Towards Editorial Transparency in Computational Journalism

Jennifer A Stark
Philip Merrill College of Journalism
University of Maryland
College Park, MD
starkja@umd.edu

Nicholas Diakopoulos
Philip Merrill College of Journalism
University of Maryland
College Park, MD
nad@umd.edu

ABSTRACT
The increasing frequency of use of data and code in journalistic projects drives the need to develop guidelines or frameworks for how to responsibly and accountably employ algorithms and data in acts of journalism. One route to the accountable use of algorithms in journalistic work is to develop standards and expectations for transparency. In this paper we describe steps toward transparency with respect to computational journalism drawing from two case studies. The first case study concerns algorithmic accountability reporting where data collected via the Uber API and government sources were analyzed to understand quality of Uber service across Washington D.C. The second case centers on editorial transparency in the creation of a tool – in this case, a Twitter bot – built as an exploration of automated surfacing of anecdotal comments from news articles. Based on our experiences in these two cases we describe approaches to sharing data and code. The benefits of transparency as well as considerations such as licensing and documentation are discussed.

Keywords
Computational journalism, algorithmic transparency, algorithmic accountability, bots, open-source.

1. INTRODUCTION
Transparency has been defined as "the ways in which people both inside and external to journalism are given a chance to monitor, check, criticize and even intervene in the journalistic process." [3]. A related notion of *algorithmic* transparency is oriented towards disclosing information about the ways in which data and algorithms are used in the news making process [6]. With the rise of data and algorithms in journalism, such as in news curation, data journalism, automated writing, social media analytics, and news recommender systems, it stands to reason that journalists also need to make their process, tools and decision-making more transparent. The purpose of this paper pertains to the practice of transparency as it relates to different computational journalism case studies that require distinct approaches, with the hope that describing these approaches may inform others and contribute to the development of a more formal practice.

Transparency is of particular importance in the era of computational and data journalism reporting, where much of the work may be obfuscated via analytic procedures, and data literacy among journalists and / or readers may hinder the valid interpretation of results. In the case of computational and data journalism, transparency might include sharing the steps taken to clean the data, the analysis code, the model or inferences made with that model, software used, as well as data and its provenance and quality [6]. If the work of journalists is transparent, it can serve to mitigate biases and allow readers to interpret the content more effectively if there are discrepancies in their understanding. Through transparency journalists are able to demonstrate that their story and the analytic evidence backing it is accurate to the best of their ability. Transparency reinforces the goal of media accountable by enabling outside stakeholders to criticize any errors made in reporting including analytic calculations, assumptions, or interpretations.

Fulfilling responsibilities often comes with a cost, but increasing transparency brings myriad benefits to journalists and to journalism more broadly. Making data, data analysis, and algorithms transparent encourages journalists to double check their process, knowing it may be scrutinized by an external audience. It encourages journalists to document and organize their code more thoroughly and effectively. All of this affords journalists the opportunity to ensure that what they are reporting is based on evidence, that that evidence is present and correct, that the steps to process the data contain no errors, and that everything is clear and follows logically.

In addition to enhancing the quality of journalistic work, transparency can also benefit the community of computational and data journalism as a whole. Advanced readers may provide constructive feedback for improvement - either in the code itself, or in the story and interpretation. Transparency can thus be useful for finding "bugs" and improve the accuracy of the work. Providing data and code enables others to extend the work, reproduce it in other jurisdictions, and consider alternative editorial points of view expressed in the technical instantiation of the work.

The goal of this paper is to demonstrate transparency in practice around two concrete computational journalism projects so as to demonstrate and provide guidelines to practitioners. While we are still hesitant to consider these to be 'best practices' we proffer them to the community in the hope they can guide others and be built upon. We will explore what considerations we made to the goals and benefits described above, and when they were considered appropriate or inappropriate for each project. The first case study focuses on algorithmic accountability reporting where data collection and analysis processes are explicitly shared, while the second relates to the building of a tool that can be edited to suit the specifications of different newsrooms.

