19. Accounting for Methods: Spreadsheets, Scripts and Programming Notebooks

Sam Leon

Abstract

This chapter explores the ways in which literate programming environments such as Jupyter Notebooks can help make data journalism reproducible, less error prone and more collaborative.

Keywords: Jupyter Notebooks, reproducibility, programming, Python, literate programming environments, data journalism

With the rise of data journalism, ideas around what can be considered a journalistic source are changing. Sources come in many forms now: Public data sets, leaked troves of emails, scanned documents, satellite imagery and sensor data. In tandem with this, new methods for finding stories in these sources are emerging. Machine learning, text analysis and some of the other techniques explored elsewhere in this book are increasingly being deployed in the service of the scoop.

But data, despite its aura of hard objective truth, can be distorted and misrepresented. There are many ways in which data journalists can introduce error into their interpretation of a data set and publish a misleading story. There could be issues at the point of data collection which prevent general inferences being made to a broader population. This could, for instance, be a result of a self-selection bias in the way a sample was chosen, something that has become a common problem in the age of Internet polls and surveys. Errors can also be introduced at the data-processing stage. Data processing or cleaning can involve geocoding, correcting misspelled names, harmonizing categories or excluding certain data points altogether if, for instance, they are considered statistical outliers. A good example of this kind of error at work is the inaccurate geocoding of IP addresses in a

widely reported study that purported to show a correlation between political persuasion and consumption of porn (Harris, 2014). Then, of course, we have the meat of the data journalist's work, analysis. Any number of statistical fallacies may affect this portion of the work, such as mistaking correlation with causation or choosing an inappropriate statistic to summarize the data set in question.

Given the ways in which collection, treatment and analysis of data can change a narrative—how does the data journalist reassure the reader that the sources they have used are reliable and that the work done to derive their conclusions is sound?

In the case that the data journalist is simply reporting the data or research findings of a third party, they need not deviate from traditional editorial standards adopted by many major news outlets. A reference to the institution that collected and analyzed the data is generally sufficient. For example, a recent *Financial Times* chart on life expectancy in the United Kingdom is accompanied by a note which says: "Source: Club Vita calculations based on Eurostat data." In principle, the reader can then make an assessment of the credibility of the institution quoted. While a responsible journalist will only report studies they believe to be reliable, the third-party institution is largely responsible for accounting for the methods through which it arrived at its conclusions. In an academic context, this will likely include processes of peer review and in the case of scientific publishing it will invariably include some level of methodological transparency.

In the increasingly common case where the journalistic organization produces the data-driven research, then they themselves are accountable to the reader for the reliability of the results they are reporting. Journalists have responded to the challenge of accounting for their methods in different ways. One common approach is to give a description of the general methodology used to arrive at the conclusions within a story. These descriptions should be framed as far as possible in plain, non-technical language so as to be comprehensible to the widest possible audience. A good example of this approach was taken by *The Guardian* and Global Witness in explaining how they counted deaths of environmental activists for their "Environmental Defenders" series (Leather, 2017; Leather & Kyte, 2017).

But—as with all ways of accounting for social life—written accounts have their limits. The most significant issue with them is that they generally do not specify the exact procedures used to produce the analysis or prepare the data. This makes it difficult, or in some cases impossible, to exactly reproduce steps taken by the reporters to reach their conclusions. In other words, a written account is generally not a reproducible one. In the example

above, where the data acquisition, processing and analysis steps are relatively straightforward, there may be no additional value in going beyond a general written description. However, when more complicated techniques are employed there may be a strong case for employing reproducible approaches.

Reproducible Data Journalism

Reproducibility is widely regarded as a pillar of the modern scientific method. It aids in the process of corroborating results and identifying and addressing problematic findings or questionable theories. In principle, the same mechanisms can help to weed out erroneous or misleading uses of data in the journalistic context.

A look at one of the most well-publicized methodological errors in recent academic history can be instructive. In a 2010 paper, Harvard's Carmen Reinhart and Kenneth Rogoff purposed to have shown that average real economic growth slows to -0.1% when a country's public debt rises to more than 90% of gross domestic product (Reinhart & Rogoff, 2010). This figure was then used as ammunition by politicians endorsing austerity measures.

As it turned out, the analysis was based on an Excel error. Rather than taking the mean of a whole row of countries, Reinhart and Rogoff had made an error in their formula which meant only 15 out of the 20 countries they looked at were incorporated. Once all the countries were considered the 0.1% "decline" became a 2.2% average increase in economic growth. The mistake was only picked up when PhD candidate Thomas Herndon and professors Michael Ash and Robert Pollin looked at the original spreadsheet that Reinhart and Rogoff had worked off. This demonstrates the importance of having not just the method written out in plain language—but also having the data and technology used for the analysis itself. But the Reinhart–Rogoff error perhaps points to something else as well—Microsoft Excel, and spreadsheet software in general, may not be the best technology for creating reproducible analysis.

