't that accurate. 😕

2:24 PM I think we can worry about that later though

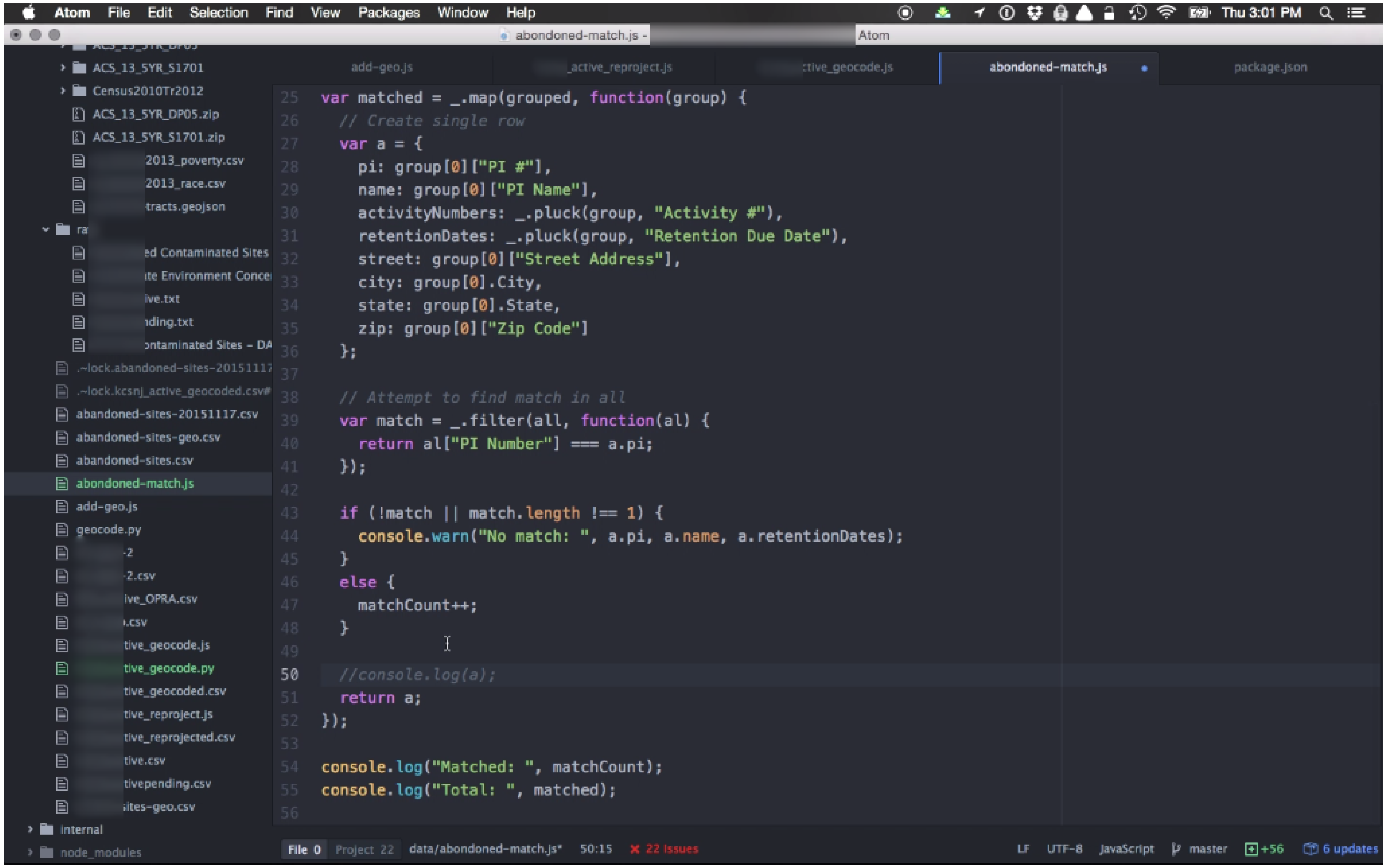
Do you think that leading "G" (i.e. G000[redacted]) matters?

2:29 PM [redacted] yeah the G is from a different system

are you finding matches without it?

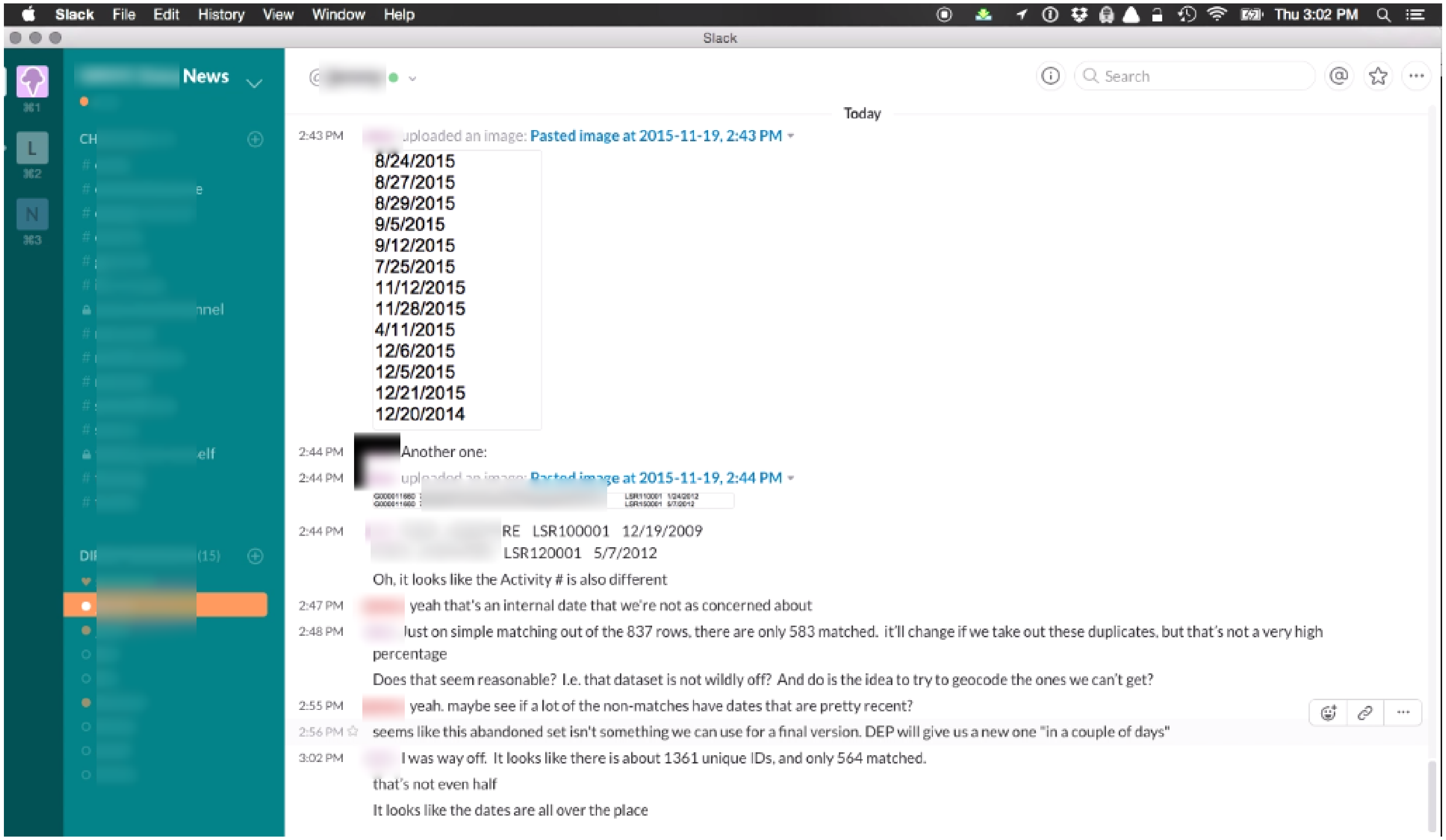
2:29 PM [redacted] havent started, but was thinking that it could just be removed, but it sounds like it does actually matter

Figure A.5: Jun asks Ray to match up 2 data sets on Slack, so they know how many more addresses they need to geocode.



```
var matched = _.map(grouped, function(group) {
  // Create single row
  var a = {
    pi: group[0]["PI #"],
    name: group[0]["PI Name"],
    activityNumbers: _.pluck(group, "Activity #"),
    retentionDates: _.pluck(group, "Retention Due Date"),
    street: group[0]["Street Address"],
    city: group[0].City,
    state: group[0].State,
    zip: group[0]["Zip Code"]
  };

  // Attempt to find match in all
  var match = _.filter(all, function(al) {
    return al["PI Number"] === a.pi;
  });

  if (!match || match.length !== 1) {
    console.warn("No match: ", a.pi, a.name, a.retentionDates);
  }
  else {
    matchCount++;
  }

  //console.log(a);
  return a;
});

console.log("Matched: ", matchCount);
console.log("Total: ", matched);
```

Figure A.6: Ray writes the console log in the code above to output the matching site information.



2:43 PM uploaded an image: Pasted image at 2015-11-19, 2:43 PM

8/24/2015
8/27/2015
8/29/2015
9/5/2015
9/12/2015
7/25/2015
11/12/2015
11/28/2015
4/11/2015
12/6/2015
12/5/2015
12/21/2015
12/20/2014

2:44 PM [redacted] Another one:

2:44 PM uploaded an image: Pasted image at 2015-11-19, 2:44 PM

2:44 PM RE LSR100001 12/19/2009
LSR120001 5/7/2012

Oh, it looks like the Activity # is also different

2:47 PM yeah that's an internal date that we're not as concerned about

2:48 PM Just on simple matching out of the 837 rows, there are only 583 matched. it'll change if we take out these duplicates, but that's not a very high percentage

Does that seem reasonable? I.e. that dataset is not wildly off? And do is the idea to try to geocode the ones we can't get?

2:55 PM yeah. maybe see if a lot of the non-matches have dates that are pretty recent?