2. CASE STUDIES
2.1 Algorithmic Accountability of Uber
2.1.1 Background
Algorithmic accountability reporting describes a branch of computational journalism where the algorithm is the object of investigation [4]. Although algorithms automate tasks and

Computation + Journalism Symposium, September 30-October 1, 2016, Palo Alto, California, USA.

decision making, they are created by humans who have their own biases and work for companies that have their own agendas. These factors make commercial or government algorithms worthy of journalistic scrutiny. In such investigations of algorithms, the data generated for a story can be considerable, consisting of data directly related to the algorithm as well as supporting data collected from other sources. In addition to the data are the methods employed by computational journalists to process, analyze, model, and visualize the information, which also requires clear and concise explanation.

2.1.2 Goals

Uber is a transport network company that connects non-commercial drivers with paying passengers via a mobile app. When passenger demand outnumbers driver supply in a given geographic area, a surge price multiplier is enacted where the cost of a ride will increase. From economic theory we might expect this price signal to act to both reduce the number of riders requesting a car (reduce demand), and increase the cars in the area (increase supply), ultimately leading to a shorter wait time for a car – or improved service. Uber uses an algorithm that determines surge pricing given the current supply and demand for any given area and has been explored by other researchers as well [2]. The goal of our investigation was to evaluate service quality provided by uberX across Washington D.C. census tracts with the hypothesis that there may be differences in service quality related to neighborhood and thus demographics. The story was published by The Washington Post, and concluded that, although a statistical model demonstrates that census tracts with higher percentages of people of color wait significantly longer for a car, we are not currently able to attribute cause [12].

2.1.3 Transparency Approach

The main mechanism of transparency employed was to make the data collection and analysis code we used freely available to anyone. That means that everything we used - as much as possible - was open-source and platform independent. All code is in Python, and all data is accessible either via python APIs including the Uber API and some data wrangling tools. We chose to host code and data on Github as it is free for open-source projects and widely used within the journalism community, including by outlets such as Buzzfeed[1], ProPublica[2], and The New York Times. Visitors can download a project, and submit issues or comments (e.g. if there is a bug or critique). If a README file is provided, this will be rendered in HTML below the code file directory. READMEs are convenient for providing project descriptions, instructions, and code examples to make the project more accessible to others.

2.1.3.1 Code & Data

A series of python scripts were used in collecting and analyzing data. First, geographical census data were collected using a python API (‘Get_geographic_data.ipynb’), so that we could later match the locations passed to the Uber API to D.C. census tracts along with their population counts and land areas. Second, ‘Mapping_points_across_DC.ipynb’ created a grid of points across D.C., which were then filtered to only include points at a valid address, and checked to ensure that all D.C. tracts were represented. The longitude-latitude pairs for each point, together with an ID for each pair, were converted to a JSON string. Third, the JSON string was passed to the ‘gatherUberData.py’ script which collected Uber surge prices and expected wait times from each location across D.C. every 3 minutes via Uber’s API. Finally, ‘UberSurgePricing_OSC.ipynb’ steps though the preparation of the data, including the gathering and preparation of demographic census data, exploration of relationships between variables, and the linear regression model.

Several layers of this project may be subject to transparency: The raw data; the creation of locations across D.C. to pass to the Uber API; and the data wrangling, analysis, and modeling. Since our goal is to be accountable to our readers, we chose to enable others to scrutinize how we manipulated and analyzed the data. This meant that the code presented should be clear, concise, and well-commented so that a user unfamiliar with python or coding in general could follow the steps taken. We used Jupyter Notebook[3] for this code, as it renders markdown as HTML, charts, and code, meaning that one can incorporate charts and text with the code, including graphics like the regression line charts published in The Washington Post article.

Making the analysis script available without data, however is rather limiting, as one cannot test alternative approaches without the data. Sharing the raw data on GitHub, however, was not possible due to the Github file size limit of 100MB. Instead, we compressed the comma delimited raw data, and put it in a Google Drive to be accessible via a link. This link was provided within the GitHub README and also within the article published in The Washington Post.

The rationale for sharing the scripts to create the locations that get passed to the Uber API was to provide a means by which others may produce the work in another town or city, using code where many challenges have already been addressed. The benefit to the original storytellers is that if an investigation is carried out in a different city and tells the same story, it lends support to the original story. If, however, it tells a different story, because the methods were the same, the interpretation may be that there is something specifically different between the two cities that is worth exploring further.

