

UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Design and Use of Computational Notebooks

Permalink

<https://escholarship.org/uc/item/0dc498tf>

Author

Rule, Adam

Publication Date

2018

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA SAN DIEGO

Design and Use of Computational Notebooks

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Cognitive Science

by

Adam Carl Rule

Committee in charge:

Professor James D. Hollan, Chair
Professor Robert El-Kareh
Professor William Griswold
Professor Philip Guo
Professor Gloria Mark
Professor Bradley Voytek

2018

Copyright
Adam Carl Rule, 2018
All rights reserved.



This work is licensed under a Creative Commons Attribution 4.0 International License.
To view a copy of this license, visit <https://creativecommons.org/licenses/by/4.0/>

The dissertation of Adam Carl Rule is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California San Diego

2018

DEDICATION

To Josh, who taught me to think

EPIGRAPH

*The words of the wise are like goads, and like nails firmly fixed are the collected sayings;
they are given by one Shepherd. My son, beware of anything beyond these.
Of making many books there is no end, and much study is a weariness of the flesh.*

Ecclesiastes 12:11-12

TABLE OF CONTENTS

Signature Page	iii
Dedication	iv
Epigraph	v
Table of Contents	vi
List of Figures	ix
List of Tables	x
Acknowledgements	xi
Vita	xiii
Abstract of the Dissertation	xiv
1 Introduction	1
1.1 The Challenge: Tracking and Sharing Data Analyses	3
1.2 Thesis	6
1.3 Contributions	6
2 Related Work	10
2.1 Data Analysis and Sensemaking	10
2.2 Computational Notebooks	13
2.3 Collaborative Visual Analytics	16
2.4 Narrative and Storytelling	17
2.5 Technical Debt	19
2.6 Active Reading	20
3 Exploration and Explanation: A Central Tension in Data Analysis	21
3.1 Introduction	21
3.2 Study 1: Analyzing 1 Million Jupyter Notebooks	23
3.2.1 Methods	24
3.2.2 Results	24
3.2.3 Discussion	27
3.3 Study 2: Narrative in Academic Notebooks	29
3.3.1 Methods	30
3.3.2 Results	30
3.3.3 Discussion	33
3.4 Study 3: Interviewing Academic Data Analysts	34

3.4.1	Methods	34
3.4.2	Results	35
3.5	Discussion	42
3.6	Conclusion	43
3.7	Synthesis	44
4	Aiding Communication of Data Analyses through Flexible Organization and Navigation	46
4.1	Introduction	46
4.2	Workshops: Identifying Design Opportunities	48
4.2.1	Brainstorming Workshop	49
4.2.2	Paper Prototyping Workshop	51
4.3	Janus: History and Hierarchy for Computational Notebooks	52
4.3.1	History: Cell and Notebook Versions	52
4.3.2	Hierarchy: Hiding Cells, Inputs, and Outputs	53
4.4	Study 1: Formative Study with Novice Analysts	54
4.4.1	Methods	55
4.4.2	Measures	57
4.4.3	Results	58
4.4.4	Discussion	61
4.5	Study 2: Technology Probe with Expert Analysts	63
4.5.1	Methods	64
4.5.2	Results	65
4.6	Discussion	70
4.7	Conclusion	72
4.8	Synthesis	72
5	ActiveNotes: Interactive Notes for Clinicians	74
5.1	Introduction	74
5.2	Background: Fragmented Medical Records	75
5.3	ActiveNotes Prototype	78
5.4	Evaluation	81
5.5	Results	81
5.6	Discussion	85
5.7	Conclusion	86
5.8	Synthesis	88
6	Conclusion	90
6.1	Contributions	91
6.2	Implications	92
6.2.1	Technical Interventions	92
6.2.2	Social Interventions	93
6.2.3	Wider Use of Notebooks	94

6.2.4	Beyond Notebooks	95
6.2.5	Education	95
6.3	Future Work	96
6.3.1	Characterizing Data Analysis	96
6.3.2	Redesigning Computational Notebooks	97
6.3.3	Computational Notebooks for Non-Programmers	97
6.3.4	Reading and Reuse of Computational Notebooks	98
6.3.5	Other Barriers to Sharing Data Analyses	99
6.4	Final Thoughts	100
	Bibliography	101