2:56 PM seems like this abandoned set isn't something we can use for a final version. DEP will give us a new one "in a couple of days"

3:02 PM I was way off. It looks like there is about 1361 unique IDs, and only 564 matched.

that's not even half

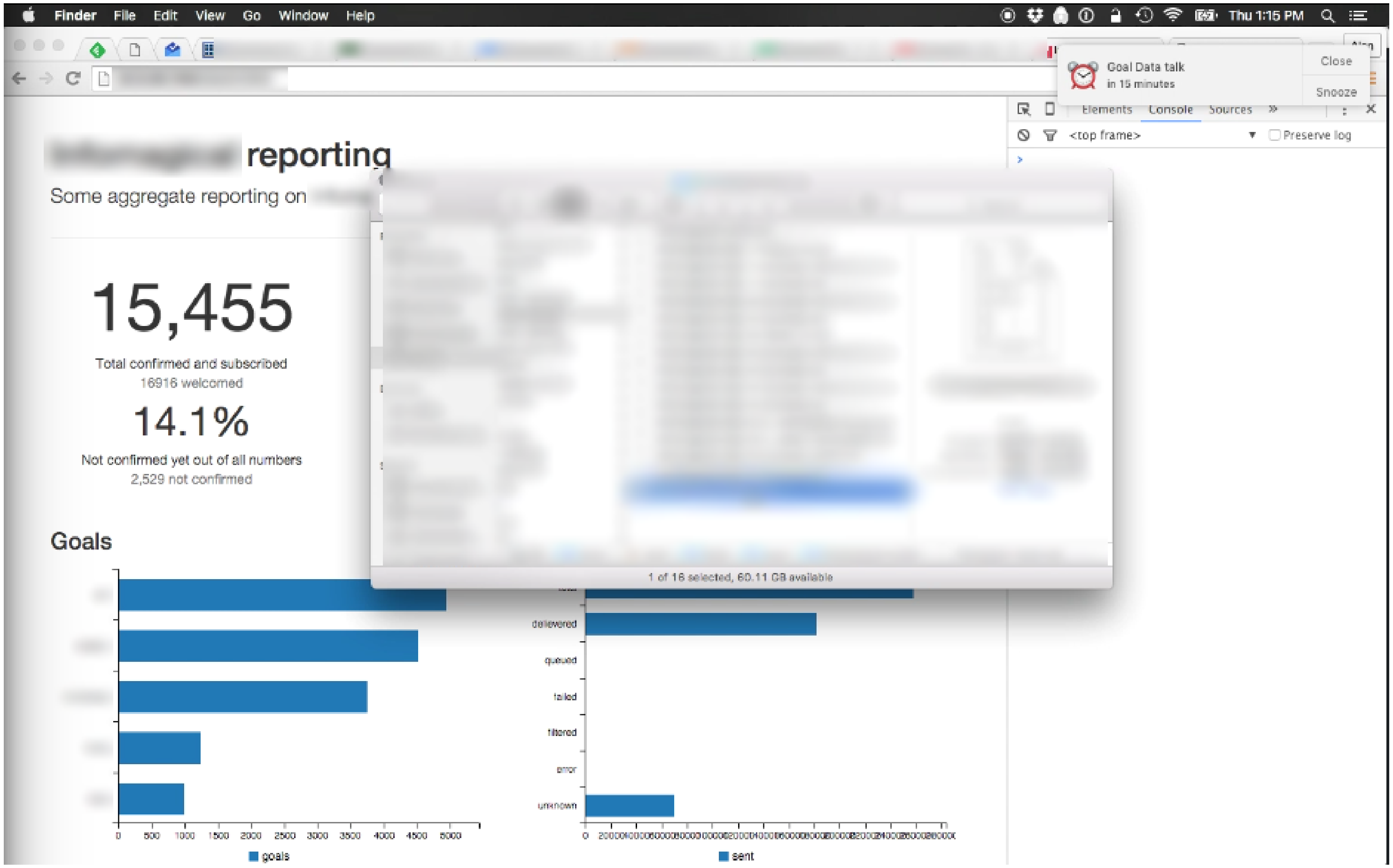
It looks like the dates are all over the place

Figure A.7: Ray shares the matching counts with Jun on Slack.

A.12 Example of Ray's provisional text work across technologies during the Health Texting project.

```
/**
 * JS for reporting endpoints and related
 */

...

// Main api for reporting
function reporting(request, response) {
  // Get data

  ...

    // Get status
    var data = {
      cached: cached,
      total: people.length,
      unique: _.groupBy(people, "phone").length,
      confirmed: confirmed.length,
      unsubscribed: _.filter(people, function(p) {
        return !p.subscribed;
      }).length,
      welcomed: _.filter(people, function(d) {
        return _.filter(d.sent, function(s) {
          return s.welcome;
        }).length;
      }).length,
      sent: _.reduce(people, function(total, p) {
        return total + p.sent.length;
      }, 0),
      received: _.reduce(people, function(total, p) {
        return total + p.received.length;
      }, 0),
      sentFailed: _.reduce(people, function(total, p) {
        return total + _.filter(p.sent, function(s) {
          return s.failed;
        }).length;
      }, 0),
      sentDelivered: _.reduce(people, function(total, p) {
        return total + _.filter(p.sent, function(s) {
          return s.status === "delivered";
        }).length;
      }, 0),

      ...
    };

...
```

Figure A.8: Excerpt from Ray's `reporting.js`, which served as a real-time provisional text for the podcasting team.



reporting

Some aggregate reporting on

15,455

Total confirmed and subscribed

16916 welcomed

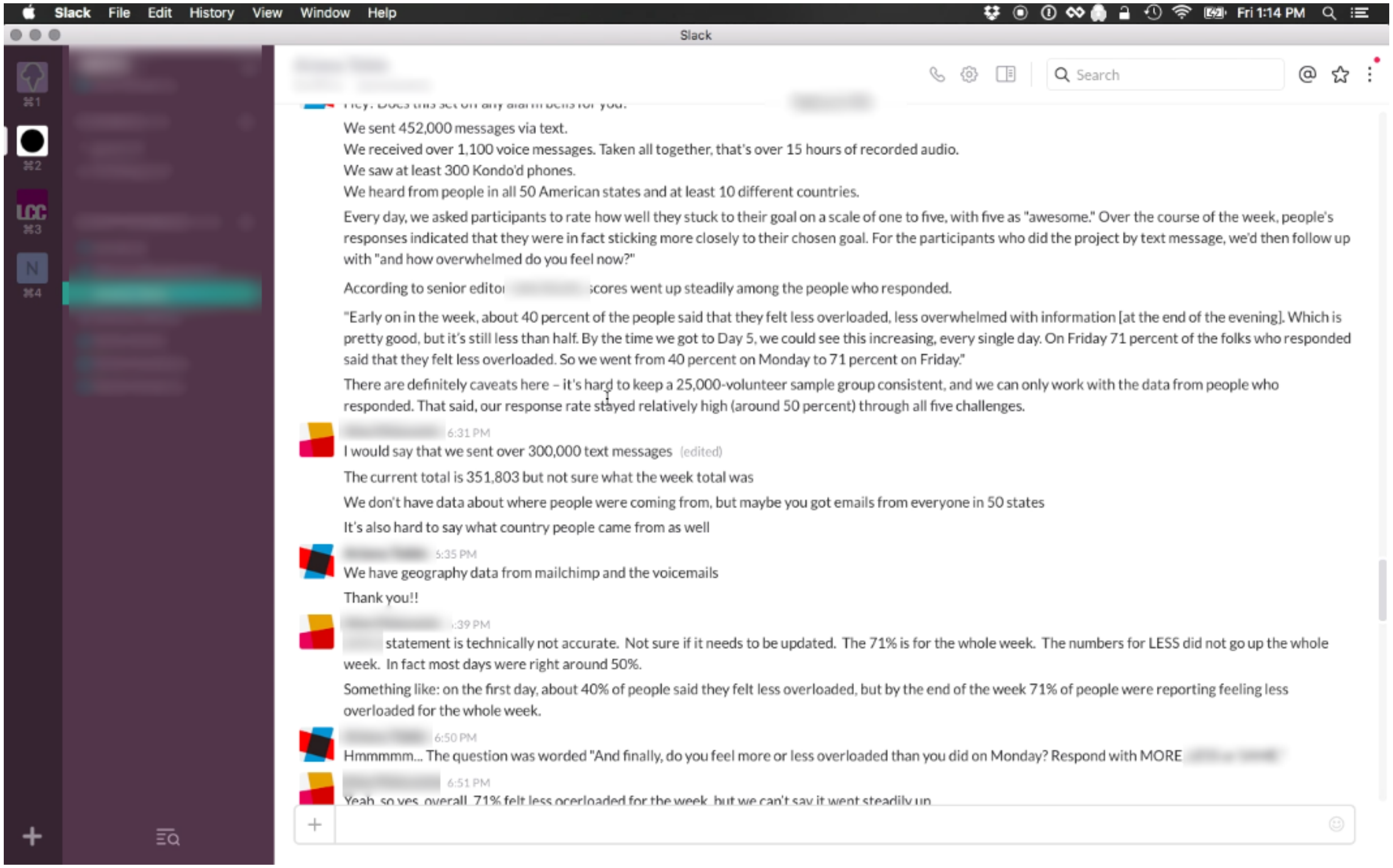
14.1%

Not confirmed yet out of all numbers

2,529 not confirmed

Goals

Figure A.9: Screenshot of Ray's dashboard for the podcasting team to use multiple slices of the database of participants in the Health Texting project.



Hey! Does this set off any alarm bells for you.

We sent 452,000 messages via text.
We received over 1,100 voice messages. Taken all together, that's over 15 hours of recorded audio.
We saw at least 300 Kondo'd phones.
We heard from people in all 50 American states and at least 10 different countries.

Every day, we asked participants to rate how well they stuck to their goal on a scale of one to five, with five as "awesome." Over the course of the week, people's responses indicated that they were in fact sticking more closely to their chosen goal. For the participants who did the project by text message, we'd then follow up with "and how overwhelmed do you feel now?"

According to senior editor [redacted] scores went up steadily among the people who responded.

"Early on in the week, about 40 percent of the people said that they felt less overloaded, less overwhelmed with information [at the end of the evening]. Which is pretty good, but it's still less than half. By the time we got to Day 5, we could see this increasing, every single day. On Friday 71 percent of the folks who responded said that they felt less overloaded. So we went from 40 percent on Monday to 71 percent on Friday."

There are definitely caveats here – it's hard to keep a 25,000-volunteer sample group consistent, and we can only work with the data from people who responded. That said, our response rate stayed relatively high (around 50 percent) through all five challenges.

[redacted] 6:31 PM
I would say that we sent over 300,000 text messages (edited)

The current total is 351,803 but not sure what the week total was

We don't have data about where people were coming from, but maybe you got emails from everyone in 50 states

It's also hard to say what country people came from as well

[redacted] 6:35 PM
We have geography data from mailchimp and the voicemails

Thank you!!