Providing all scripts allows others to run the whole project themselves from the beginning. However, most users may only be interested in specific parts of a project. For this reason, we also provided Pandas[4] dataframes as pickled files, and comma delimited .csv file outputs from various points along the project, so that users can access the project from several intermediate points. By increasing the granularity of intermediate results and making them transparent we hoped to improve the usability of the transparency information disclosed. Another benefit of this approach is that it makes internal collaboration on analyses easier. Pickling is a python method for data serialization. Unfortunately, while python and its libraries may be open-source, pickling data precluded access to the data by people using other data processing or statistical software. While pickling has many advantages and disadvantages that are beyond the scope of this paper, a comma-delimited file fulfills most requirements. Therefore, future projects will likely only save progressive dataframes as csv files.

Where possible steps were carried out programmatically as this improves replicability by reducing errors and variability between studies. This is not always possible. One example is where data is not generated from code or collected from APIs. In these cases it

[1] https://github.com/BuzzFeedNews

[2] https://github.com/propublica

[3] https://jupyter.org

[4] http://pandas.pydata.org

is helpful to indicate how data was obtained. In this case study, supporting data included census data accessible from various government agency web sites as comma delimited files. Therefore, we included a link to the website in the code notebook at the point that the code accesses the data, as well as the file obtained providing access to the data used, the source of that data, and an opportunity for others to test alternative data or data subsets.

A second example is where tools need to be integrated into the analytic or visualization stack that only have a graphical user interface, rather than a code interface, making documentation of how a specific output was achieved difficult. Charts of the linear model were created with code and included in the Jupyter Notebook, but the maps created for the Washington Post article were created in Carto[5] (formerly known as CartoDB). This software is free up to three maps, after which subscription is required. Steps to create the maps were not included with the study documentation as the maps were purely for illustration and did not reflect the linear model directly (they mapped actual wait time seconds, for example, rather than standardized and centered values). Excluding them does not negatively impact the ability for someone to recreate or edit the project. Attribution to Carto was provided on the maps in the article as per Carto's ToU.

2.2 Anecbotal Comment Bot

2.2.1 Background

Whereas algorithmic accountability reporting makes computation the object of study for investigative journalists, another equally important area of computational journalism activity relates directly to the creation and use of computational tools or platforms in the newsroom to assist journalists in activities such as data collection, moderation, alerting, storytelling, dissemination, personalization, and other news-related tasks [5, 11].

2.2.2 Goals

Our second case study stems from our work exploring the automation of news dissemination via social media bots [8]. We developed a twitter bot to identify news article comments that exhibited aspects of being anecdotal in nature, and tweet the comment as an image. The intention was that the bot could then be used as a reference tool for other news organizations that want to create their own news bots.

2.2.3 Twitterbot Algorithm

First, the bot listens to the twitter stream via the twitter API for predefined URLs (e.g. URLs containing 'npr.org/' or 'nytimes.com'). Second, it opens the link in the tweet to the news article and loads the parent-level comments (ignoring replies to comments) using an API to return the comments. Third, it will process the text in the comments for comment length, reading level, and degree of personal experience using a dictionary of self-reference words, or words relating to family and friends [10]. This part constitutes the "anecdotal" aspect of the bot given that previous work has shown the score is useful for surfacing comments that describe personal experiences. The three scores are then weighted in a linear combination to calculate an overall "anecdotal" score for each of the comments. Finally, one comment is picked at random from the top three scoring comments, and tweeted as a .png file.

[5] https://carto.com

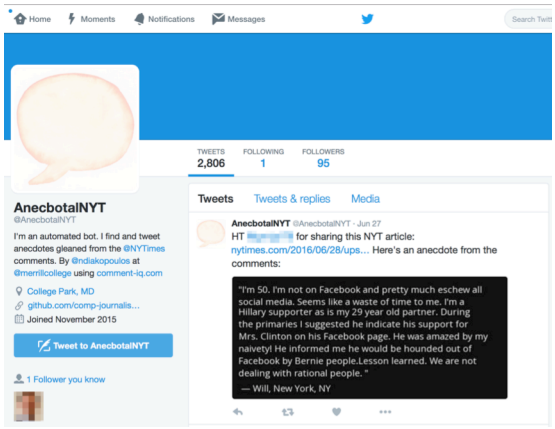

Figure 1. Screenshot of @AnecbotalNYT with example of a comment from The New York Times tweeted as an image.

