What Makes a Well-Documented Notebook? A Case Study of Data Scientists' Documentation Practices in Kaggle

APRIL YI WANG, University of Michigan, USA
DAKUO WANG, IBM Research, USA
JAIMIE DROZDAL, Rensselaer Polytechnic Institute, USA
XUYE LIU, Rensselaer Polytechnic Institute, USA
SOYA PARK, MIT CSAIL, USA
STEVE ONEY, University of Michigan, USA
CHRISTOPHER BROOKS, University of Michigan, USA

Many data scientists use computational notebooks to test and present their work, as a notebook can weave code and documentation together (computational narrative), and support rapid iteration on code experiments. However, it is not easy to write good documentation in a data science notebook, partially because there is a lack of a corpus of well-documented notebooks as exemplars for data scientists to follow. To cope with this challenge, this work looks at Kaggle — a large online community for data scientists to host and participate in machine learning competitions — and considers highly-voted Kaggle notebooks as a proxy for well-documented notebooks. Through a qualitative analysis at both the notebook level and the markdown-cell level, we find these notebooks are indeed well documented in reference to previous literature. Our analysis also reveals nine categories of content that data scientists write in their documentation cells, and these documentation cells often interplay with different stages of the data science lifecycle. We conclude the paper with design implications and future research directions.

CCS Concepts: • **Human-centered computing** → **Interactive systems and tools**; **Empirical studies in HCI**; • **Computing methodologies** → *Natural language generation*; • **Software and its engineering** → *Documentation.*

Additional Key Words and Phrases: Computational notebooks, code documentation, data science, Kaggle, machine learning

ACM Reference Format:
April Yi Wang, Dakuo Wang, Jaimie Drozdal, Xuye Liu, Soya Park, Steve Oney, and Christopher Brooks. 2021. What Makes a Well-Documented Notebook? A Case Study of Data Scientists' Documentation Practices in Kaggle. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts (CHI '21 Extended Abstracts), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3411763.3451617

1 INTRODUCTION

Computational notebooks (e.g., Jupyter Notebooks or Google CoLabs) have become the predominant coding environment for data scientists and machine learning engineers [13, 32]. Computational notebooks allow users to write "computational narratives" that combine code, natural language documentation, and visual output in the same user interface. Compared to other coding environments (e.g., rigid script in a console), computational notebooks better serve the exploratory

Manuscript submitted to ACM

needs of the data science workflow [27]; a data scientist can rapidly iterate and explore alternative solutions to a problem and see the results. The combination of code and prose also allows data scientists to easily share and present their solutions to other data scientists or business partners.

Despite these benefits of having documentation and code together in a computational narrative, in reality, data scientists often neglect to write documentations in their notebooks [13, 25]. This raises a research question that is underexplored thusfar: *how can we build technologies to support data scientists' documentation practice?* Our ultimate research goal is to develop features to alleviate data scientists' burden of writing documentation, while still enabling rapid iteration and exploration. As a first step to fulfill this vision, we need to build an empirical understanding of what constitutes good documentation in computational notebooks.

Data science is highly interdisciplinary. It often involves various roles [32], spans across different stages of the data science lifecycle [30], and has a wide spectrum of usage scenarios [20]. Thus, different data scientists' documentation best practices may be different depending on the user persona and the user scenario. For example, a notebook written for one's future self may have a different definition of documentation best practices than the notebooks written for other audiences. This complex problem space imposes a challenge for researchers to find a representative notebook data corpus to analyze the documentation practices, which may partially explain why this topic is understudied.