[redacted] 6:39 PM
[redacted] statement is technically not accurate. Not sure if it needs to be updated. The 71% is for the whole week. The numbers for LESS did not go up the whole week. In fact most days were right around 50%.

Something like: on the first day, about 40% of people said they felt less overloaded, but by the end of the week 71% of people were reporting feeling less overloaded for the whole week.

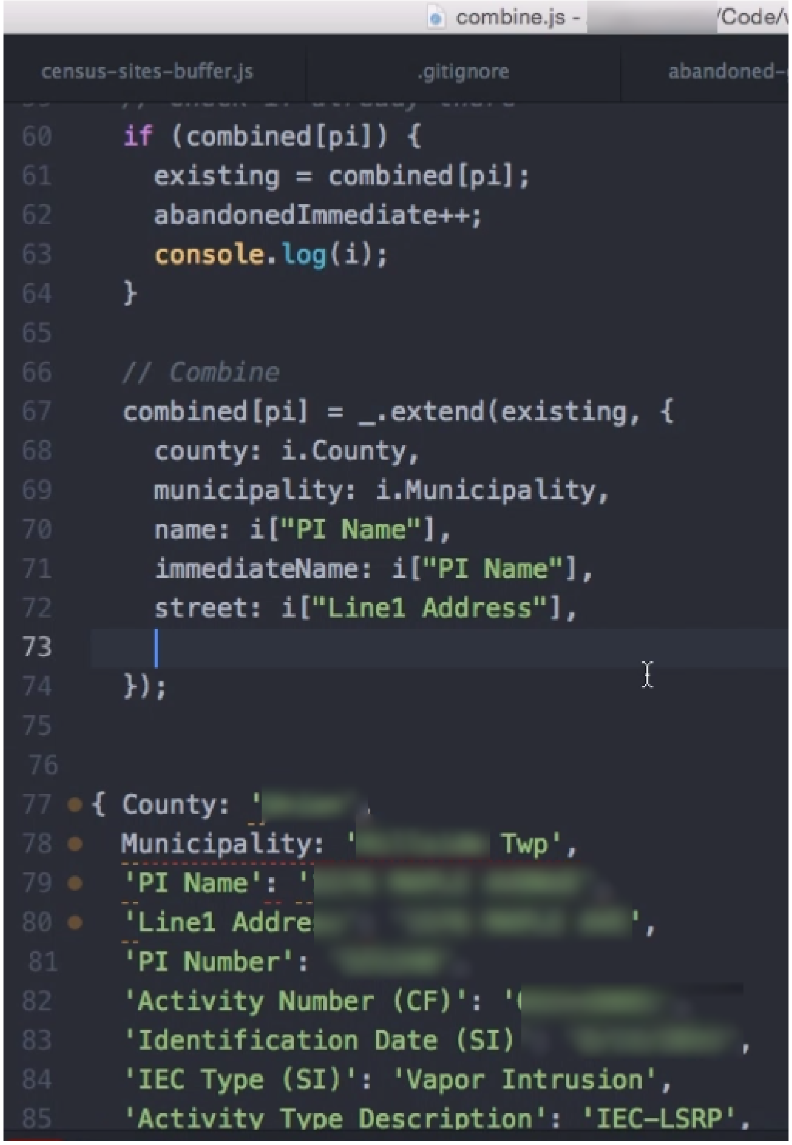
[redacted] 6:50 PM
Hmmmmm... The question was worded "And finally, do you feel more or less overloaded than you did on Monday? Respond with MORE [redacted]

[redacted] 6:51 PM
Yeah, so yes, overall, 71% felt less ocerloaded for the week, but we can't say it went steadily up

Figure A.10: One of the assistants for the podcast, Avery, verifies a script for the day's show.

A.13 Extended excerpt of the `combine.js` retrospective account

Transcription of Retrospective Account (20160107)

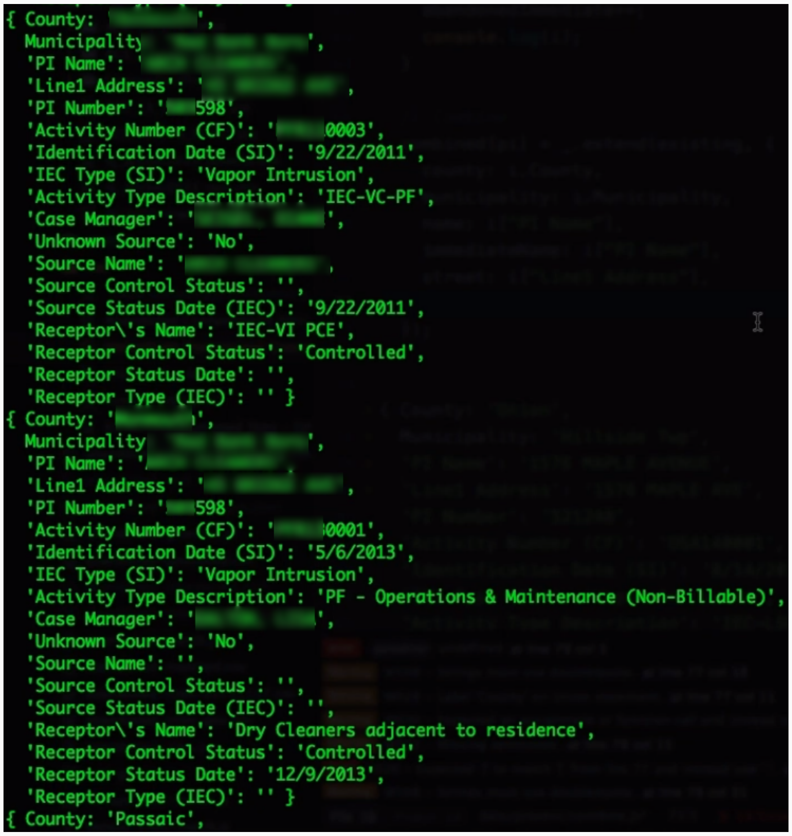
Timestamp	Action description	Screen capture of video
0:20 – 0:25	- Video: Pauses a moment on line 73 within the `// Combine` code before swtiching to Terminal with a readout of the site data within object arrays	

combine.js - [redacted] /Code/

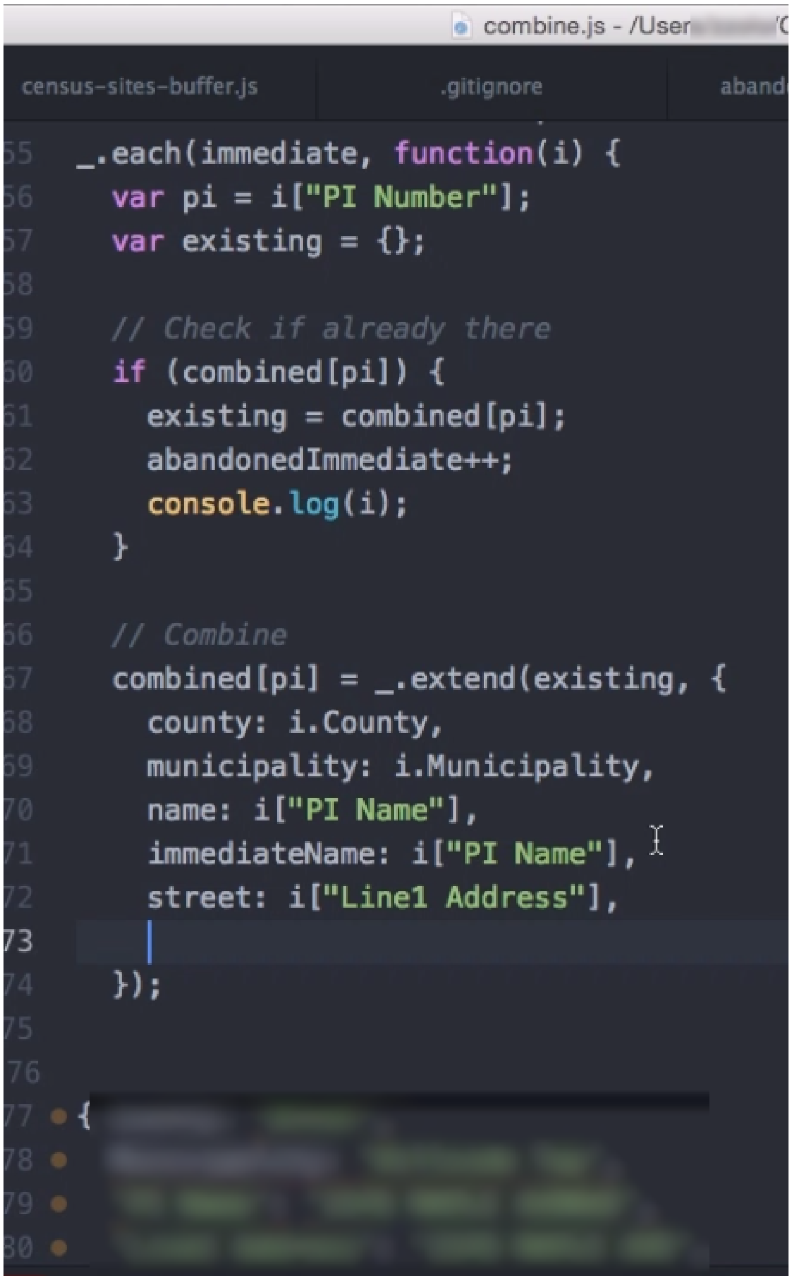
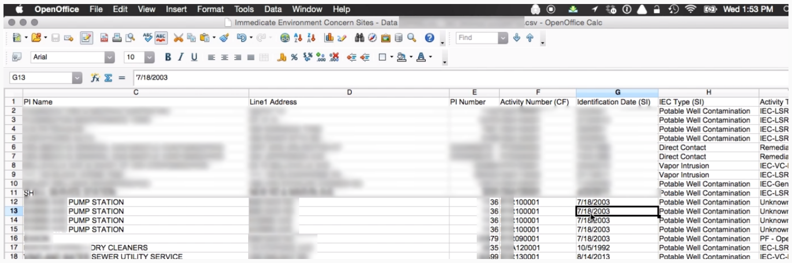
census-sites-buffer.js .gitignore abandoned-

```
60    if (combined[pi]) {
61      existing = combined[pi];
62      abandonedImmediate++;
63      console.log(i);
64    }
65
66    // Combine
67    combined[pi] = _.extend(existing, {
68      county: i.County,
69      municipality: i.Municipality,
70      name: i["PI Name"],
71      immediateName: i["PI Name"],
72      street: i["Line1 Address"],
73      
74    });
75
76
77  { County: '[redacted]
78    Municipality: '[redacted] Twp',
79    'PI Name': '[redacted]
80    'Line1 Addre[redacted]',
81    'PI Number': [redacted]
82    'Activity Number (CF)': '[redacted]
83    'Identification Date (SI)[redacted],
84    'IEC Type (SI)': 'Vapor Intrusion',
85    'Activity Type Description': 'IEC-LSRP',
```

Timestamp	Action description	Screen capture of video

0:26 – 0:40	- Video: Scans through the data and pauses at the pictured. Note the re-occurring site with the same PI number.
	- "Definitely looking at data."
		