Twitter, news comment APIs, and independent comment system APIs such as Disqus, have daily or hourly rate limits. So as not to reach these limits and also to not be reported or flagged as spam on twitter, there are several filtering steps that act to prevent over-tweeting, and that ensure high quality of comments tweeted by the bot. Once the bot identifies a tweet with a specific URL, the bot assesses whether the associated article has a number of comments over a user-defined threshold (e.g. more than 100 comments). This ensures that there is robust conversation or that the article is popular. At the tenth tweet containing a link to an article with 100+ comments, the bot collects all parent comments over a certain length. If the comment system is Disqus only comments from authors with a user-defined reputation level are considered. The comment texts are then passed to the text-scoring algorithm, sorted in descending score order, and one comment chosen at random from the top three comments. If a rate limit is reached, the bot pauses activity for 24 hours.

2.2.4 Transparency Approach

As with the Uber project, the code is available on GitHub. The README describes the background of the bot project, its goal, usage (installation, set-up, testing), links to the example config files, links to our twitter bot accounts, and links to twitter, Disqus and NYT API instructions, license information and contact details. Importantly, the README also includes accessibly written editorial transparency about where the data driving the bot comes from, how it's sampled, how comments are scored and ranked by the bot, and other editorial rules. The motivation for these descriptions was the set of algorithmic transparency information described in [6] across layers of data, model, inference, and interface.

To facilitate informed editing of the code by new users, we could have relied on within-code comments explaining each line and how one might edit the code for a desired outcome. However, this method can be overwhelming and time consuming for a user new to coding, or unfamiliar with python, and with all the possible edits one could make, given free reign. Therefore, we parameterized a set of editorial rules that may be relevant to a newsroom in the code, and made just these elements editable in a configuration file. This simplifies the editing process, reduces the time required to make those changes and to iterate, and is programming-language agnostic. We separated the config sections into blocks to further structure the process. These blocks were for (1) Twitter (API keys, secret, tokens and the twitter bot handle), (2) comment system information, including API name (e.g.

Disqus or NYT), keys, filter object (the keyword to use to filter tweets), a list of news organization URLs to search for in the tweets containing the filter word, and the minimum number of comments required for the comments to undergo natural language processing. For Disqus comments, this section also includes setting an author reputation level and the name of the Disqus comment forum. (3) Text analytics (score weights for the length, reading level and anecdotal level), (4) Text configuration parameters (font, font size and color), and (5) and finally background design (background color and a filename of a logo, if desired). These last two sections allow news organizations to implement their signature style. Additionally, we provided example config files for both NYT and Disqus comment systems to support written instructions in the README.

One issue encountered when creating tools is how to balance flexibility with the original goal of the tool and simplicity of use. If each part of the code was parameterized, the configuration file would become unwieldy, and the tool may lose its identity. For example, with Anecbotal we did not parameterize the dictionary of words indicating "personal experiences". This dictionary is essentially what makes the bot anecdotal in that it enables the text analysis to calculate a score based on counts of words in that dictionary that are found in the article comments. It is not unreasonable to suppose that editors may want to use this bot to surface comments with other specific editorial intents such as certain political or sports content, which would require their own politics- or sports-specific dictionaries.

There are also challenges encountered where it would benefit tool users if an aspect of code was abstracted, but cannot be due to case-by-case uniqueness. With Anecbotal we found that some news article URLs contained a unique identifier to the Disqus thread, like a slug or query string. Passing this URL to the Disqus API would sometimes result in an error. When present, this identifier may be peculiar to the specific news organization. It is difficult to parameterize in a way that accounts for each and every variation, so instead we included a function in the main code for the bot where one can add code for their specific circumstance. The disadvantage is that this requires some coding knowledge, but we try to mitigate this limitation in the README where the issue is described along with the code section to edit.

It was important that we also made the text processing transparent, even if it was not parameterized aside from the score weights. Therefore we included a section in the README about the model used to calculate the scores, and links to supporting papers, articles and projects. Given this background information, more knowledgeable coders may make their own edits if required.

3. DISCUSSION

Journalism ethics encourages practitioners to adhere to certain tenets such as independence of politics or special interest groups, reporting true and accurate information, and increasingly to be held accountable by one's audience via transparency [9]. In this work we have described the concrete application of journalistic transparency to two diverse cases of computational journalism. Here we step back and reflect on our experiences more broadly, considering practical impacts as they may relate to other computational journalists considering transparency.