LIST OF FIGURES

Figure 1.1:	Two methods of tracking the process of exploratory data analysis . .	4
Figure 1.2:	A computational notebook	5
Figure 1.3:	Contributions of this dissertation	7
Figure 2.1:	Models of the data analysis process	12
Figure 2.2:	Three types of computational notebook	15
Figure 3.1:	The growth of Jupyter Notebooks on Github	23
Figure 3.2:	Notebook length	25
Figure 3.3:	Notebook organizational features	26
Figure 4.1:	The Janus Jupyter Notebook extension	53
Figure 4.2:	The example notebook comparing housing prices used in Study 1 . .	56
Figure 4.3:	Task performance with Janus	59
Figure 5.1:	ActiveNotes in use	79
Figure 5.2:	ActiveNotes checkout screen	80
Figure 6.1:	Example visualization of how one notebook evolved over time . . .	96

LIST OF TABLES

Table 3.1: Length and content of academic computational notebook by genre . .	32
Table 5.1: Shorthand used when placing orders	82

ACKNOWLEDGEMENTS

I owe a great debt to teachers, colleagues, family, and friends. There is not space here to do their generosity justice but I must make some attempt to thank them, or at least alert the reader that these pages bear their marks as well as my own.

Jim, thank you for treating me as a junior colleague from the start, for being fascinated by the people and stories behind research, for your steady confidence in me, and for teaching me that knotty problems are often simpler after a morning surf. Aurélien, in many ways I feel I have simply followed your footsteps as you provided council and encouragement from across the pond. Rob, Bill, Philip, Gloria, and Brad, thank you for being the epitome of what a committee should be, diverse in expertise, generous and gracious with advice, and simply a joy to be around. And to the members of the Design Lab, thank you for giving me an intellectual home. I hope to do you proud.

Mom and Dad, thank you for the roots and wings, the grounding and the freedom. Josh, this dissertation is in many ways a product of your example over the years of a man who takes the life of the mind so seriously as to laugh in delight at the world. I do not think I would have become a man of letters but for my great respect and admiration for you. Oresta, what a ten seasons it has been! Thank you for taking care of me as I, stumbling at times, try to take care of you.

And LORD, thank you for this mind to think and world to explore.

CHAPTER 3, in part, includes portions of material as it appears in *Exploration and Explanation in Computational Notebooks* by Adam Rule, Aurelien Tabard, and Jim Hollan in the proceedings of the ACM international conference on Human Factors in Computing Systems (CHI '18). (<https://doi.org/10.1145/3173574.3173606>) The dissertation author was the primary investigator and author of this paper.

CHAPTER 4, in part, includes portions of material in *Aiding Communication of*

Complex Data Analyses in Computational Notebooks by Adam Rule, Ian Drosos, Aurelien Tabard, and Jim Hollan as it was submitted to the ACM international conference on Computer Supported Cooperative Work. The dissertation author was the primary investigator and author of this paper.

CHAPTER 5, in part, includes portions of material as it appears in *Validating free-text order entry for a note-centric EHR* by Adam Rule, Steven Rick, Michael Chiu, Phillip Rios, Shazia Ashfaq, Alan Calvitti, Wesley Chan, Nadir Weibel, and Zia Agha in the proceedings of the 2015 annual symposium of the American Medical Informatics Association (AMIA '15). The dissertation author was the primary author of this paper.

VITA

- 2011 B. S. in Industrial Engineering, *Highest Honors*, University of Illinois, Urbana-Champaign
- 2013 M. S. in Human Centered Design and Engineering, University of Washington
- 2018 Ph. D. in Cognitive Science, University of California, San Diego

PUBLICATIONS

Adam Rule, Aurelien Tabard, Jim Hollan. (2018) *Exploration and Explanation in Computational Notebooks*. In proceedings of the ACM international conference on Human Factors in Computing Systems (CHI '18).

Adam Rule, Aurelien Tabard, Jim Hollan. (2017) *Using Visual Histories to Reconstruct the Mental Context of Suspended Activities*. *Human-Computer Interaction* 32 (5-6), 511-558.