		{ County: [redacted]',
		  Municipality[redacted]',
		  'PI Name': '[redacted]
		  'Line1 Address': '[redacted],
		  'PI Number': '[redacted]598',
		  'Activity Number (CF)': '[redacted]0003',
		  'Identification Date (SI)': '9/22/2011',
		  'IEC Type (SI)': 'Vapor Intrusion',
		  'Activity Type Description': 'IEC-VC-PF',
		  'Case Manager': '[redacted]',
		  'Unknown Source': 'No',
		  'Source Name': '[redacted]
		  'Source Control Status': '',
		  'Source Status Date (IEC)': '9/22/2011',
		  'Receptor\'s Name': 'IEC-VI PCE',
		  'Receptor Control Status': 'Controlled',
		  'Receptor Status Date': '',
		  'Receptor Type (IEC)': '' }
		{ County: '[redacted]',
		  Municipality[redacted]',
		  'PI Name': '[redacted]
		  'Line1 Address': '[redacted],
		  'PI Number': '[redacted]598',
		  'Activity Number (CF)': '[redacted]0001',
		  'Identification Date (SI)': '5/6/2013',
		  'IEC Type (SI)': 'Vapor Intrusion',
		  'Activity Type Description': 'PF - Operations & Maintenance (Non-Billable)',
		  'Case Manager': '[redacted]',
		  'Unknown Source': 'No',
		  'Source Name': '',
		  'Source Control Status': '',
		  'Source Status Date (IEC)': '',
		  'Receptor\'s Name': 'Dry Cleaners adjacent to residence',
		  'Receptor Control Status': 'Controlled',
		  'Receptor Status Date': '12/9/2013',
		  'Receptor Type (IEC)': '' }
		{ County: 'Passaic',

0:40 – 0:50	- Video: Navigates back to Atom and reviews the same code: `// Combine`	

Timestamp	Action description	Screen capture of video
		
0:50 – 1:12	- Video: Navigates to the IEC data set (CSV file), which is open in Calc spreadsheet program. - "So this is where I discover that there's multiple rows per site. I think that that's what I'd be trying to process there."	

combine.js - /User[redacted]

census-sites-buffer.js .gitignore aband

```
_.each(immediate, function(i) {
  var pi = i["PI Number"];
  var existing = {};

  // Check if already there
  if (combined[pi]) {
    existing = combined[pi];
    abandonedImmediate++;
    console.log(i);
  }

  // Combine
  combined[pi] = _.extend(existing, {
    county: i.County,
    municipality: i.Municipality,
    name: i["PI Name"],
    immediateName: i["PI Name"],
    street: i["Line1 Address"],
    
  });


{[redacted]
```

OpenOffice File Edit View Insert Format Tools Data Window Help
Wed 1:53 PM
Find
Arial
10

Immedicate Environment Concern Sites - Data [redacted].csv - OpenOffice Calc

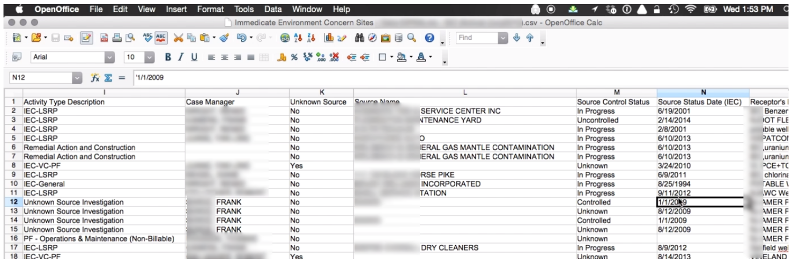
G13 '7/18/2003

	C	D	E	F	G	H	
1	PI Name	Line1 Address	PI Number	Activity Number (CF)	Identification Date (SI)	IEC Type (SI)	Activity T
2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-LSR
3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-LSR
4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-LSR
5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-LSR
6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Direct Contact	Remedia
7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Direct Contact	Remedia
8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Vapor Intrusion	IEC-VC-I
9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Vapor Intrusion	IEC-LSR
10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-Gen
11	SH[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-LSR
12	[redacted] PUMP STATION	[redacted]	[redacted]36	[redacted]100001	7/18/2003	Potable Well Contamination	Unknowr
13	[redacted] PUMP STATION	[redacted]	[redacted]36	[redacted]100001	7/18/2003	Potable Well Contamination	Unknowr
14	[redacted] PUMP STATION	[redacted]	[redacted]36	[redacted]100001	7/18/2003	Potable Well Contamination	Unknowr
15	[redacted] PUMP STATION	[redacted]	[redacted]36	[redacted]100001	7/18/2003	Potable Well Contamination	Unknowr
16	[redacted]	[redacted]	[redacted]79	[redacted]090001	7/18/2003	Potable Well Contamination	PF - Ope
17	[redacted]DRY CLEANERS	[redacted]	[redacted]35	[redacted]A120001	10/5/1992	Potable Well Contamination	IEC-LSR
18	[redacted] SEWER UTILITY SERVICE	[redacted]	[redacted]99	[redacted]130001	8/14/2013	Potable Well Contamination	IEC-VC-I

<table>
<tr><th>Timestamp</th><th>Action description</th><th>Screen capture of video</th></tr>
<tr><td>1:12 – 1:25</td><td>

- Video: Scrolls horizontally across the particular rows of interest to see the other columns.
- "But the multiple rows doesn't mean new information; it just means there's different information in each row -- slightly different."

</td><td>



Immedicate Environment Concern Sites [redacted].csv - OpenOffice Calc

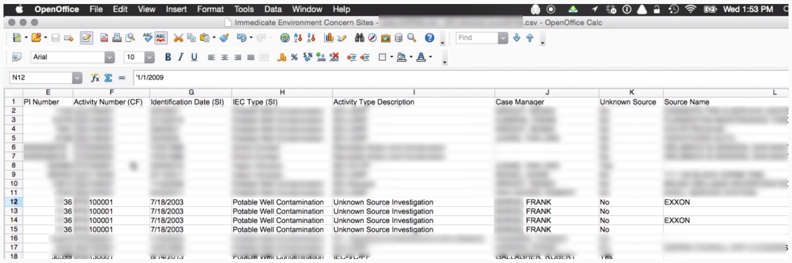
N12 '1/1/2009

<table>
<tr><th></th><th>I</th><th>J</th><th>K</th><th>L</th><th>M</th><th>N</th><th></th></tr>
<tr><td>1</td><td>Activity Type Description</td><td>Case Manager</td><td>Unknown Source</td><td>Source Name</td><td>Source Control Status</td><td>Source Status Date (IEC)</td><td>Receptor's</td></tr>
<tr><td>2</td><td>IEC-LSRP</td><td>[redacted]</td><td>No</td><td>[redacted] SERVICE CENTER INC</td><td>In Progress</td><td>6/19/2001</td><td>[redacted] Benzer</td></tr>
<tr><td>3</td><td>IEC-LSRP</td><td>[redacted]</td><td>No</td><td>[redacted] NTENANCE YARD</td><td>Uncontrolled</td><td>2/14/2014</td><td>[redacted] OT FLE</td></tr>
<tr><td>4</td><td>IEC-LSRP</td><td>[redacted]</td><td>No</td><td>[redacted]</td><td>In Progress</td><td>2/8/2001</td><td>[redacted] ble well</td></tr>
<tr><td>5</td><td>IEC-LSRP</td><td>[redacted]</td><td>No</td><td>[redacted] O</td><td>In Progress</td><td>6/10/2013</td><td>[redacted] PATCO</td></tr>
<tr><td>6</td><td>Remedial Action and Construction</td><td></td><td>No</td><td>[redacted] ERAL GAS MANTLE CONTAMINATION</td><td>In Progress</td><td>6/10/2013</td><td>[redacted] ,uraniun</td></tr>
<tr><td>7</td><td>Remedial Action and Construction</td><td></td><td>No</td><td>[redacted] ERAL GAS MANTLE CONTAMINATION</td><td>In Progress</td><td>6/10/2013</td><td>[redacted] ,uraniun</td></tr>
<tr><td>8</td><td>IEC-VC-PF</td><td>[redacted]</td><td>Yes</td><td></td><td>Unknown</td><td>3/24/2010</td><td>[redacted] PCE+TC</td></tr>
<tr><td>9</td><td>IEC-LSRP</td><td>[redacted]</td><td>No</td><td>[redacted] RSE PIKE</td><td>In Progress</td><td>6/9/2011</td><td>[redacted] chlorina</td></tr>
<tr><td>10</td><td>IEC-General</td><td>[redacted]</td><td>No</td><td>[redacted] INCORPORATED</td><td>In Progress</td><td>8/25/1994</td><td>[redacted] TABLE \</td></tr>
<tr><td>11</td><td>IEC-LSRP</td><td>[redacted]</td><td>No</td><td>[redacted] TATION</td><td>In Progress</td><td>9/11/2012</td><td>[redacted] WC We</td></tr>
<tr><td>12</td><td>Unknown Source Investigation</td><td>[redacted] FRANK</td><td>No</td><td>[redacted]</td><td>Controlled</td><td>1/1/2009</td><td>[redacted] AMER F</td></tr>
<tr><td>13</td><td>Unknown Source Investigation</td><td>[redacted] FRANK</td><td>No</td><td></td><td>Unknown</td><td>8/12/2009</td><td>[redacted] AMER F</td></tr>
<tr><td>14</td><td>Unknown Source Investigation</td><td>[redacted] FRANK</td><td>No</td><td>[redacted]</td><td>Controlled</td><td>1/1/2009</td><td>[redacted] AMER F</td></tr>
<tr><td>15</td><td>Unknown Source Investigation</td><td>[redacted] FRANK</td><td>No</td><td></td><td>Unknown</td><td>8/12/2009</td><td>[redacted] AMER F</td></tr>
<tr><td>16</td><td>PF - Operations &amp; Maintenance (Non-Billable)</td><td>[redacted]</td><td>No</td><td></td><td>Unknown</td><td></td><td>[redacted] AMER F</td></tr>
<tr><td>17</td><td>IEC-LSRP</td><td>[redacted]</td><td>No</td><td>[redacted] DRY CLEANERS</td><td>In Progress</td><td>8/9/2012</td><td>[redacted] field we</td></tr>
<tr><td>18</td><td>IEC-VC-PF</td><td>[redacted]</td><td>Yes</td><td></td><td>Unknown</td><td>8/14/2013</td><td>[redacted] ELAND</td></tr>
</table>

</td></tr>
<tr><td>1:25 – 1:45</td><td>

- Video: Continues to scroll horizontally across the particular rows of interest to see the other columns.
- Hovers around the Activity Number column for a moment before switching back to Atom.