3.1.1 What to Open-source and Why?

Transparency facilitates reproducible and replicable research: basic principles in scientific research that are also relevant to computational journalism. *Reproducibility* requires the code and data are available so a user can rerun the original analysis on the original data. This is the most basic requirement for checking code and verifying results. This was achieved in the algorithmic accountability reporting project by sharing the raw data on Google Drive and the analysis code as a Jupyter Notebook on GitHub. *Replicability*, on the other hand, requires achieving the same outcome with *independent* data collection, code and analysis. If the same outcome can be achieved with a different sample, experimenters and analysis software, then it is more likely to be true. This was accomplished by sharing the collection code on GitHub. The code could be downloaded and edited as appropriate, together with the analysis code to conduct an independent experiment. In our case - and perhaps in data stories across journalism - *replicability per se* may not always apply due to unique social-economic, demographic, government or other relationships relevant to the story. For example, we found that wait times for an uberX in neighborhoods with more people of color are longer: if one carries out similar analysis on a sample of data collected in, say, Warwick, Rhode Island (90% white, 5% poverty, lacking public transport), getting a different result would not make our report in D.C. to be less true or impactful as a story. Instead, then, the value of sharing data collection and analysis code may be to enable others to perform similar research with different sample data as has been suggested by other computational journalism tools that have focused on a particular geography but theorized about extensibility to other jurisdictions [1]. Providing the opportunity for others to verify data, as well as reuse code for one's own stories is important. We believe this is a conversation that will remain ongoing as journalism methods, tools, and standards evolve.

Whereas transparency in the Uber project was intended to promote accountability and to facilitate reproducibility, replicability, or advancement of our work, transparency in the Abecbotal project is aimed more at the editorial process: the goal of transparency here is to facilitate editing the code in order to produce the desired output according to the specific goals or aesthetic values of the news organization. Building tools that can be applied in different newsrooms benefits the industry as a whole, rather than commercializing them per se. Technological development is hastened, and consumer-engagement is quickly assessed, measured, and the technology abandoned as a failure or grown rapidly. The potential cost is ongoing support. However, this can be mitigated to some extent with the appropriate license (see below).

3.1.2 Documentation

The difference between an open-source project being a success or a failure can rest on the quality of documentation. The importance of good documentation and time required to write it is not lost on Knight-Mozilla OpenNews, who organize code convenings specifically to provide dedicated time for journalists to write documentation for journalism projects[6]. The two case studies covered in this paper used different documentation approaches. In the algorithmic accountability project, the README provided a very general overview, including a link to The Washington Post article, the goal of the project, dependencies, the purpose of each code file, and a data dictionary for the raw data and link to that data on our Google Drive. Contextual documentation was provided in the code Notebooks. Buzzfeed has used similar contextual documentation where passages from the article, analysis description, and Notebook links were integrated into the

[6] https://opennews.org/what/community/convenings/

README[7]. In contrast, the vast majority of documentation for Anecbotal resided in the README, which described the purpose, references, dependencies, and instructions.

Both projects used software external to the project itself, including Carto, Twitter and Disqus. Documenting these external steps can be difficult, and might be unnecessary if they already have relevant instructions (e.g. for setting up a Twitter or Disqus application) that you can link to in the README.

3.1.3 Sharing, Storing, and Archiving

Projects can create lots of data, or access myriad data sources. Our case studies had unique data sharing, usage, and data creation expectations. The Uber story *collected* almost 3GB of raw data, *created* additional data via preprocessing and analysis, and *downloaded* additional data from government sources. The Anecbotal project accumulated none. Kasianovitz & Roberts [7] make recommendations for sharing, preserving and archiving data. Choice of storage and dissemination of a project may depend on the purpose of the project and how long it needs to be available, as some platforms may be less stable in terms of longevity than others. Considerations for third party storage or sharing platforms include how new is the platform (new data-sharing startup vs. established storage company), current or future data limits, and charges for storage space now or in the future. Another consideration is the relevance of the analysis tools or the APIs in the future. Advances in analysis or changes in the API may mean that rerunning the code as it is very unlikely. Kasianovitz & Roberts [7] point out that while GitHub and Google Drive– used for storing and sharing both case studies – are sufficient for access by other parties, they are not archives.. If long-term storage is deemed critical, institutional or library-based archiving may be viable options [7].