To help fill this gap, we analyzed the notebooks submitted as solutions to Kaggle competitions. Kaggle is a large online data science community where users from diverse backgrounds can host or participate in data science competitions across different fields, submit their solutions as notebooks, and compete with other solutions in a public leaderboard. The public leaderboard ranks notebooks based on the performance metrics of the solution (i.e., accuracy). Community members can also vote on their favorite notebooks. However, the top-voted notebooks by the community are often not the top-ranked ones on the leaderboard. We hypothesize these highly-voted notebooks generally have high readability and better documentation, which is why they are highly regarded by the community members. These notebooks are generated from diverse user personas and for different user scenarios. Thus, we can use them to build a data corpus as a proxy to understand what constitutes good documentation for data scientists' notebooks.

In all, we sampled 80 highly-voted Kaggle notebooks and conducted a qualitative content analysis to explore their documentation practices. Our analysis is at both the notebook level and the cell level.

At the notebook level, we aim to answer the question:

RQ1 Are the top-voted Kaggle notebooks a good approximation of well-documented notebooks, in comparison to the generally available Github notebooks reported in a previous study [25]?

At the code cell level, we aim to further explore:

RQ2 What types of documentation do data scientists write in the top-voted notebooks?

RQ3 And, how do the documentation practices interplay with the different stages of the data science lifecycle [30]?

2 RELATED WORK

2.1 Challenges of Documentation in Computational Notebooks

Data scientists often use computational notebooks to combine a variety of media, including text explanations, graphs, forms, interactive visualizations, code segments, and their outputs, into computational narratives. The data science community has widely adopted computational notebooks under the premise that it can help data scientists to effectively create, communicate, and collaborate on their analysis work [22].

However, many data scientists find the documentation practice expensive and tedious, thus computational notebooks are often not appropriately documented. For example, Rule et al. examined 1 million open-source computational notebooks from Github and found that one in four lacked any sort of written documentation [25]. Among this data corpus, they further sampled 221 academic computational notebooks, which they considered are higher quality notebooks. Their analysis found that academic computational notebooks contained documentation cells for the purpose of introduction, describing analytical steps, explaining the reasoning, and discussing results. This known challenge of having incomplete or no documentation indeed hinders the readability and reusability of these notebooks, when they are shared with other collaborators or with one's future self [2].

Some researchers attribute the cause of this challenge to the iterative and explorative nature of data science projects; data scientists often need to explore multiple hypotheses and candidate coding solutions [14, 17, 23], many of which will be discarded later. Documenting those alternatives while coding would impose an unnecessary workload to data scientists, and could interfere with their thinking process of coding.

Another possible explanation of why data scientists do not write good quality documentation is that data science projects often involve interdisciplinary domain knowledge, multiple stakeholders, and many iterative stages of a data science lifecycle [7, 19, 30–32]. For example, Wang et al. [31] summarized a typical data science workflow as three high-level phases: data preparation, model building, and model deployment, where each phase contains many sub-goals and activities; Similarly, [30] proposes a taxonomy of ten stages and six user personas to describe the complex research space. It is possible that a good documentation practice for one user persona in a particular data science stage may be considered not so good for another user persona or in another stage.

We aim to build novel features to support data scientists' documentation practices while still maintaining the benefits of notebooks (rapid iteration and exploration). Thus, we need to better understand the best practices of data scientist documentation from a good corpus of data science notebooks. In addition to understanding what contents are documented in such notebooks, we also want to explore how the documentation interplay with data science lifecycle.

2.2 Shared Computational Notebooks in Github and Kaggle

Computational notebooks have the promise of easy sharing and communicating the story of data analysis with others. Källén et al. [12] found that cell-level cloning is common in Jupyter notebooks hosted on GitHub, where "type 1" clones (exact copy) are seen more often than "type 2" clones (where variables are renamed) and "type 3" clones (where a few statements were changed). Zhang et al. [33] found that academic notebooks on GitHub contain more codes related to data exploration and less on model development. However, these studies often focused on the code sharing and reusing practices, rather than documentation practices. As we describe above, Github notebooks often lack documentation (one in four has no documentation) [25], thus we need to look for a better data corpus to explore data scientists' documentation practices.