</td><td>



Immedicate Environment Concern Sites - [redacted].csv - OpenOffice Calc

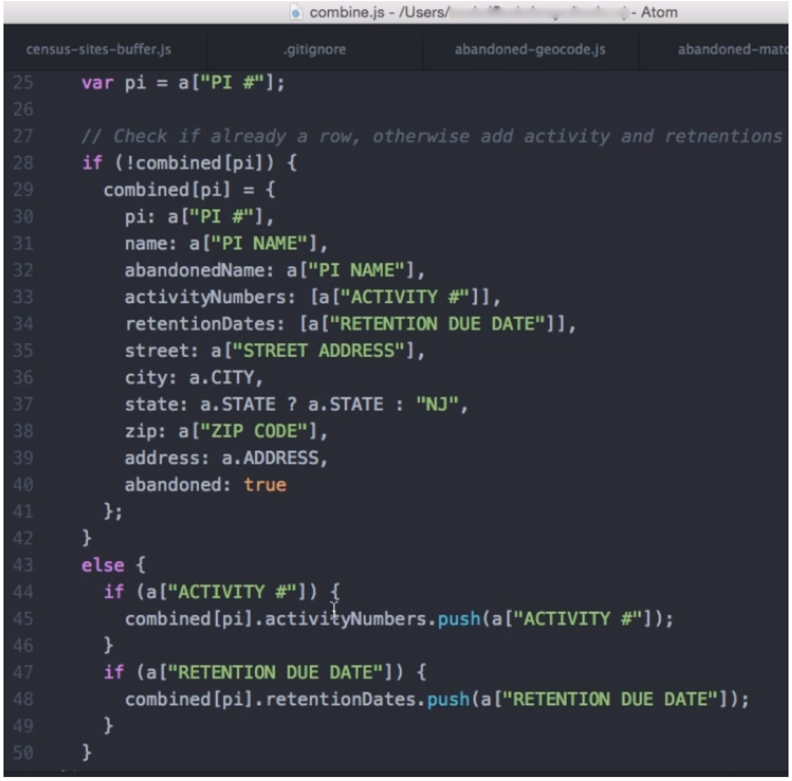
N12 '1/1/2009

<table>
<tr><th></th><th>E</th><th>F</th><th>G</th><th>H</th><th>I</th><th>J</th><th>K</th><th>L</th></tr>
<tr><td>1</td><td>PI Number</td><td>Activity Number (CF)</td><td>Identification Date (SI)</td><td>IEC Type (SI)</td><td>Activity Type Description</td><td>Case Manager</td><td>Unknown Source</td><td>Source Name</td></tr>
<tr><td>2</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td></tr>
<tr><td>3</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td></tr>
<tr><td>4</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td></tr>
<tr><td>5</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td></tr>
<tr><td>6</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td></td><td>[redacted]</td><td>[redacted]</td></tr>
<tr><td>7</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td></td><td>[redacted]</td><td>[redacted]</td></tr>
<tr><td>8</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td></td></tr>
<tr><td>9</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td></tr>
<tr><td>10</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td></tr>
<tr><td>11</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td></tr>
<tr><td>12</td><td>[redacted] 36</td><td>[redacted] 100001</td><td>7/18/2003</td><td>Potable Well Contamination</td><td>Unknown Source Investigation</td><td>[redacted] FRANK</td><td>No</td><td>EXXON</td></tr>
<tr><td>13</td><td>[redacted] 36</td><td>[redacted] 100001</td><td>7/18/2003</td><td>Potable Well Contamination</td><td>Unknown Source Investigation</td><td>[redacted] FRANK</td><td>No</td><td></td></tr>
<tr><td>14</td><td>[redacted] 36</td><td>[redacted] 100001</td><td>7/18/2003</td><td>Potable Well Contamination</td><td>Unknown Source Investigation</td><td>[redacted] FRANK</td><td>No</td><td>EXXON</td></tr>
<tr><td>15</td><td>[redacted] 36</td><td>[redacted] 100001</td><td>7/18/2003</td><td>Potable Well Contamination</td><td>Unknown Source Investigation</td><td>[redacted] FRANK</td><td>No</td><td></td></tr>
<tr><td>16</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td></td></tr>
<tr><td>17</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td><td>[redacted]</td></tr>
<tr><td>18</td><td>[redacted]</td><td>[redacted]</td><td>8/14/2013</td><td>Potable Well Contamination</td><td>IEC-VC-PF</td><td>[redacted]</td><td>Yes</td><td></td></tr>
</table>

</td></tr>
<tr><td>1:45 – 2:20</td><td>

- Video: Scrolls up to the `Activity #` data point within the conditional for the abandoned data.
- "This bit of code is saying, 'If there's one row we've already created, only add the activity number or retention due date," because those are the things that are essentially different information, or changing per site -- per row."

</td><td>

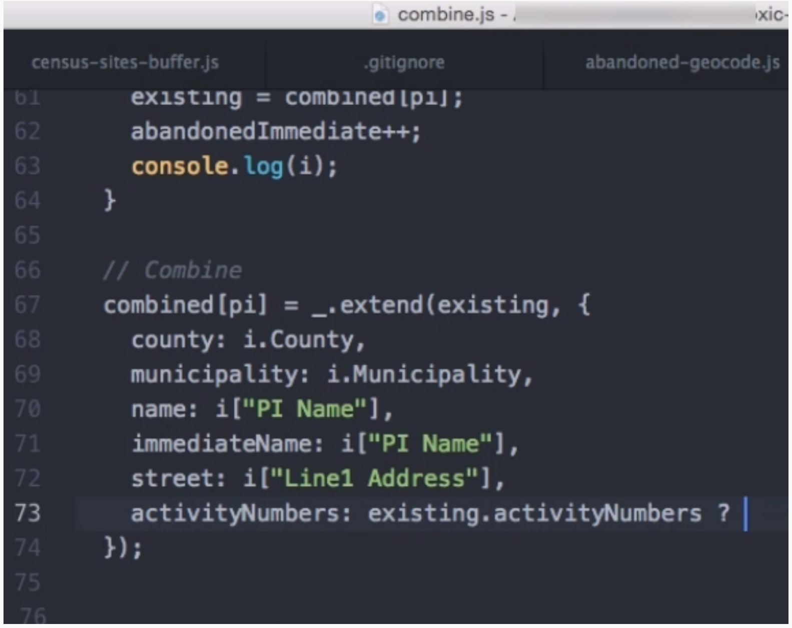


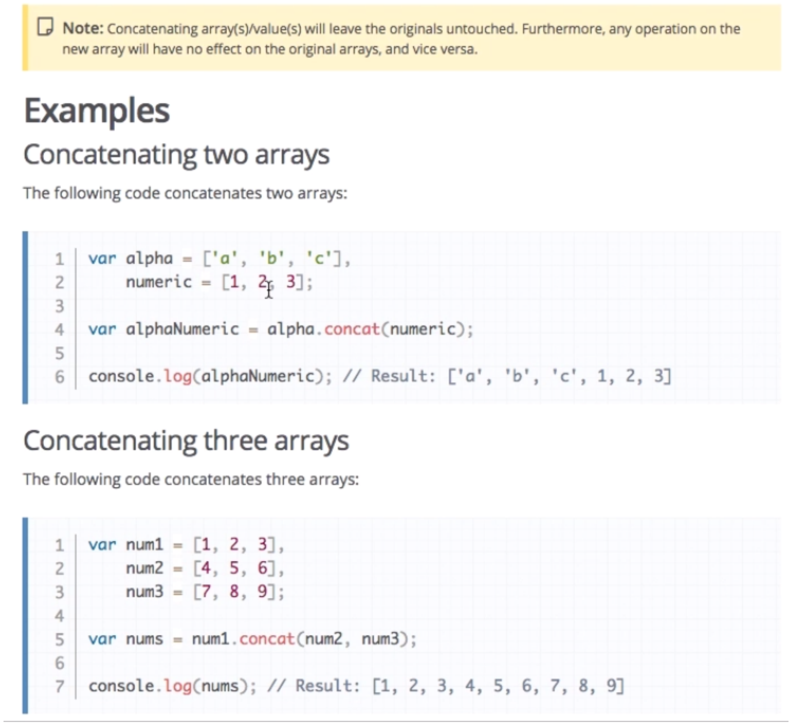
combine.js - /Users/[redacted] - Atom

```
  var pi = a["PI #"];

  // Check if already a row, otherwise add activity and retnentions
  if (!combined[pi]) {
    combined[pi] = {
      pi: a["PI #"],
      name: a["PI NAME"],
      abandonedName: a["PI NAME"],
      activityNumbers: [a["ACTIVITY #"]],
      retentionDates: [a["RETENTION DUE DATE"]],
      street: a["STREET ADDRESS"],
      city: a.CITY,
      state: a.STATE ? a.STATE : "NJ",
      zip: a["ZIP CODE"],
      address: a.ADDRESS,
      abandoned: true
    };
  }
  else {
    if (a["ACTIVITY #"]) {
      combined[pi].activityNumbers.push(a["ACTIVITY #"]);
    }
    if (a["RETENTION DUE DATE"]) {
      combined[pi].retentionDates.push(a["RETENTION DUE DATE"]);
    }
  }
```

