6. Number of Grown Sheep That Were Sheared

> When knowledge passes into code, it changes state; like water turned to ice, it becomes a new thing, with new properties. We *use* it; but in a human sense we no longer *know* it.
>
> —ELLEN ULLMAN

The point of data's collection has been called its genesis moment. This is apt, not so much in a biblical sense as in a biological one: from the point of collection on, there are many messy cleavings and bubbling transformations ahead. By the time we act on a piece of data (or it acts on us), it may be nearly unrecognizable as the modest record we once collected. A large part of this transfiguration will come as a necessity of digitization. As a record of a real-world thing is reformatted and trimmed and parsed to fit into a computer, it changes, often in ways that affect how stories might be told and decisions may be made. Meanwhile, as the record is munged and refactored and set forth into a computational wilderness, the real-world thing itself might have changed, forcing us and our databases to reckon with decisions about accurate (and timely) representation.

Remember that the amount of data that can be conjured from any given thing is almost limitless. Pick up a plain gray rock

from the side of the road, and play the same data-ing game we did with the bird in chapter 2. Very quickly you'll have assembled a set of descriptors and values: size, weight, color, texture, shape, material. If you take that rock to a laboratory, these data can be made greatly more precise, and instrumentation beyond our own human sensorium can add to the list of records: temperature, chemical composition, carbon date. From there comes a kind of fractal unfolding of information that begins to occur, where each of these records in turn manifests its own data. The time at which the measurement was made, the instrument used to record it, the person who performed the task, the place where the analysis was performed. In turn, each of these new metadata records can carry its own data: the age of the person who performed the task, the model of the instrument, the temperature of the room. Data begets data, which begets metadata, repeat, repeat, repeat. It's data all the way down.

This infinity mirror of data and metadata can be exhausting for people who are trying to decide exactly what to record about a thing. Right now, a cataloger at a library holds an old book in their hands that has just arrived as a donation. There are so very many things that they could enter into the catalog about the book, for a book is a thing with a particular penchant for producing data. The number of pages, the type of binding, the typefaces used to set the text, the presence of a dust jacket—all of this before we get to whatever it is that the author might have to say. As a means of sanity for the cataloger and as a defense against bloated databases and overstuffed card catalogs, the library has defined a particular set of things that the cataloger should record about the book: the title, the author, the publisher, the year—all of the things we might expect to find listed in a bibliography. There is simply no means for our cataloger to add more data than what

is allowed: that the book in their hands smells vaguely of campfire smoke is of no business at all to the database.

The Library of Congress is, technically, what it says on the package. That is, it is a library whose primary purpose is to serve members of Congress. It holds the world's largest collection of books, fourteen million photographs, five and a half million maps, miles of manuscripts, seven Stradivari, Walt Whitman's walking stick, the contents of Lincoln's pockets when he was assassinated—all there (at least in theory) in case a congressperson requests them. This is something like saying that the purpose of the body is to produce saliva or that Coney Island is a hot dog stand. The Library of Congress is the nation's de facto national library, and the largest chunk of its mighty archival and cataloging and librarian-ing armaments are aimed at the public. For the last 150 years, this public has mostly been academic researchers, who come to read and write (with their quietest pencils) in the library's twenty-one reading rooms. Since 2016, when Dr. Carla Hayden (previously the president of Baltimore's public library system) was appointed to the position of librarian of Congress, the institution has been slowly tacking toward a new course—one aimed away from quiet research and to a louder, more gregarious type of learning.

I spent most of 2017 and 2018 at the library finding things, coming up with new ways to find things, and talking to librarians, archivists, and historians about things they'd already found. I was there as the institution's first innovator in residence (a title they made up), tasked to come up with new ways that the public might engage with the library's vast holdings. I approached the problem in a decidedly un-computational way, by having as many conversations with as many library staff as I could. Under the auspices of recording a podcast called *Artist in the Archive*, I met

with librarians and archivists, conservators and researchers, technologists and administrators.