A number of recent studies started to explore another data science online community — Kaggle. Kaggle provides a platform where users or organizations can post datasets as machine learning challenges, and community members can submit their solutions in a notebook to those challenges. If a solution has the highest model accuracy, it wins the competition. In addition, community members can vote up or down on others' submitted notebooks. Often, the winning solutions are not the highly-voted ones, as community members voted on the readability and completeness of the computational narrative. Cheng and Zachry [3] interviewed why users participate in Kaggle competitions, and what challenges they have. Tauchert et al. [28] studied the motivation of organizers hosting Kaggle competitions. They

characterized the practice as harvesting crowd wisdom in online communities, as the solutions to a competition can inspire organizers with state-of-art approaches and connect organizers with promising competition participants.

In this work, we leverage the Kaggle online community to collect highly-voted notebooks as a corpus for analysis to understand the documentation practices in computational notebooks.

2.3 Code Documentation Practice in Software Engineering

Although documentation practice is underexplored in data science notebooks, we may leverage existing understandings of code documentaiton practices in software engineering. Programmers often write comments in their source code to make the code easily understood by both themselves and other developers [21]. Writing clear and comprehensive documentation is critical to software projects' development and maintenance [4, 11, 18, 24, 26]. However, writing documentation is also time-consuming. Thus, many projects (especially open source projects) have low quality documentation, partially due to the low level of intrinsic enjoyment for doing documentation [6].

The documentation practice in software engineering projects is different from the ones in data science projects. Documentation in software engineering primarily describes what the code does, thus this documentation follows an established pattern. For these straightforward and patterned documenation tasks, a template-based approach is sufficient to support users to generate documentations For example, tools like JavaDoc[8] allow programmers to annotate code with tags (e.g., `@param`, `@return`) and automatically generate documentation using these tags.

Our work focuses more on the underexplored research topic about data science documentation. Documentation in data science is less structured and more extensive. It may describe how a data set is constructed, explain the motivation behind an analysis, and interpret the model results and visualization. It can be much more complicated than the documentation in software engineering projects.

3 METHOD

To understand data science documentation best practices in computational notebooks, we conducted a qualitative analysis on a set of 80 highly-voted Kaggle notebooks. The notebooks that we analyzed were Jupyter notebooks [15], which consists of individual "cells". Every cell in a notebook contains either code or documentation written in Markdown (a language for creating formatted text).

3.1 Data Collection

We collected notebooks from two popular Kaggle competitions — a) House Price Prediction [9] and b) Titanic Survival Prediction [10]. We chose these two competitions because they are the most popular competitions (5,280 notebooks submitted for House Price and 6,300 notebooks submitted for Titanic Survival), and because many data science courses use these two competitions as tutorials [1, 5]. We collected the top 1% of the submitted notebooks from each competition based on their voting numbers, which resulted in 53 for House Price and 63 for Titanic Survival. We then cleaned the dataset by filtering out the notebooks that were not written in English and the ones that are not relevant to the particular challenge (e.g., a computational notebook introduces a new Python library, but it is irrelavent to the particular data science challenge), which returned 80 valid notebooks for analysis (39 for House Price and 41 for Titanic Survival).

3.2 Data Analysis

We began the analysis by describing how these 80 notebooks look like (**RQ1**), and the general characteristics of these documentation practices. Then, five members of the research team conducted an iterative open coding practice to