</td></tr>
</table>

Timestamp	Action description	Screen capture of video

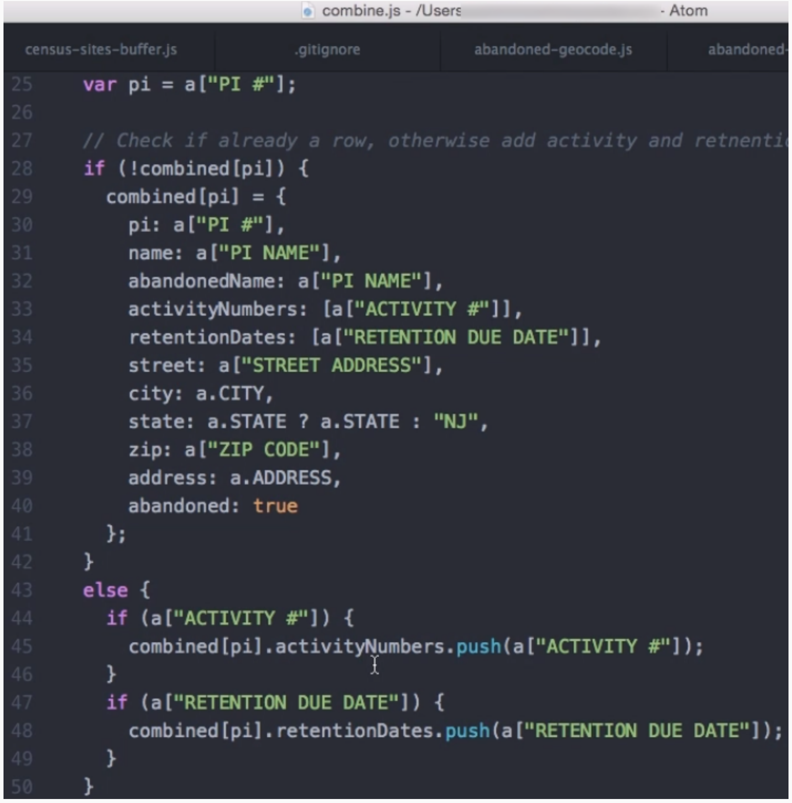
2:20 – 2:36	• Video: Scrolls down to the `Activity #` data point within the `_.extend()` method for to combine the abandoned data with the IEC data (line 73).
	• Video: Begins to code line 73: `activityNumbers:` . Copies and pastes `existing` from line 62 (knows that this is the abandoned data from above). Adds the new value to the `activityNumbers` key: `existing.activityNumbers` , then a `?` to start a ternary conditional statement.
		 combine.js -
		census-sites-buffer.js
		.gitignore
		abandoned-geocode.js
		`61 existing = combined[pi];`
		`62 abandonedImmediate++;`
		`63 console.log(i);`
		`64 }`
		`65`
		`66 // Combine`
		`67 combined[pi] = _.extend(existing, {`
		`68 county: i.County,`
		`69 municipality: i.Municipality,`
		`70 name: i["PI Name"],`
		`71 immediateName: i["PI Name"],`
		`72 street: i["Line1 Address"],`
		`73 activityNumbers: existing.activityNumbers ?`
		`74 });`
		`75`
		`76`

2:36 – 2:56	• Video: Navs to Chrome browser and Googles "js concat"; Clicks on Mozilla Developer Network documentation for the JS function.
	• "This is me looking up the `concat` function for arrays."
		
		Note: Concatenating array(s)/value(s) will leave the originals untouched. Furthermore, any operation on the new array will have no effect on the original arrays, and vice versa.
		Examples
		Concatenating two arrays
		The following code concatenates two arrays:
		`var alpha = ['a', 'b', 'c'],`
		`numeric = [1, 2, 3];`
		
		`var alphaNumeric = alpha.concat(numeric);`
		
		`console.log(alphaNumeric); // Result: ['a', 'b', 'c', 1, 2, 3]`
		Concatenating three arrays
		The following code concatenates three arrays:
		`var num1 = [1, 2, 3],`
		`num2 = [4, 5, 6],`
		`num3 = [7, 8, 9];`
		
		`var nums = num1.concat(num2, num3);`
		
		`console.log(nums); // Result: [1, 2, 3, 4, 5, 6, 7, 8, 9]`

<table>
<tr><th>Timestamp</th><th>Action description</th><th>Screen capture of video</th></tr>
<tr><td>3:01 – 3:54</td><td>

- I ask Ray to pause the video and ask: "Do you recall what prompted you to write that conditional?"
- Ray begins to navigate the timing of the video, remarking, "Umm. I gotta look around the code a bit to understand what's ... "
- Stops video at place pictured.
- Ray: "Right here it's saying, 'Do we have a row for this PI #?. If it doesn't create a new one, then add to the activity numbers.' That makes sense to me."

</td><td>

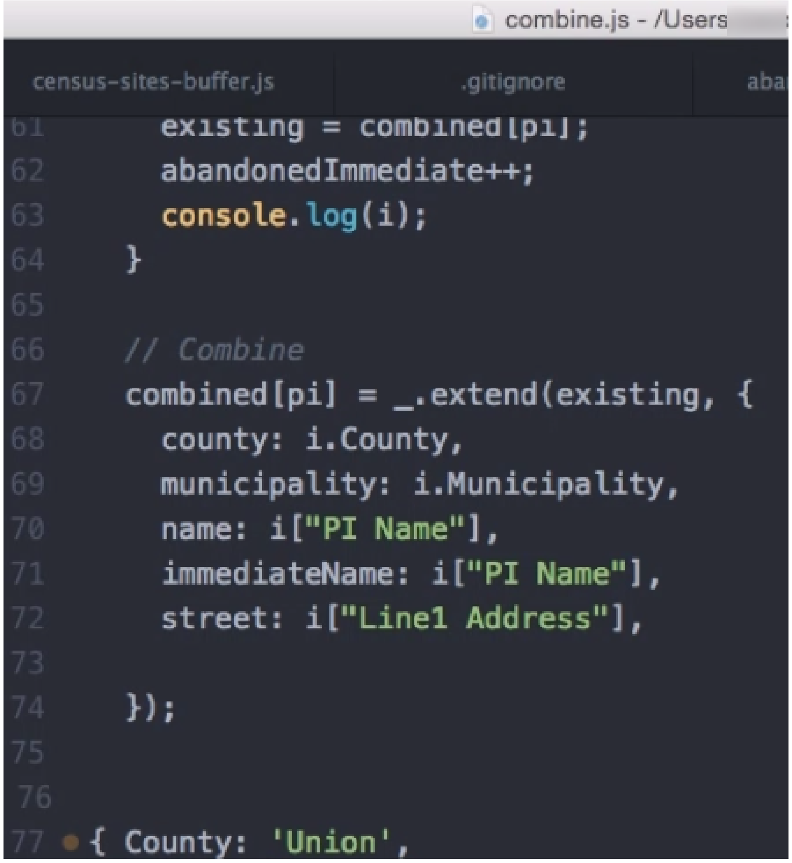


```
    var pi = a["PI #"];

    // Check if already a row, otherwise add activity and retnenti
    if (!combined[pi]) {
      combined[pi] = {
        pi: a["PI #"],
        name: a["PI NAME"],
        abandonedName: a["PI NAME"],
        activityNumbers: [a["ACTIVITY #"]],
        retentionDates: [a["RETENTION DUE DATE"]],
        street: a["STREET ADDRESS"],
        city: a.CITY,
        state: a.STATE ? a.STATE : "NJ",
        zip: a["ZIP CODE"],
        address: a.ADDRESS,
        abandoned: true
      };
    }
    else {
      if (a["ACTIVITY #"]) {
        combined[pi].activityNumbers.push(a["ACTIVITY #"]);
      }
      if (a["RETENTION DUE DATE"]) {
        combined[pi].retentionDates.push(a["RETENTION DUE DATE"]);
      }
    }
```

</td></tr>
<tr><td>3:54 – 4:26</td><td>

- Ray: What I don't understand a little bit [scrolls between abandoned and IEC data] is what this is doing. [See pictured code.] Oh, you know what, so this ...

</td><td>



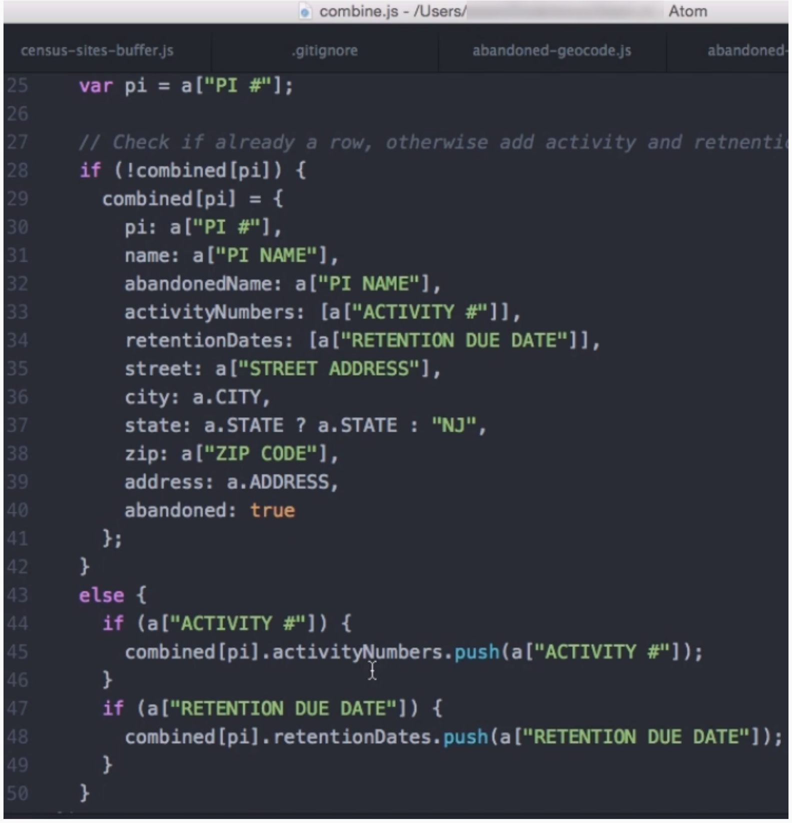
```
      existing = combined[pi];
      abandonedImmediate++;
      console.log(i);
    }

    // Combine
    combined[pi] = _.extend(existing, {
      county: i.County,
      municipality: i.Municipality,
      name: i["PI Name"],
      immediateName: i["PI Name"],
      street: i["Line1 Address"],

    });


{ County: 'Union',
```

</td></tr>
</table>

<table>
<tr><th>Timestamp</th><th>Action description</th><th>Screen capture of video</th></tr>
<tr><td>4:26 – 4:50</td><td>

- Ray: Yeah, I think this is the abandoned data. So we're combining those with the Immediate. So the abandoned, we're going through, we're saying the PI is the identifier, so if we have one, add to it, if not, create one. Then, I think there's a bit of code that we don't see.