Adam Rule, Steven Rick, Michael Chiu, Phillip Rios, Shazia Ashfaq, Alan Calvitti, Wesley Chan, Nadir Weibel, Zia Agha. (2015) *Validating free-text order entry for a note-centric EHR*. In proceedings of the 2015 annual symposium of the American Medical Informatics Association (AMIA '15).

Adam Rule, Aurelien Tabard, Karen Boyd and Jim Hollan. (2015) *Restoring the Context of Interrupted Work with Desktop Thumbnails*. In proceedings of the 37th Annual Conference of the Cognitive Science Society (CogSci '15).

Adam Rule, Jodi Forlizzi. (2012) *Designing interfaces for multi-user, multi-robot systems*. In proceedings of the ACM international conference on Human Robot Interaction (HRI '12).

ABSTRACT OF THE DISSERTATION

Design and Use of Computational Notebooks

by

Adam Carl Rule

Doctor of Philosophy in Cognitive Science

University of California San Diego, 2018

Professor James D. Hollan, Chair

Individuals and organizations increasingly rely on data analysis to generate insights and make decisions. Yet, small changes in how data are collected, cleaned, or modeled can lead to vastly different results. If data-driven insights are to be reviewed, reused, or trusted the process used to generate them must be tracked and communicated in detail. But data analysis is typically an iterative and exploratory process that is hard to articulate, especially when it involves programming. Computational notebooks aim to ease tracking and sharing of complex analyses by enabling analysts to write rich *computational narratives* combining executable code, interactive visualizations, and explanatory text in a single document. While millions of people use computational note-

books, we know little about how they use them, or how well they help people track and share complex analyses.

In this dissertation I present three studies of how people currently use computational notebooks, demonstrating that few notebooks, even those published alongside academic papers, have much in the way of narrative. Instead, most notebooks are loose collections of notes and scripts that even the original analyst struggles to understand. I then present two systems demonstrating how computational notebooks might be designed to support clearer communication of complex analyses. The first system, Janus, shows how current notebooks might be modified to aid both ongoing analysis and later communication by adding interactive hierarchy for selectively showing and hiding portions of the notebook. The second system, ActiveNotes, a prototype clinical note editor, demonstrates how computational notebooks might support data-driven work even when programming is not the primary means of interacting with data.

Together, these studies demonstrate that tracking and sharing of complex analyses is hindered by a tension between exploration and explanation, but that computational notebooks and other media can reduce this tension by supporting not only the combination of, but also flexible organization and navigation of analytical steps, explanatory text, and computed results.

1 Introduction

This dissertation explores how people use computational notebooks to perform, document, and share data-driven work. It finds a tension between exploration and explanation hinders data analysts from clearly communicating complex analyses, but that computational notebooks can reduce this tension by supporting flexible navigation and organization of analytical components such as executable code, computed results, and explanatory text. This dissertation also explores how computational notebooks might help professionals document and share data-driven work in domains where general-purpose programming is not the primary means of interacting with data. Together these investigations help us understand how computational notebooks and other interactive technologies can help individuals and organizations think with data.

The cost of collecting, storing, and manipulating data has fallen dramatically over the past 50 years, enabling data to multiply in nearly every sphere of life [80]. Businesses increasingly rely on data about their customers and products to generate value [50]. Governments munge data to learn about their constituents and set policy [57, 69, 70]. Scientists collect and model data to study domains as diverse as engineering, the life sciences, and arts. Individuals collect data about their physical activity, spending, and happiness to foster self-awareness [19, 60]. Profits, policy, innovation, and health all increasingly rely on collecting and analyzing data.

But those who understand data are in short supply. McKinsey, a consulting firm, estimates that in 2018 the United States alone will face “a shortage of 140,000 to 190,000 people with analytical expertise and 1.5 million managers and analysts with the skills to understand and make decisions based on the analysis of big data” [62]. There is an urgent need to both train new analysts and develop tools and techniques that enable them to work more effectively.