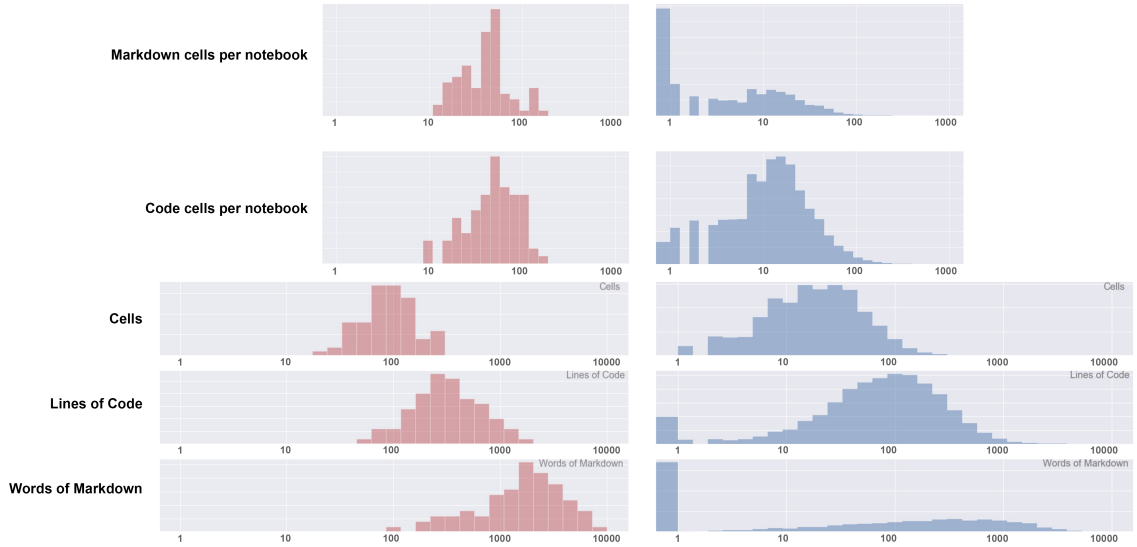

Fig. 1. We replicated the notebook level descriptive analysis by Rule et al. [25] to the 80 highly-voted notebooks on Kaggle. The left side (in red) represents the descriptive visualization of the 80 highly-voted computational notebooks from Kaggle (noted as Kaggle Corpus) and the right side (in blue) represents the descriptive visualization of the 1 million computational notebooks on Github (noted as Github Corpus). The Kaggle Corpus is better documented compared to the Github Corpus.

analyze the cell-level documentation practices. We categorized each documentation cell's purpose and content type **(RQ2).** We also coded each cell with a data science lifecycle stage to which the cell belongs **(RQ3)** (e.g., data cleaning or modeling training [30]). We achieved pair-wise inter-rater reliability ranged 0.78–0.95 (Cohen's κ) In total, our analysis covered 4,427 code cells and 3,606 Markdown (documentation) cells in the 80 notebooks.

4 RESULTS

Our analysis shows that these 80 highly-voted computational notebooks are indeed well-documented notebooks, in comparison to the Github data corpus (RQ1). Also, we extracted **nine different categories for the content** to describe the documentation practices at the cell level (RQ2). In addition, we found these Kaggle notebooks' documentation only covered **four** of the ten stages of the data science lifecycle (RQ3).

4.1 Highly-Voted Kaggle Notebooks are Well-Documented Notebooks.

We found that on average, each notebook we analyzed contains 55.3 code cells and 45.1 Markdown cells. We replicated the notebook descriptive analysis that Rule et al. used to analyze 1 million computational notebooks on Github [25]. As shown in Figure 1, the left side represents the descriptive visualization of the 80 highly-voted computational notebooks from Kaggle (noted as Kaggle Corpus) and the right side represents the descriptive visualization of the 1 million computational notebooks on Github (noted as Github Corpus). We found that the Kaggle Corpus has more total cells per notebook (Median = 95) than Github Corpus (Median = 18). The kaggle Corpus has a roughly equal ratio of Markdown cells and code cells per notebook, while Github Corpus is unbalanced with the majority of cells being code cells. Notably, Kaggle Corpus has more total words in Markdown cells (Median = 1728) than Github Corpus (Median = 218). This result indicates that the 80 highly-voted computational notebooks are better documented than general Github notebooks.

Table 1. We identified 9 categories based on the purpose of Markdown cells. Note that a Markdown cell may belong to multiple categories of contents or none of the categories.