Excel hides much of the process of working with data by design. Formulas—which do most of the analytical work in a spreadsheet—are only visible when clicking on a cell. This means that it is harder to review the actual steps taken to reaching a given conclusion. While we will never know for sure, one may imagine that had Reinhart and Rogoff's analytical work been done in a language in which the steps had to be declared explicitly (e.g., a programming language) the error could have been spotted prior to publication.

Excel-based workflows generally encourage the removal of the steps taken to arrive at a conclusion. Values rather than formulas are often copied across to other sheets or columns, leaving the “undo” key as the only route back to how a given number was actually generated. “Undo” histories, of course, are generally erased when an application is closed, and are therefore not a good place for storing important methodological information.

The Rise of the Literate Programming Environment: Jupyter Notebooks in the Newsroom

An emerging approach to methodological transparency is to use so-called “literate programming” environments. Organizations like *Buzzfeed*, *The New York Times* and Correctiv are using them to provide human-readable documents that can also be executed by a machine in order to reproduce exactly the steps taken in a given analysis.¹

First articulated by Donald Knuth in the 1980s, literate programming is an approach to writing computer code where the author intersperses code with ordinary human language explaining the steps taken (Knuth, 1992). The two main literate programming environments in use today are Jupyter Notebooks and R Markdown.² Both produce human-readable documents that mix plain English, visualizations and code in a single document that can be rendered in HTML and published on the web. Original data can be linked to explicitly, and any other technical dependencies such as third-party libraries will be clearly identified.

Not only is there an emphasis on human-readable explanation, the code is ordered so as to reflect human logic. Documents written in this paradigm can therefore read like a set of steps in an argument or a series of answers to a set of research questions.

The practitioner of literate programming can be regarded as an essayist, whose main concern is with exposition and excellence of style. Such an author, with thesaurus in hand, chooses the names of variables carefully and explains what each variable means. He or she strives 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 reinforce each other. (Knuth, 1984)

1 <https://github.com/TheUpshot>

2 <http://jupyter.org/>, <https://rmarkdown.rstudio.com/>

A good example of the form is found in *Buzzfeed News*' Jupyter Notebook detailing how they analyzed trends in California's wildfires.³ Whilst the notebook contains all the code and links to source data required to reproduce the analysis, the thrust of the document is a narrative or conversation with the source data. Explanations are set out under headings that follow a logical line of enquiry. Visualizations and charts are used to bring out key themes.

One aspect of the "literate" approach to programming is that the documents produced (as Jupyter Notebook or R Markdown files) may be capable of reassuring even those readers who cannot read the code itself that the steps taken to produce the conclusions are sound. The idea is similar to Steven Shapin and Simon Schaffer's account of "virtual witnessing" as a means of establishing matters of fact in early modern science. Using Robert Boyle's experimental program as an example, Shapin and Schaffer set out the role that "virtual witnessing" had:

The technology of virtual witnessing involves the production in a reader's mind of such an image of an experimental scene as obviates the necessity for either direct witness or replication. Through virtual witnessing the multiplication of witnesses could be, in principle, unlimited. It was therefore the most powerful technology for constituting matters of fact. The validation of experiments, and the crediting of their outcomes as matters of fact, necessarily entailed their realization in the laboratory of the mind and the mind's eye. What was required was a technology of trust and assurance that the things had been done and done in the way claimed. (Shapin & Schaffer, 1985)

Documents produced by literate programming environments such as Jupyter Notebooks—when published alongside articles—may have a similar effect in that they enable the non-programming reader to visualize the steps taken to produce the findings in a particular story. While the non-programming reader may not be able to understand or run the code itself, comments and explanations in the document may be capable of reassuring them that appropriate steps were taken to mitigate error.

Take, for instance, a recent *Buzzfeed News* story on children's home inspections in the United Kingdom.⁴ The Jupyter Notebook has specific steps to check that data has been correctly filtered (Figure 19.1), providing a backstop against the types of simple but serious mistakes that caught Reinhart and

3 <https://buzzfeednews.github.io/2018-07-wildfire-trends/>

4 <https://www.buzzfeed.com/richholmes/care-price>

```
In [11]: # Make sure that we've identified all private-sector owners
assert (
    as_at_data_filtered
    .loc[lambda df: df["Sector"] == "Private"]
    ["Owner"].isnull()
    .sum()
) == 0
```

Figure 19.1. A cell from the *Buzzfeed* Jupyter notebook with a human readable explanation or comment explaining that its purpose is to check that the filtering of the raw data was performed correctly. Source: Jeremy Singer-Vine, *Buzzfeed*.