</td><td>



```
  var pi = a["PI #"];

  // Check if already a row, otherwise add activity and retnenti
  if (!combined[pi]) {
    combined[pi] = {
      pi: a["PI #"],
      name: a["PI NAME"],
      abandonedName: a["PI NAME"],
      activityNumbers: [a["ACTIVITY #"]],
      retentionDates: [a["RETENTION DUE DATE"]],
      street: a["STREET ADDRESS"],
      city: a.CITY,
      state: a.STATE ? a.STATE : "NJ",
      zip: a["ZIP CODE"],
      address: a.ADDRESS,
      abandoned: true
    };
  }
  else {
    if (a["ACTIVITY #"]) {
      combined[pi].activityNumbers.push(a["ACTIVITY #"]);
    }
    if (a["RETENTION DUE DATE"]) {
      combined[pi].retentionDates.push(a["RETENTION DUE DATE"]);
    }
  }
```

</td></tr>
</table>

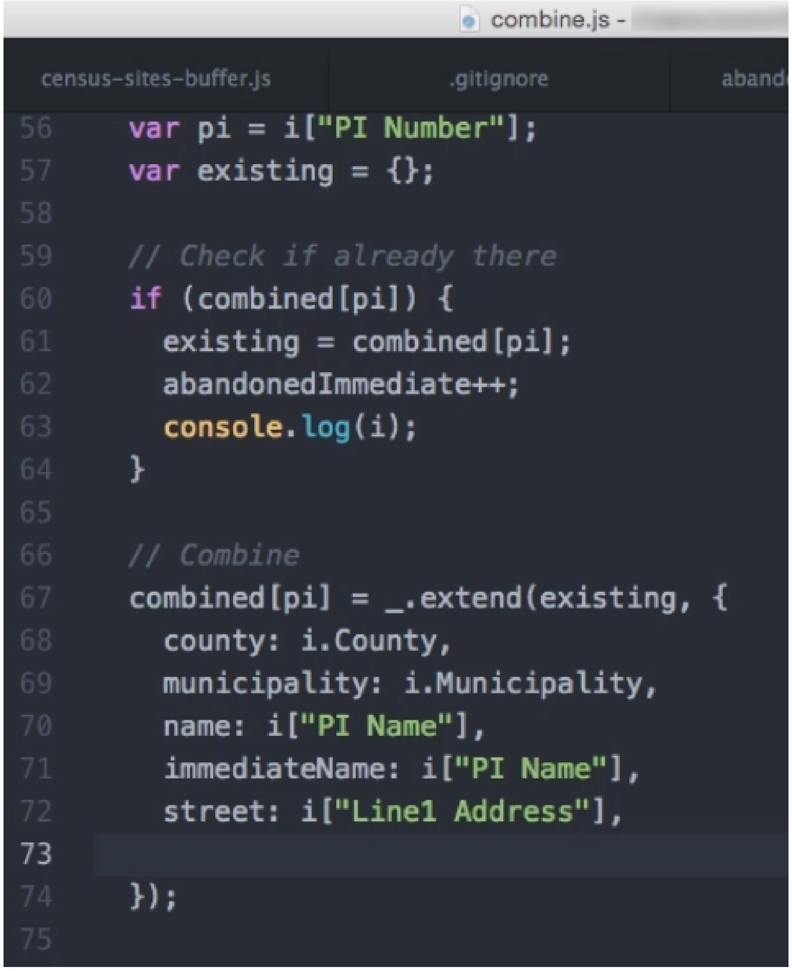
<table>
<tr><th>Timestamp</th><th>Action description</th><th>Screen capture of video</th></tr>
<tr><td>4:50 – 6:04</td><td>

- Video: Navigates the video and stops at the pictured code
- Ray: And then this [code] here [see pictured]; and it's hard to tell without seeing everything, but. I could probably bring it up, but well actually looking at this it has changed completely, so I'm not sure if this code exists. We could go back to the [git] logs, but ...

  I think this is then dealing with the Immediate sites, so what we're trying to do is something similar except ... So we're adding the data from the Immediate data set that isn't in the abandoned data set, so that's why we're extending it. And then the concatenation is to those activity numbers that we were adding to before, but we already know there's going to be something there.

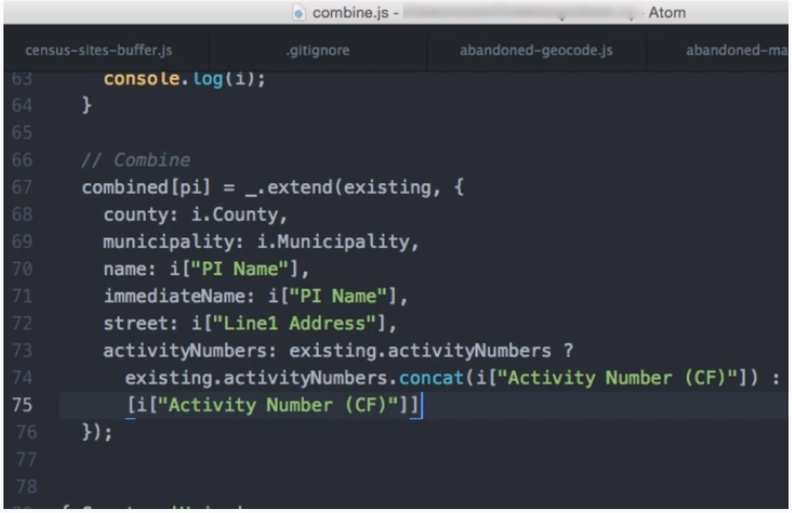
  [resumes playing of the video after navigating to where we last left off.]

</td><td>



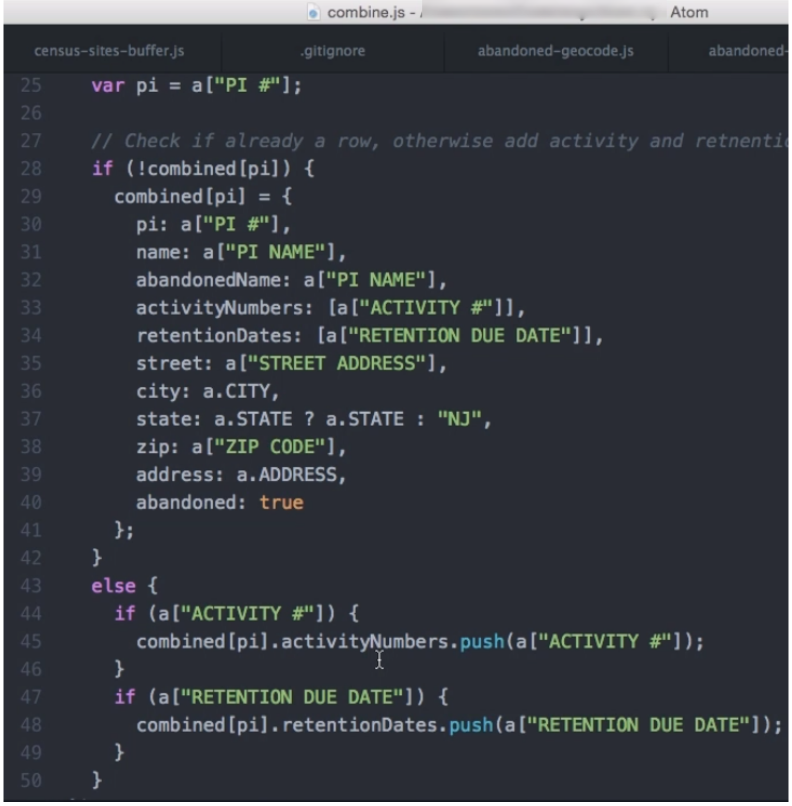
```
  var pi = i["PI Number"];
  var existing = {};

  // Check if already there
  if (combined[pi]) {
    existing = combined[pi];
    abandonedImmediate++;
    console.log(i);
  }

  // Combine
  combined[pi] = _.extend(existing, {
    county: i.County,
    municipality: i.Municipality,
    name: i["PI Name"],
    immediateName: i["PI Name"],
    street: i["Line1 Address"],

  });

```

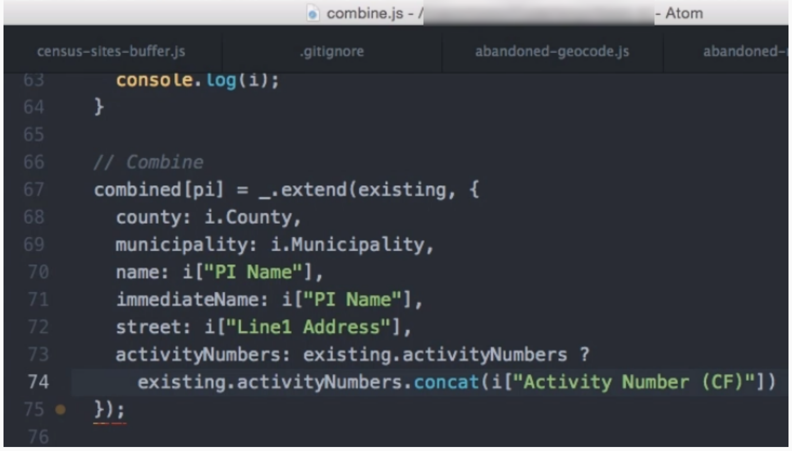
</td></tr>
</table>

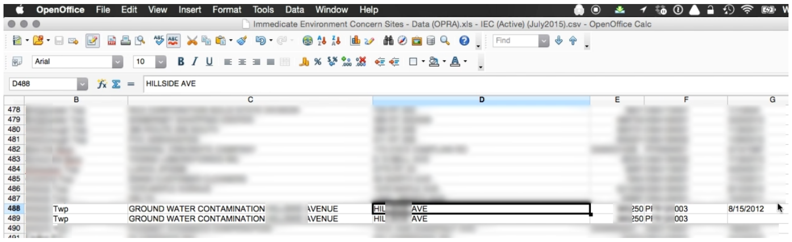
Timestamp	Action description	Screen capture of video
6:04 – 6:26	- Video: Ray writes out ternary with the concat function. - Ray: Well, we don't, but if there is something there, then we're going to add to that array. And if not, I assume we create a new array ... [watches him finish writing the code]. Yeah.	