Category	N	Description	Example
Process	2115 (58.65%)	The Markdown cell describes what the following code cell is doing. This always appears before the relevant code cell.	`Transforming Feature X to a new binary variable`
Headline	1167 (32.36%)	The Markdown cell contains a headline in Markdown syntax. The cell is used for navigation purposes or marking the structure of the notebook. It may be relevant to a nearby code cell.	`# Blending Models`
Result	692 (19.19%)	The Markdown cell explains the output. This type always appears after the relevant code cell.	`It turns out there is a long tail of outlying properties...`
Education	414 (11.48%)	The Markdown cell provides a rich content as an educational tutorial, but may not be relevant to a specific code cell.	`Multicollinearity increases the standard errors of the coefficients.`
Reason	227 (6.30%)	The Markdown cell explains the reasons why certain functions are used or why a task is performed. This may appear before or after the relevant code cell.	`We do this manually, because ML models won't be able to reliably tell the differences.`

Todo	202 (5.60%)	The Markdown cell describes a list of actions for future implementations. This normally is not relevant to a specific code cell.	`1. Apply models`
			`2. Get cross validation scores`
			`3. Calculate the mean`

Reference	200 (5.55%)	The Markdown cell contains an external reference. This is also relevant to the adjacent code cell.	`Gradient Boosting Regression` `Refer [here](https://...)`
Meta-Information	141 (3.91%)	The Markdown cell contains meta-information such as project overview, author's information, and a link to the data sources. This often is not relevant to a specific code.	`The purpose of this notebook is to build a model with Tensorflow.`
Summary	51 (1.41%)	The Markdown cell summarizes what has been done so far for a section or a series of steps. This often is not relevant to a specific code.	`**In summary**` `By EDA we found a strong impact of features like Age, Embarked..`

4.2 Data Science Documentation Covers a Broad Range of Topics and Purposes.

As Table 1 shows, we coded nine categories of documentation in the Markdown cells. Our analysis revealed that Markdown cells are mostly used to describe what the adjacent code cell is doing (Process, 58.65%). Second to the Process category, 32.36% Markdown cells are used to specify a headline for organizing the notebook into separate functional sections and for navigation purposes (Headline).

Markdown cells can also be used to explain beyond the adjacent code cells. We found that many Markdown cells are created to describe the outputs from code execution (Result, 19.19%), to explain results or critical decisions (Reason, 6.30%), or to provide an outline for the readers to know what they are going to do in a list of todo actions (Todo, 5.60%), and/or to recap what has been done so far (Summary, 1.41%).

We observed that 11.48% Markdown cells explain what a general data science concept means, or how a function works (Education), while 5.54% Markdown cells are connected with external references for readers to further explore

Table 2. We coded each Markdown cell to which data science stage (or task) they belong. We identified 4 stages with 13 tasks. Note that a Markdown cell may belong to multiple stages or none of the stages.

Stage	Total	Task	N
Environment Configuration	162 (4.49%)	Library Loading	33 (0.92%)
		Data Loading	129 (3.58%)
Data Preparation and Exploration	1336 (37.05%)	Data Preparation	91 (2.52%)
		Exploratory Data Analysis	960 (26.62%)
		Data Cleaning	285 (7.90%)
Feature Engineering and Selection	375 (10.40%)	Feature Engineering	120 (3.32%)
		Feature Transformation	178 (4.94%)
		Feature Selection	77 (2.14%)
Model Building and Selection	994 (27.57%)	Model Building	247 (6.85%)
		Data Sub-Sampling and Train-Test Splitting	61 (1.69%)
		Model Training	377 (10.45%)
		Model Parameter Tuning	81 (2.25%)
		Model Validation and Assembling	288 (6.32%)

the topics (Reference). We believe these are the extra efforts that the notebook owners dedicated, to attract a broader audience, especially beginners in the Kaggle community. In addition, some authors want to leave their own signature, and so they spend spaces at the beginning of the notebooks to debrief the project, to add the author's information, or even to add their mottos (Meta-Information, 3.91%).