Somehow all of these conversations kept coming back to this moment of data genesis, when a book or a map or an audio recording produces a catalog record, for it is in this moment when the object's life as a thing that might be found is largely defined. If the cataloger has a bit of time, and if they are thorough, our object might get a record that will serve it well for future searches. An exact date, an accurate place, an extra line of descriptive text: all of these things privilege the object greatly in the context of being found, they increase its chances of being included in research and in the stories that get told about the past. Conversely, many objects are data-fied with a spareness that all but guarantees them a life on the last page of search results.

Sometimes, an object is fortunate enough to be rescued from search obscurity. Consider a lucky newspaper clipping from 1863, which for 156 years was mislabeled as an Arabic translation of Lincoln's Emancipation Proclamation. In 2019, a volunteer transcriber in Kuwait came upon the clipping on the LOC's crowdsourcing platform and sent a note to the library that the text was not in fact in Arabic, but perhaps Armenian. Research by the library staff later determined the text to be Neo-Aramaic, a language spoken by Assyrian and Chaldean Christians in Urmia, Iran. The next day, the clipping's catalog record was changed, and it is now the number one search result for "Neo-Aramaic." A century and a half later, it is just beginning its life as a findable thing.

And then there's the Sheep Census.

When Julie Miller started her job in the Manuscript Division of the Library of Congress, she set out to survey the particular archival terrain of her specialty, early American history. This was a daunting task: the library holds more than sixty million items in its Manuscript Division, stored in more than thirteen thousand

collections. Conservatively assuming it would take ten minutes to review a single document, all of the manuscript collections would take roughly a thousand years to comprehensively explore. Not having more than one lifetime, Miller decided to take a survey approach, using the library's online catalog to peruse the holdings.

One particular document, for whatever reason, caught her eye. It was labeled "Massachusetts Sheep Census, 1787." Apart from that, there was no other information about what it was or where it had come from. Miller went into the archive and found the document, stored in a flat file among miles and miles of Hollinger boxes neatly lined up on metal shelves. It was a stiff piece of paper measuring about sixteen inches square. On the front was a hand-drawn table with thirty columns and thirty rows, a spreadsheet created two centuries before Excel. On the back was a single note, written in an elaborate script: "No. 2 a List taken from the Year 1787."

While many of the column headings were illegible due to physical damage, those that can be read made it clear to Miller that the so-called census was in fact a tax document. Through research into local newspapers and genealogy databases, she found out the names on the document were not from Massachusetts but from a small town called Canterbury, Connecticut. At some point in the summer of 1787, an official had been dispatched to the town to count the inhabitants and to assess what they owned for the purposes of taxation. "There's no income tax at this point," Miller explains, "so they're being taxed on what were called their polls, their heads."

The details the document records about a household are specific: it divides the "polls"—people—into those aged twenty-five to seventy and those aged sixteen to twenty-one, because these age groups were to be taxed at different rates. There are columns for the number of livestock (pigs, oxen, bulls, steer, cows, and heifers) each person owned, as well as acreage and type of land. Only particular possessions are counted: clocks and silver

watches. "Only three of them own clocks," says Miller, "which is interesting because you would think that everyone would own a clock, but in fact in the eighteenth century a clock was a kind of a cutting-edge technological device, and in fact not many people had one, and in reality they didn't have much use for, so it was kind of a status item."

Because there was no income tax in America in 1787, earnings are not listed, but these specific questions about watches and clocks act as proxies for wealth. The questions themselves, as much as the data recorded, give us an insight into the unique politics of the time and place. That sheep, for example, are counted in their own individual column, tells us that there was some specific difference in purpose that tax collectors had for asking about them. As it turns out, sheep were tax deductible in 1787.

"This is a period when New England was beginning to think about how it might participate in the Industrial Revolution," Miller told me. George Washington had visited early textile factories in New England shortly after he became president. He and his Treasury secretary, Alexander Hamilton, saw promise in a burgeoning milling industry. Textile production needed mills, mills needed wool, and wool needed sheep. This kind of manufacturing wouldn't gain a foothold in Connecticut until the 1820s, but we find a sign of its provenance in the last column of this document: "Number of Grown Sheep That Were Sheared." "Sheep ownership," says Miller, "may have signaled a willingness to belong to a woolly vanguard, to be part of something new and potentially profitable."