If, like the Uber project, there are data from third party sources like the government, it may be recommended to save, store and share a copy of the data rather than rely on a link. This will protect the project against website reorganization and dead links, or if the third party change their data policies in terms of their own archives. This will also ensure the data remains intact over the course of longitudinal research so if the third party updates or replace a database, your data will remain preserved.

Under circumstances of violation of privacy, or if the data is proprietary, sharing the original data may not be possible. In these cases, the data may be shared after some degree of aggregation of the data to prevent the ability to identify individual persons. Alternatively, if data cannot be shared, the data and the analysis code can still be documented.

3.1.4 Licensing

Licenses are essential if the data and code are to be open-sourced. Without a license, the work is only available to view – not to use - unless explicitly stated. Code and data require different licenses, descriptions for which can be found online for open data[8], and for open-source code[9]. If sharing data from the government, they will likely have a Terms and Conditions notice detailing the license and how to cite the data source. Data collected via an API may require separate licensing or permissions from the organization. Therefore it is important that we license the work with an open-source license with affordances and restrictions appropriate to both our goals and what it is that is being licensed. . We selected the MIT License for Anecbotal as it is most permissible, ensuring that others can use it and share it for free, and states the code is shared "as is" The Uber case study, however, is a mixed project consisting of both open code (the data collection and analysis) and open data (government census data and Uber API data). In this case, GitHub recommends using multiple licenses, and explicitly states what each of them is for (http://choosealicense.com/non-software/).

[7] https://github.com/BuzzFeedNews/2015-12-H-2-visas-and-experience-requirements

[8] http://opendatacommons.org/about/

[9] https://opensource.org/licenses

4. CLOSING

We hope these case studies and examples act to begin formalizing a best practice, with the expectation that standards or requirements will develop along with the field of computational journalism.

5. ACKNOWLEDGMENTS

Tow Center for Digital Journalism for funding this work, Knight-Mozilla OpenNews for funding and organizing the bot-themed Code Convening that helped advance open sourcing Anecbotal.

6. REFERENCES

[1] Broussard, M. 2015. Artificial Intelligence for Investigative Reporting: Using an expert system to enhance journalists' ability to discover original public affairs stories. *Digital Journalism*. 3, 6 (2015), 814–831.

[2] Chen, L. et al. 2015. Peeking Beneath the Hood of Uber. *Proc. of the 2015 ACM Conference on Internet Measurement Conference - IMC*. (2015), 495–508.

[3] Deuze, M. 2005. What is journalism?: Professional identity and ideology of journalists reconsidered. *Journalism*. 6, 4 (2005), 442–464.

[4] Diakopoulos, N. 2015. Algorithmic Accountability: Journalistic investigation of computational power structures. *Digital Journalism*. 3, 3 (2015), 398–415.

[5] Diakopoulos, N. 2016. Computational Journalism and the Emergence of News Platforms. *The Routledge Companion to Digital Journalism Studies. Eds. Scott Eldridge II and Bob Franklin*.

[6] Diakopoulos, N. and Koliska, M. 2016. Algorithmic Transparency in the News Media. *Digital Journalism*. forthcoming, (2016).

[7] Kasianovitz, K. and Roberts, R.L. 2015. No Data, No Computation, No Replication or Re-use: the Utility of Data Management and Preservation Practices for Computational Journalism. Proc. Computation + Journalism Symposium.

[8] Lokot, T. and Diakopoulos, N. 2016. News Bots. *Digital Journalism*. 4, 6, 682–699.

[9] McBride, K. and Rosenstiel, T. eds. 2014. *The New Ethics of Journalism: Principles for the 21st Century*. CQ Press.

[10] Park, D. et al. 2016. Supporting Comment Moderators in Identifying High Quality Online News Comments. *Proc. Conference on Human Factors in Computing Systems (CHI)*. (2016), 1114–1125.

[11] Shearer, Matt; Simon, Basile; Gieger, Clément, B.N.L. 2014. Datastringer: easy dataset monitoring for journalists. Proc. Computation + Journalism Symposium.

[12] Stark, J. and Diakopoulos, N. 2016. Uber seems to offer better service in areas with more white people. That raises some tough questions. *The Washington Post*.