4.3 Notebook Documentations Interplay with the Stages of the Data Science Lifecycle.

We coded Markdown cells based on where they belong in the data science workflow [30]. We found that the data science problems on Kaggle competitions contain clearly project goals and datasets, where data scientists do not need to go through processes like transforming business goals to data science goals and collecting data. Notebooks on Kaggle also do not reflect the model deployment stage, where in practice, data scientists need to deploy models, monitor performances, and further improve their models. In total, we identified four stages and 13 tasks. The four stages include **environment configuration** (4.50%), **data preparation** and exploration (37.05%), **feature engineering and selection** (10.40%), and **model building and selection** (27.57%). At the finer-grained task level, in particular, notebook authors create more Markdown cells for documenting exploratory data analysis tasks (26.62%) and model training tasks (10.45%). The rest of the Markdown cells are evenly distributed along with other tasks.

5 DISCUSSION

5.1 Best Practices for Data Science Documentation and Design Implications

Our findings suggest that the highly-voted computational notebooks on Kaggle present well-documentation practices among data science professionals. Novice data scientists can learn not only data science programming and problem-solving skills, but also the documentation practices from online data science communities. Instructors can benefit from these documentation practices so that they can better educate and prepare future data scientists. In addition, tool designers can get design inspirations to improve the current data science programming tools to better support documentation writing. Below, we discuss several design opportunities.

Our analysis identified nine categories of documentation in computational notebooks, and we suggest building automation techniques (AI) to support human data scientists' documentation tasks, similar to prior work on leveraging AI to support the model building tasks [29]. For different types of documentaitons, we can use design different generation approahces. For example, the Process category, which simply describes what the code does, may be inferred by parsing the API documentation. Thus, the notebook can automatically complete documentation, as in software engineering coding systems [8, 16]. But for some other categories, such as Education and Reason, there needs a more human-centered approach. Maybe a simple prompt to nudge users is more effective for helping users to create this documentation.

The timing of suggesting documentation is also an important consideration. Previous studies [13, 25] suggest that data scientists are hesitant to follow the practice of literate programming to create documentation during the development of code, despite they acknowledge the benefits that these informal notes have for recalling the analysis path. What if we can use some automation or prompt to help data scientists generate documentation during the exploration process? Alternatively, some data scientists may prefer to focus only on the thinking of code, and leave the documentation work to the end after they finish all the coding in the notebook. Then, the technology can be designed to scan through the notebook code cells, and automatically fill in the documentations in appropriate positions. This is promising as there are many breakthroughs in NLP and ML for automatically generating natural language descriptions for code snippets [16].

5.2 Limitation and Future Work

In this work, we used highly-voted Kaggle notebooks as an approximation to the well-documented notebooks and explored their documentation practices. Our result indeed shows that our corpus is generally well documented, and it yields insightful understandings of data scientists' documentation practices. However, we acknowledge that our approximation may be inaccurate, and there may be other data corpora that are better suited for understanding the documentation practices of data scientists. For example, the code in our Kaggle notebook corpus stopped at the Model Building stage of the lifecycle, yet we know in a real data science project, there are many other stages (model deployment and model runtime monitoring), but due to the nature of Kaggle competition setup, this corpus can not reveal documentation practices of those later stages. We welcome other researchers to join our effort to find and share their data corpora for the whole research community. [1] In addition, it is worth examining the general data science documentation practice on Kaggle by looking at both high-voted notebooks and low popularity notebooks.

Future works can leverage our data corpus to further explore other data science best practices, given our work only looked at the documentation behavior. Another future direction is to actually implement documentation-support features to support the variety of documentation types in a notebook. And once such features are implemented, a user study is required to further evaluate the benefits and tradeoffs of how the different design decisions may influence users' coding or documentation behaviors.