Miller's investigations into the "Sheep Census" produced a kind of data rescue two in one. Not only is the document itself now findable under the correct headings (Connecticut, tax records, 1787), but some glimpse of the ordinary lives of these Canterburians is now also available to researchers. "Quantitative data is often the only written record of people who never learned to write, such as

slaves, the poor, and, in early America, many women," Miller writes. "In fact, quantitative data is often the only extant information about all kinds of people who went about their daily business making no special mark on history. That is to say, most people."

The proclamation that wasn't in Arabic, the sheep census that was neither: two objects that found their way out of data obscurity. Alongside them in the library's collection are millions of objects that will never be so lucky, items with absent or inaccurate dates or descriptions or subject headings. They are, in a system that prioritizes data, invisible, along with information they contain and the stories they might tell. What's crucial to understand with the sheep census is that both the yes/no binary choice of collecting data or not and the more nuanced decisions of how to store the data have greatly influenced how the data version of Canterbury, Connecticut, in 1787 matches to the real world at that same time and place.

=

In a 2015 essay about this mismatch between data's reality and the actual world, Jacob Harris conjures computer programmers who've been asked to create a database that tracks prisoners in a fictional kingdom called Zenda. These developers approach the problem in exactly the way you or I or anyone else faced with this problem might, by building a data structure—a schema—that fits their estimated version of reality. The programmers' database records the names and birth dates of the Zenda prisoners, along with a true/false value for whether each person is currently incarcerated, the idea being that upon release, captive = true would be changed to captive = false. These true/false values, called Booleans, are extremely convenient for programmers: first, because they take up very little space in a computer's memory; second,

because they can be operated on with ease. Boolean logic—the ways in which these true/false values can be added or multiplied or divided—is central to computer code. Almost any computer program will have instructions in which Booleans are evaluated, choose-your-own-adventure points in which the computer does different things depending on how combinations of these true/false values are evaluated. If captive = true, lock gate. If captive = false, open gate. In most data applications, subsets of information are often retrieved from the database using Boolean matches—SELECT * FROM prisoners WHERE captive = true.

Harris's hypothetical developers quickly realize that while the "captive" Boolean is convenient for the database, it doesn't mesh well with the reality of the prison. What if, for example, a prisoner has been apprehended but not yet imprisoned? On the other hand, what if they've been acquitted but not yet released? The programmers' first instinct is to add more Boolean values: "captured," "captive," "acquitted," "released." The developers lean back in their chairs, confident that reality has been modeled, that the true/false cascade they've built will tell the story of all of the prisoners. And then, to their dismay, a prisoner is acquitted, then released, then captured again. What are the poor Booleans to do?

Harris's essay is about how seemingly inconsequential coding decisions can have an extraordinary impact on data's capacity for communication. Specifically it's about how the precision imposed on data by computers can clash with what Harris called the "murky reality" of the real world. His writing underlines a critical point, often missed: that while computational bias can come from big decisions, it can also come from small ones. While we urgently need to be critical of the way machine learning systems are authored, we also need to pay attention to the impact of procedural minutiae—like whether a data point is being stored as a Boolean or as a number or as a piece of text.

I'd wondered since I read the piece whether Harris's Zenda prison example might be a metaphor for some real-world system, if the Boolean conundrums the hypothetical programmers faced might have come from some actual problem that Harris had worked on in *The New York Times*'s newsroom. Finally, two years after he left the *Times* to wrangle data for the public sector, Harris admitted as much in a tweet. A few months later I sat down with him to speak about Booleans and data structures and the world's most infamous military prison.

In November 2008, *The New York Times* published "The Guantánamo Docket," an interactive database of information about the roughly 780 people who have been (or still are) imprisoned at the Guantánamo Bay detention camp on the south shore of Cuba. This information had, since 9/11, been a missing data set, classified information that had not been released to the public. This changed in February 2010, when Chelsea Manning leaked 750,000 documents to WikiLeaks, including nearly 800 files concerning Guantánamo. The "Gitmo Files" contained assessments, interviews, and internal memos about detainees. Since the Manning leaks, "The Guantánamo Docket" has been updated through journalists' investigations and from documents received through FOIA requests.