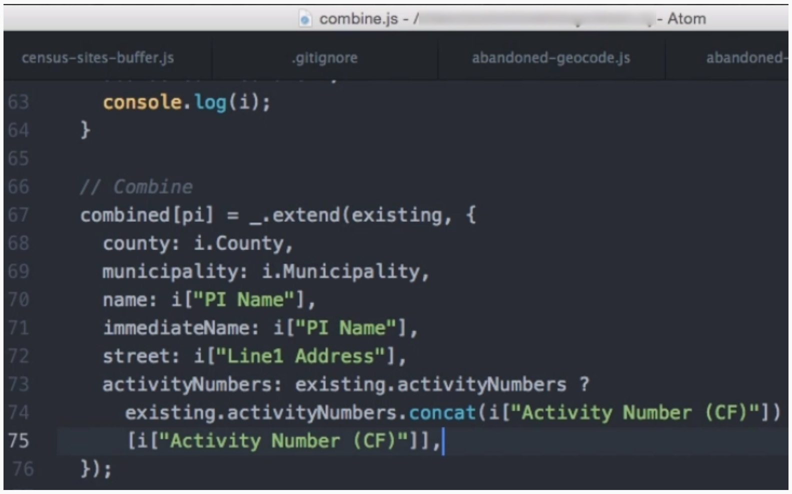
```
    console.log(i);
  }

  // Combine
  combined[pi] = _.extend(existing, {
    county: i.County,
    municipality: i.Municipality,
    name: i["PI Name"],
    immediateName: i["PI Name"],
    street: i["Line1 Address"],
    activityNumbers: existing.activityNumbers ?
      existing.activityNumbers.concat(i["Activity Number (CF)"]) :
      [i["Activity Number (CF)"]]
  });
```

Timestamp	Action description	Screen capture of video

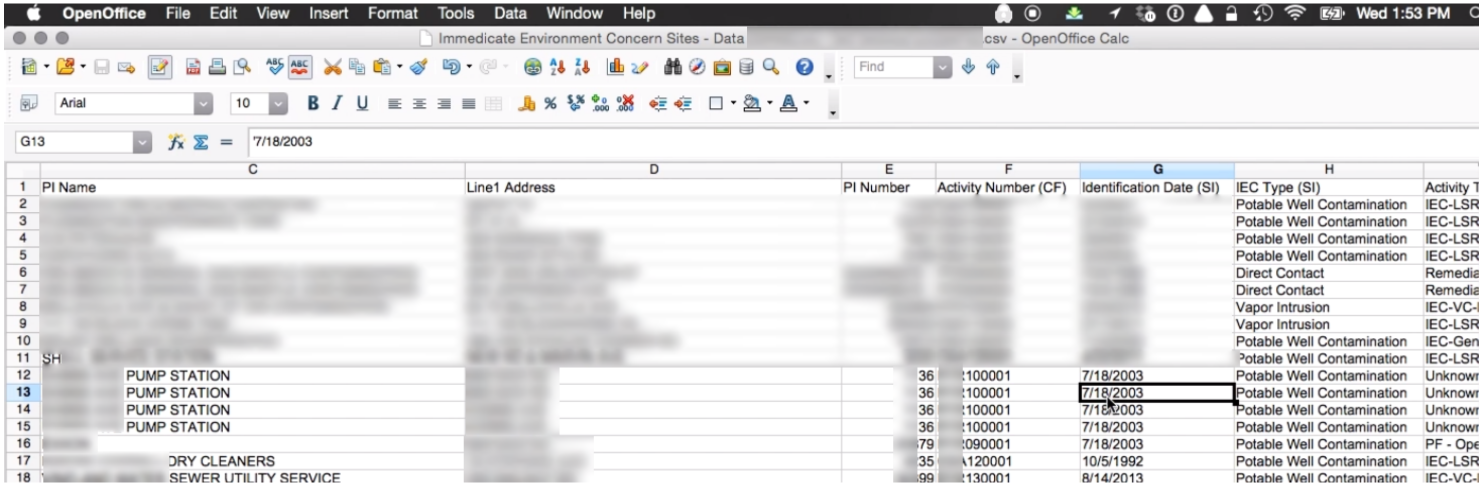
6:26 – 8:00	• Video: Ray pauses the video.
	• Ray: I'm curious. [Navigates back to the abandoned data array object.] You know, in reality, it's (Immediate data set ternary conditional) actually the same code as this (abandoned data set conditionals) -- it does the same thing as these lines (43-49). But I don't know why I did it differently. Maybe I changed this later? Well, one of the reasons I probably -- the difference here is that we're creating one object. Here (the conditional) we're doing a little more, err, a little less, err, well we're not creating a big object like this, we're creating 2 smaller things. But either way, they're essentially doing the same thing.
	
	I'm not entirely sure why [Navigates back to the Immediate data begin combined with the abandoned.]
		
		combine.js - / [redacted] Atom
		census-sites-buffer.js · .gitignore · abandoned-geocode.js · abandoned-
		`25    var pi = a["PI #"];`
		`26`
		`27    // Check if already a row, otherwise add activity and retnenti`
		`28    if (!combined[pi]) {`
		`29      combined[pi] = {`
		`30        pi: a["PI #"],`
		`31        name: a["PI NAME"],`
		`32        abandonedName: a["PI NAME"],`
		`33        activityNumbers: [a["ACTIVITY #"]],`
		`34        retentionDates: [a["RETENTION DUE DATE"]],`
		`35        street: a["STREET ADDRESS"],`
		`36        city: a.CITY,`
		`37        state: a.STATE ? a.STATE : "NJ",`
		`38        zip: a["ZIP CODE"],`
		`39        address: a.ADDRESS,`
		`40        abandoned: true`
		`41      };`
		`42    }`
		`43    else {`
		`44      if (a["ACTIVITY #"]) {`
		`45        combined[pi].activityNumbers.push(a["ACTIVITY #"]);`
		`46      }`
		`47      if (a["RETENTION DUE DATE"]) {`
		`48        combined[pi].retentionDates.push(a["RETENTION DUE DATE"]);`
		`49      }`
		`50    }`

8:00 – 8:30	• Video: Ray looks at the extension of the abandoned data with the Immediate data set.
	• Ray: I think the main reason I did it this way is because we're doing it inside the object itself. [hovers mouse over code of concern.] And so we can do an if statement around those things.
		
		combine.js - / [redacted] - Atom
		census-sites-buffer.js · .gitignore · abandoned-geocode.js · abandoned-
		`63      console.log(i);`
		`64    }`
		`65`
		`66    // Combine`
		`67    combined[pi] = _.extend(existing, {`
		`68      county: i.County,`
		`69      municipality: i.Municipality,`
		`70      name: i["PI Name"],`
		`71      immediateName: i["PI Name"],`
		`72      street: i["Line1 Address"],`
		`73      activityNumbers: existing.activityNumbers ?`
		`74        existing.activityNumbers.concat(i["Activity Number (CF)"])`
		`75    });`
		`76`

Timestamp	Action description	Screen capture of video
8:30 – 9:30	- Video: Video still running and now Ray is reading the IEC data set in Calc in the video. - Ray: Again, I'm just looking for different information for the same site and what those look like. [Watches himself read the data set more in Calc] Again, just trying to determine which fields [in the CSV data set] are multiple fields -- have multiple values. So, I'm looking at the data to visually see it and then coding it to an array, essentially, if it needs to be.	 Immedicate Environment Concern Sites - Data (OPRA).xls - IEC (Active) (July2015).csv - OpenOffice Calc D488 HILLSIDE AVE 478–487 [redacted] 488 [redacted] Twp GROUND WATER CONTAMINATION [redacted] AVENUE HIL [redacted] AVE [redacted] 250 PI [redacted] 003 8/15/2012 489 [redacted] Twp GROUND WATER CONTAMINATION [redacted] AVENUE HIL [redacted] AVE [redacted] 250 PI [redacted] 003 490 [redacted]

9:30 – 9:45	- Video: Switches to Atom and completes the ternary conditional.
	- Ray: ... So, I'm looking at the data to visually see it and then coding it to an array, essentially, if it needs to be.
		
		combine.js - / [redacted] - Atom
		census-sites-buffer.js .gitignore abandoned-geocode.js abandoned-
		`    console.log(i);`
		`  }`
		
		`  // Combine`
		`  combined[pi] = _.extend(existing, {`
		`    county: i.County,`
		`    municipality: i.Municipality,`
		`    name: i["PI Name"],`
		`    immediateName: i["PI Name"],`
		`    street: i["Line1 Address"],`
		`    activityNumbers: existing.activityNumbers ?`
		`      existing.activityNumbers.concat(i["Activity Number (CF)"])`
		`      [i["Activity Number (CF)"]],`
		`  });`

A.14 IEC toxic site data rows 12 and 13



OpenOffice File Edit View Insert Format Tools Data Window Help
Wed 1:53 PM
Immedicate Environment Concern Sites - Data [redacted] csv - OpenOffice Calc
Find
Arial
10
G13

7/18/2003

	C	D	E	F	G	H	
1	PI Name	Line1 Address	PI Number	Activity Number (CF)	Identification Date (SI)	IEC Type (SI)	Activity T
2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-LSR
3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-LSR
4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-LSR
5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-LSR
6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Direct Contact	Remedia
7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Direct Contact	Remedia
8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Vapor Intrusion	IEC-VC-I
9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Vapor Intrusion	IEC-LSR
10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-Gen
11	SH [redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Potable Well Contamination	IEC-LSR
12	[redacted] PUMP STATION	[redacted]	[redacted]36	[redacted]100001	7/18/2003	Potable Well Contamination	Unknown
13	[redacted] PUMP STATION	[redacted]	[redacted]36	[redacted]100001	7/18/2003	Potable Well Contamination	Unknown
14	[redacted] PUMP STATION	[redacted]	[redacted]36	[redacted]100001	7/18/2003	Potable Well Contamination	Unknown
15	[redacted] PUMP STATION	[redacted]	[redacted]36	[redacted]100001	7/18/2003	Potable Well Contamination	Unknown
16	[redacted]	[redacted]	[redacted]79	[redacted]090001	7/18/2003	Potable Well Contamination	PF - Ope
17	[redacted]DRY CLEANERS	[redacted]	[redacted]35	[redacted]120001	10/5/1992	Potable Well Contamination	IEC-LSR
18	[redacted] SEWER UTILITY SERVICE	[redacted]	[redacted]99	[redacted]130001	8/14/2013	Potable Well Contamination	IEC-VC-I

Figure A.11: IEC toxic site data rows 12 and 13.