6 CONCLUSION

We present an empirical study that aims to explore the best practices of writing documentation in computational notebooks by qualitatively analyzing 80 highly-voted notebooks sampled from Kaggle. In comparison to prior work, our sampled notebooks are indeed well-documented. In particular, the analyses extract nine types of documentation practices, and reveal that documentation interplay with the data science lifecycle. These findings point to promising future work on designing automated features to support data scientists' notebook documentation practices.

[1] Our data corpus is open sourced and available upon request.

ACKNOWLEDGMENTS

We thank all of our participants for their help in the study, and the anonymous reviewers for their valuable feedback.

REFERENCES

[1] Liang Bai and Yanli Hu. 2018. Problem-driven teaching activities for the capstone project course of data science. In *Proceedings of ACM Turing Celebration Conference-China*. 130–131.

[2] Souti Chattopadhyay, Ishita Prasad, Austin Z Henley, Anita Sarma, and Titus Barik. 2020. What's Wrong with Computational Notebooks? Pain Points, Needs, and Design Opportunities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.

[3] Ruijia Cheng and Mark Zachry. 2020. Building Community Knowledge In Online Competitions: Motivation, Practices and Challenges. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW2 (2020), 1–22.

[4] Sergio Cozzetti B de Souza, Nicolas Anquetil, and Káthia M de Oliveira. 2005. A study of the documentation essential to software maintenance. In *Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information*. 68–75.

[5] Jesus Fernandez-Bes, Jerónimo Arenas-García, and Jesús Cid-Sueiro. [n.d.]. Energy generation prediction: Lessons learned from the use of Kaggle in Machine Learning Course. *Group* 7, 8 ([n. d.]), 9.

[6] R Stuart Geiger, Nelle Varoquaux, Charlotte Mazel-Cabasse, and Chris Holdgraf. 2018. The types, roles, and practices of documentation in data analytics open source software libraries. *Computer Supported Cooperative Work (CSCW)* 27, 3-6 (2018), 767–802.

[7] Youyang Hou and Dakuo Wang. 2017. Hacking with NPOs: collaborative analytics and broker roles in civic data hackathons. *Proceedings of the ACM on Human-Computer Interaction* 1, CSCW (2017), 53.

[8] JavaDoc 2020. JavaDoc. https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html.

[9] Kaggle Competition 2020. House Prices - Advanced Regression Techniques. https://www.kaggle.com/c/house-prices-advanced-regression-techniques.

[10] Kaggle Competition 2020. Titanic - Machine Learning from Disaster. https://www.kaggle.com/c/titanic.

[11] Mira Kajko-Mattsson. 2005. A survey of documentation practice within corrective maintenance. *Empirical Software Engineering* 10, 1 (2005), 31–55.

[12] Malin Källén, Ulf Sigvardsson, and Tobias Wrigstad. 2020. Jupyter Notebooks on GitHub: Characteristics and Code Clones. *arXiv preprint arXiv:2007.10146* (2020).

[13] Mary Beth Kery, Marissa Radensky, Mahima Arya, Bonnie E John, and Brad A Myers. 2018. The story in the notebook: Exploratory data science using a literate programming tool. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–11.

[14] Mary Beth Kery, Marissa Radensky, Mahima Arya, Bonnie E. John, and Brad A. Myers. 2018. The Story in the Notebook: Exploratory Data Science Using a Literate Programming Tool. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. ACM, New York, NY, USA, Article 174, 11 pages. https://doi.org/10.1145/3173574.3173748

[15] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. 2016. Jupyter Notebooks – a publishing format for reproducible computational workflows.. In *ELPUB*. 87–90.

[16] Alexander LeClair, Siyuan Jiang, and Collin McMillan. 2019. A neural model for generating natural language summaries of program subroutines. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 795–806.

[17] Jiali Liu, Nadia Boukhelifa, and James R Eagan. 2019. Understanding the role of alternatives in data analysis practices. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 66–76.