Rogoff out.⁵ While the exact content of the code may not be comprehensible to the non-technical reader, the presence of these tests and backstops against error with appropriately plain English explanations may go some way to showing that the steps taken to produce the journalist's findings were sound.

More Than Just Reproducibility

Using literate programming environments for data stories does not just help make them more reproducible.

Publishing code can aid collaboration between organizations. In 2016, Global Witness published a web scraper that extracted details on companies and their shareholders from the Papua New Guinea company register.⁶ The initial piece of research aimed to identify the key beneficiaries of the corruption-prone trade in tropical timber, which is having a devastating impact on local communities. While Global Witness had no immediate plans to reuse the scraper it developed, the underlying code was published on GitHub—the popular code-sharing website.

Not long after, a community advocacy organization, ACT NOW!, downloaded the code from the scraper, improved it and incorporated it into their iPNG project that lets members of the public cross-check names of company shareholders and directors against other public interest sources.⁷ The scraper is now part of the core data infrastructure of the site, retrieving data from the Papua New Guinea company registry twice a year.

Writing code within a literate programming environment can also help to streamline certain internal processes where others within an organization

5 <https://github.com/BuzzFeedNews/2018-07-ofsted-inspections/blob/master/notebooks/oo-analyze-ofsted-data.ipynb>

6 <https://github.com/Global-Witness/papua-new-guinea-ipa>

7 <https://pngiportal.org/>

need to understand and check an analysis prior to publication. At Global Witness, Jupyter Notebooks have been used to streamline the legal review process. As notebooks set out the steps taken to get a certain finding in a logical order, lawyers can then make a more accurate assessment of the legal risks associated with a particular allegation.

In the context of investigative journalism, one area where this can be particularly important is where assumptions are made around the identity of specific individuals referenced in a data set. As part of our recent work on the state of corporate transparency in the United Kingdom, we wanted to establish which individuals controlled a very large number of companies. This is indicative (although not proof) of them being a so-called “nominee” which in certain contexts—such as when the individual is listed as a Person of Significant Control (PSC)—is illegal. When publishing the list of names of those individuals who controlled the most companies, the legal team wanted to know how we knew a specific individual, let’s say John Barry Smith, was the same as another individual named John B. Smith.⁸ A Jupyter Notebook was able to clearly capture how we had performed this type of deduplication by presenting a table at the relevant step that set out the fields that were used to assert the identity of individuals.⁹ These same processes have been used at Global Witness for fact-checking purposes as well.

Jupyter Notebooks have also proven particularly useful at Global Witness when there is need to monitor a specific data set over time. For instance, in 2018 Global Witness wanted to establish how the corruption risk in the London property market had changed over a two-year period.¹⁰ We acquired a new snapshot from the land registry of properties owned by foreign companies and reused and published a notebook we had developed for the same purpose two years previously.¹¹ This yielded comparable results with minimal overheads. The notebook has an additional advantage in this context, too: It allowed Global Witness to show its methodology in the absence of being able to republish the underlying source data which, at the time of analysis, had certain licensing restrictions. This is something very difficult to do in a spreadsheet-based workflow. Of course, the most effective

8 <https://www.globalwitness.org/en/campaigns/corruption-and-money-laundering/anonymous-company-owners/companies-we-keep/>

9 https://github.com/Global-Witness/the-companies-we-keep/blob/master/companies_we_keep_analysis.ipynb

10 <https://www.globalwitness.org/en/blog/two-years-still-dark-about-86000-anonymously-owned-uk-homes/>

11 https://github.com/Global-Witness/overseas-companies-land-ownership/blob/master/overseas_companies_land_ownership_analysis.ipynb

way of accounting for your method will always be to publish the raw data used. However, journalists often use data that cannot be republished for reasons of copyright, privacy or source protection.

While literate programming environments can clearly enhance the accountability and reproducibility of a journalist's data work, alongside other benefits, there are some important limitations.