Data analysis has been described as simply “looking at data to see what it seems to say” [99], but knowing where and how to look is not as simple as it may seem. Insights derived from data are highly dependent on the questions asked and the methods used to inspect them. Two analysts given the same dataset may draw vastly different conclusions [40, 79]. Even during analysis, deciding what to do next often requires extensive knowledge of what one has already done with the data and why [27, 38]. Moreover, the scale of data analyzed today typically requires writing and running numerous small computer programs to collect, clean, and model them. The workings of each of these programs may be difficult to understand in isolation, much less when they are combined.

These challenges are compounded by the fact that data analysis is increasingly collaborative [36, 50], especially in scientific domains. Even if they work in the same office or field, collaborators may have vastly different skill-sets, terminologies, and goals for an analysis [25]. Assumptions need to be made explicit and methods explained in detail if data and the insights derived from them are to be clearly communicated [33]. Those who wish to contribute to open science and publicly share their work face even greater challenges making their analyses legible not just to a particular group of colleagues, but to anyone who might read them. Explaining the exploratory process of data analysis is rarely a straightforward task.

1.1 The Challenge: Tracking and Sharing Data Analyses

Data analysts need to keep a detailed record of their analytical steps, reasoning, and results if others are to review, resume, or build on their work. However, tracking and sharing data analysis is complicated by the number and diversity of steps involved [29, 50, 92] as well as the professional judgment guiding their selection and execution [33]. In the end, most analysts have only incomplete or messy records of their process, especially when their analysis involves programming.

Consider Figure 1.1 which shows two different methods of tracking data analysis employed by two different biologists. Figure 1.1(a) shows a partial list of files in one of the biologist's computer folders [29]. He had run the same analysis script over and over again, tweaking the parameters of the script each time and generating dozens of figures whose similar filenames contain the settings of each run (e.g., "ld2" and "5000kb"). When resuming the analysis after a break, this analyst had difficulty recalling which run had produced the most promising results. Figure 1.1(b) shows a second biologist's attempts to track her analysis [92]. Since she had to manage data across multiple websites and applications she chose to track her steps in a Word document. Her cryptic notes are a collage of activity: manipulating data from her command line, referencing notes in another notebook, instructions to copy-paste and rerun a step, file paths to yet more data, notes on variations of the analysis with different parameters, and even raw data pasted directly into the Word document itself.