[18] Walid Maalej and Martin P Robillard. 2013. Patterns of knowledge in API reference documentation. *IEEE Transactions on Software Engineering* 39, 9 (2013), 1264–1282.

[19] Yaoli Mao, Dakuo Wang, Michael Muller, KUSH VARSHNEY, IOANA Baldini, CASEY Dugan, and ALEKSANDRA MOJSILOVIĆ. 2020. How Data Scientists Work Together With Domain Experts in Scientific Collaborations. In *Proceedings of the 2020 ACM conference on GROUP*. ACM.

[20] Michael Muller, Ingrid Lange, Dakuo Wang, David Piorkowski, Jason Tsay, Q. Vera Liao, Casey Dugan, and Thomas Erickson. 2019. How Data Science Workers Work with Data: Discovery, Capture, Curation, Design, Creation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. ACM, New York, NY, USA, Article 126, 15 pages. https://doi.org/10.1145/3290605.3300356

[21] Yoann Padioleau, Lin Tan, and Yuanyuan Zhou. 2009. Listening to programmers—Taxonomies and characteristics of comments in operating system code. In *2009 IEEE 31st International Conference on Software Engineering*. IEEE, 331–341.

[22] Jeffrey M. Perkel. 2018. Why Jupyter is data scientists' computational notebook of choice. *Nature* 563 (2018), 145. https://doi.org/10.1038/d41586-018-07196-1

[23] Mohammed Suhail Rehman. 2019. Towards Understanding Data Analysis Workflows using a Large Notebook Corpus. In *Proceedings of the 2019 International Conference on Management of Data*. 1841–1843.

[24] Tobias Roehm, Rebecca Tiarks, Rainer Koschke, and Walid Maalej. 2012. How do professional developers comprehend software?. In *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 255–265.

[25] Adam Rule, Aurélien Tabard, and James D Hollan. 2018. Exploration and explanation in computational notebooks. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[26] Lin Shi, Hao Zhong, Tao Xie, and Mingshu Li. 2011. An empirical study on evolution of API documentation. In *International Conference on Fundamental Approaches To Software Engineering*. Springer, 416–431.

[27] Krishna Subramanian, Nur Hamdan, and Jan Borchers. 2020. Casual Notebooks and Rigid Scripts: Understanding Data Science Programming. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 1–5.

[28] Christoph Tauchert, Peter Buxmann, and Jannis Lambinus. 2020. Crowdsourcing Data Science: A Qualitative Analysis of Organizations' Usage of Kaggle Competitions. In *Proceedings of the 53rd Hawaii International Conference on System Sciences*.

[29] Dakuo Wang, Josh Andres, Justin Weisz, Erick Oduor, and Casey Dugan. 2021. AutoDS: Towards Human-Centered Automation of Data Science. In *Proceedings of the CHI 2021*.

[30] Dakuo Wang, Q. Vera Liao, Yunfeng Zhang, Udayan Khurana, Horst Samulowitz, Soya Park, Michael Muller, and Lisa Amini. 2021. How Much Automation Does a Data Scientist Want?. In *preprint*.

[31] Dakuo Wang, Justin D Weisz, Michael Muller, Parikshit Ram, Werner Geyer, Casey Dugan, Yla Tausczik, Horst Samulowitz, and Alexander Gray. 2019. Human-AI Collaboration in Data Science: Exploring Data Scientists' Perceptions of Automated AI. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 1–24.

[32] Amy X Zhang, Michael Muller, and Dakuo Wang. 2020. How do Data Science Workers Collaborate? Roles, Workflows, and Tools. *arXiv preprint arXiv:2001.06684* (2020).

[33] Ge Zhang, Mike A Merrill, Yang Liu, Jeffrey Heer, and Tim Althoff. 2020. CORAL: COde RepresentAtion Learning with Weakly-Supervised Transformers for Analyzing Data Analysis. *arXiv preprint arXiv:2008.12828* (2020).