The database for "The Guantánamo Docket" was, as Harris explained to me, built in nearly the same way as the fictive Zenda one—starting with a simple structure based on initial assumptions and then rebuilding and retooling from there. In a newsroom, under deadline, developers often need to leap before they learn. "You don't really know that much about the data, because you're about to load it because you just got a big CSV dump or you have a bunch of records you're going to scan. You don't have time to go through them, because you're operating on deadline time.

"You start with the most easily obtained cases first," Harris says. "Then you start hunting down the things that are harder to

find. You might have a record of all the most common prisoners that you know about, the ones that have been listed in press releases or things like that." At this point, the only real option to the developer is to build data structures around the data that they can see. Any cases in which prisoner data doesn't quite fit the structures are going to be, fingers firmly crossed, labeled as edge cases: strange one-offs that don't represent the bulk of the expected data. "Then you start hunting down other names," Harris tells me. "Those are going to be the ones that are usually a lot murkier in terms of what you can get, but also the ones you're going to encounter a lot later. You don't know when you find that, is this going to be the only edge case that's like this or are there going to be tens of them? Hundreds of them? Thousands of them?"

As an example, let's say a record is found for a prisoner from Guantánamo who doesn't have an exact birth date listed, just a year. How does this person fit into the schema that's been written, one which requires a birthday in MM-DD-YYYY format, such as 06-24-1982? "Do you give them the birth date of January 1, as a specific date that year? Or do you rewrite the database, breaking the date down into three parts: a year, a month, and a day?" The first way is easier, of course. But it depends on any future user of the database knowing that January 1 is a proxy for "we don't know." And what about prisoners who were actually New Year's babies? This is the start of a long three-way wrestling match, between the programmer's time, the computer's requirements, and the real world as it actually exists.

"You dream of one day throwing out the database and starting again with the one that will get it all right," Harris says. "But you know you have to maintain this one in the future." And so instead of redrafting code to fit better to reality, more often the data is redrafted to fit the expectations of the computer. In the real world of the newsroom, which in most ways matches the broader world of

programming and data science, this Guantánamo story is playing out every day, with databases and their rigid structures dictating what pieces of the real world are stored as bits and bytes, Booleans and floating point numbers and strings, and which pieces are discarded, trimmings of the computer's "cookie-cutter logic."

=

"I think a lot of computer systems and a lot of data entry systems work," Jacob Harris told me near the end of our interview, "because they train us to think like the computer." Understanding this encourages us to look at "The Guantánamo Docket" with a view for which parts of the story might have been cast to the computation margins by the needs of a database and the decisions of a programmer. Indeed we can ask the question anytime we read a news story or listen to a politician's speech: What is missing from a data story because it found itself outside a computer's uncompromising ways of thinking? Harris points to gender, and how it has often been handled as data by programmers in the past: as a male/female binary. "That's terrible if you're not someone who falls into either of those categories. What you're basically telling those people is you don't matter as a person."

We learned in the last chapters that data can bestow privilege and that its absence can push a thing out toward the margins. Here we see that the processes in which a thing is data-fied and the constraints of the structures made to hold that information can also have a profound effect on how that thing can participate in the databases and search engines, newspaper articles and court hearings, in the record of history.

This effect—where data is trimmed or transfigured to match the expectations of the machine—can be called schematic bias. For those involved in building data systems, a schema is a kind

of blueprint, a map of which types of information will be stored, in what form, and which types of information will be rejected. In cognitive science, a schema is a pattern of thought, a framework of preconceived ideas that directs how a person sees the world: if you observe something that fits neatly into your schema, it gets filed easily and efficiently into your memory. On the other hand, schema-foreign things will often not be noticed or remembered, or they will be modified to fit into what you expect based on the frameworks you've constructed. The many machines that order our data lives are working the same way, paying more attention to the things that fit neatly into their schema and ignoring things that don't—or changing them to fit.

It's particularly important to understand how schematic biases are amplified. How a decision made by a developer in a newsroom affects how a data point is stored, how a visualization is made, how a story is told, how a public understands. The structures built to store data affect how things are found and lost, how histories are written and who is included in them. Algorithms, with their expansionary tendencies, can loop these omissions upon themselves until they become wide sinkholes, affecting the ways in which people live (and lose) their data lives.