One such limitation is that to reproduce (rather than just follow or “virtually witness”) an approach set out in a Jupyter Notebook or R Markdown document you need to know how to write, or at least run, code. The relatively nascent state of data journalism means that there is still a fairly small group of journalists, let alone general consumers of journalism, who can code. This means that it is unlikely that the GitHub repositories of newspapers will receive the same level of scrutiny as, say, peer-reviewed code referenced in an academic journal where larger portions of the community can actually interrogate the code itself. Data journalism may, therefore, be more prone to hidden errors in code itself when compared to research with a more technically literate audience. As Jeff Harris (2013) points out, it might not be long before we see programming corrections published alongside traditional reporting corrections. It is worth noting in this context that tools like Workbench (which is also mentioned in Stray's chapter in this book) are starting to be developed for journalists, which promise to deliver some of the functionality of literate programming environments without the need to write or understand any code.¹²

At this point it is also worth considering whether the new mechanisms for accountability in journalism may not just be new means through which a pre-existing “public” can scrutinize methods, but indeed play a role in the formation of new types of “publics.” This is a point made by Andrew Barry in his essay “Transparency as a Political Device”:

Transparency implies not just the publication of specific information; it also implies the formation of a society that is in a position to recognize and assess the value of—and if necessary to modify—the information that is made public. The operation of transparency is addressed to local witnesses, yet these witnesses are expected to be properly assembled, and their presence validated. There is thus a circular relation between the constitution of political assemblies and accounts of the oil economy—one brings the other into being. Transparency is not just intended to make information public, but to form a public which is interested in being informed. (Barry, 2010)

12 <http://workbenchdata.com/>

The methods elaborated on above for accounting for data journalistic work in themselves may play a role in the emergence of new groups of more technically aware publics that wish to scrutinize and hold reporters to account in ways not previously possible before the advent and use of technologies like literate programming environments.

This idea speaks to some of Global Witness' work on data literacy in order to enhance the accountability of the extractives sector. Landmark legislation in the European Union that forces extractives companies to publish project-level payments to governments for oil, gas and mining projects, an area highly vulnerable to corruption, has opened the possibility for far greater scrutiny of where these revenues actually accumulate. However, Global Witness and other advocacy groups within the Publish What You Pay coalition have long observed that there is no pre-existing "public" which could immediately play this role. As a result, Global Witness and others have developed resources and training programmes to assemble journalists and civil society groups in resource-rich countries who can be supported in developing the skills to use this data to more readily hold companies to account. One component of this effort has been the development and publication of specific methodologies for red-flagging suspicious payment reports that could be corrupt.¹³

Literate programming environments are currently a promising means through which data journalists are making their methodologies more transparent and accountable. While data will always remain open to multiple interpretations, technologies that make a reporter's assumptions explicit and their methods reproducible are valuable. They aid collaboration and open up an increasingly technical discipline to scrutiny from various publics. Given the current crisis of trust in journalism, a wider embrace of reproducible approaches may be one important way in which data teams can maintain their credibility.

Works Cited

- Barry, A. (2010). Transparency as a political device. In M. Akrich, Y. Barthe, F. Muniesa, & P. Mustar (Eds.), *Débordements: Mélanges offerts à Michel Callon* (pp. 21–39). Presses des Mines. <http://books.openedition.org/pressesmines/721>
- Harris, J. (2013, September 19). *The Times* regrets the programmer error. *Source*. <https://source.opennews.org/articles/times-regrets-programmer-error/>

13 <https://www.globalwitness.org/en/campaigns/oil-gas-and-mining/finding-missing-millions/>

- Harris, J. (2014, May 22). Distrust your data. *Source*. <https://source.opennews.org/articles/distrust-your-data/>
- Knuth, D. E. (1984). Literate programming. *The Computer Journal*, 27(2), pp. 97–111. <https://doi.org/10.1093/comjnl/27.2.97>
- Knuth, D. E. (1992). *Literate programming*. Center for the Study of Language and Information.
- Leather, B. (2017, July 13). Environmental defenders: Who are they and how do we decide if they have died in defence of their environment? *The Guardian*. <https://www.theguardian.com/environment/2017/jul/13/environmental-defenders-who-are-they-and-how-do-we-decide-if-they-have-died-in-defence-of-their-environment>
- Leather, B., & Kyte, B. (2017, July 13). Defenders: Methodology. Global Witness. <https://www.globalwitness.org/en/campaigns/environmental-activists/defendersmethodology/>
- Reinhart, C. M., & Rogoff, K. S. (2010). *Growth in a time of debt* (Working Paper No. 15639). National Bureau of Economic Research. <https://doi.org/10.3386/w15639>
- Shapin, S., & Schaffer, S. (1985). *Leviathan and the air-pump: Hobbes, Boyle, and the experimental life*. Princeton University Press.

About the Author

Sam Leon is Data Investigations Lead at Global Witness, where he researches and campaigns against corruption and environmental destruction.