Both methods demonstrate some of the challenge of tracking and sharing data analysis. Iterative analyses tend to produce multiple similar results that take time and energy to document and distinguish. Analyses often require multiple steps across multiple tools, none of which keeps a full record of the process. Manually tracking one's steps is laborious and often produces cryptic notes that even the original analyst has a hard time understanding. In the end neither record is of much use to a collaborator.

```

xterm
rwrwrwr 1 elinor sequence 118540 Jul 6 13:20 OSA_May2011/QQ_pngs/GREY_GREY_re10.25.maf0.05.young_v_old.cov1.ld5.5000kb.P.wide.p5e-05.png
rwrwrwr 1 elinor sequence 118375 Jul 6 13:22 OSA_May2011/QQ_pngs/GREY_GREY_re10.25.maf0.05.young_v_old.cov1.ld8.1000kb.P.p0001.png
rwrwrwr 1 elinor sequence 130972 Jul 6 13:21 OSA_May2011/QQ_pngs/GREY_GREY_re10.25.maf0.05.young_v_old.cov1.ld8.1000kb.P.p0005.png
rwrwrwr 1 elinor sequence 117048 Jul 6 13:21 OSA_May2011/QQ_pngs/GREY_GREY_re10.25.maf0.05.young_v_old.cov1.ld8.1000kb.P.p1e-05.png
rwrwrwr 1 elinor sequence 117830 Jul 6 13:21 OSA_May2011/QQ_pngs/GREY_GREY_re10.25.maf0.05.young_v_old.cov1.ld8.1000kb.P.p5e-05.png
rwrwrwr 1 elinor sequence 119079 Jul 6 13:22 OSA_May2011/QQ_pngs/GREY_GREY_re10.25.maf0.05.young_v_old.cov1.ld8.5000kb.P.wide.p0001.png
rwrwrwr 1 elinor sequence 131628 Jul 6 13:21 OSA_May2011/QQ_pngs/GREY_GREY_re10.25.maf0.05.young_v_old.cov1.ld8.5000kb.P.wide.p0005.png
rwrwrwr 1 elinor sequence 117697 Jul 6 13:22 OSA_May2011/QQ_pngs/GREY_GREY_re10.25.maf0.05.young_v_old.cov1.ld8.5000kb.P.wide.p1e-05.png
rwrwrwr 1 elinor sequence 118502 Jul 6 13:21 OSA_May2011/QQ_pngs/GREY_GREY_re10.25.maf0.05.young_v_old.cov1.ld8.5000kb.P.wide.p5e-05.png
rwrwrwr 1 elinor sequence 127945 Jul 6 13:14 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld2.2500kb.P.p0001.png
rwrwrwr 1 elinor sequence 159282 Jul 6 13:15 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld2.2500kb.P.p0005.png
rwrwrwr 1 elinor sequence 127989 Jul 6 13:14 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld2.2500kb.P.p1e-05.png
rwrwrwr 1 elinor sequence 128005 Jul 6 13:14 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld2.2500kb.P.p5e-05.png
rwrwrwr 1 elinor sequence 128528 Jul 6 13:15 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld2.5000kb.P.wide.p0001.png
rwrwrwr 1 elinor sequence 159950 Jul 6 13:15 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld2.5000kb.P.wide.p0005.png
rwrwrwr 1 elinor sequence 128632 Jul 6 13:15 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld2.5000kb.P.wide.p1e-05.png
rwrwrwr 1 elinor sequence 127925 Jul 6 13:15 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld5.2500kb.P.p0001.png
rwrwrwr 1 elinor sequence 159257 Jul 6 13:15 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld5.2500kb.P.p0005.png
rwrwrwr 1 elinor sequence 127970 Jul 6 13:15 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld5.2500kb.P.p1e-05.png
rwrwrwr 1 elinor sequence 127982 Jul 6 13:15 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld5.2500kb.P.p5e-05.png
rwrwrwr 1 elinor sequence 128470 Jul 6 13:16 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld5.5000kb.P.wide.p0001.png
rwrwrwr 1 elinor sequence 159928 Jul 6 13:15 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld5.5000kb.P.wide.p0005.png
rwrwrwr 1 elinor sequence 128590 Jul 6 13:15 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld5.5000kb.P.wide.p1e-05.png
rwrwrwr 1 elinor sequence 128606 Jul 6 13:15 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld5.5000kb.P.wide.p5e-05.png
rwrwrwr 1 elinor sequence 127974 Jul 6 13:16 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld8.2500kb.P.p0001.png
rwrwrwr 1 elinor sequence 159303 Jul 6 13:15 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld8.2500kb.P.p0005.png
rwrwrwr 1 elinor sequence 128018 Jul 6 13:16 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld8.2500kb.P.p1e-05.png
rwrwrwr 1 elinor sequence 128035 Jul 6 13:17 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld8.2500kb.P.p5e-05.png
rwrwrwr 1 elinor sequence 128487 Jul 6 13:16 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld8.5000kb.P.wide.p0001.png
rwrwrwr 1 elinor sequence 159941 Jul 6 13:16 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld8.5000kb.P.wide.p0005.png
rwrwrwr 1 elinor sequence 128599 Jul 6 13:16 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld8.5000kb.P.wide.p1e-05.png
rwrwrwr 1 elinor sequence 128616 Jul 6 13:15 OSA_May2011/QQ_pngs/IWH_IWH_re10.25.maf0.05.cc.cov1.ld8.5000kb.P.wide.p5e-05.png

```

(a)

The image shows a Word document with several paragraphs of text, each highlighted in a different color and annotated with arrows pointing to descriptive labels on the right. The text includes:

- Command lines:** A block of shell commands starting with 'new caca seqs from PFI : ds /mount/glabrata/multi-strains/CACA : mkdir ecc : cp dedans CACA100-121_out (trimmes aux deux extremités) : emacs CACA100-121_cut : viré seqs trop courtes<-190 : viré plusieurs centaines de seqs : voir "procnewcaca070209.doc" :
- File path to data:** A block starting with 'we -l CACANewnames : 13344 CACANewnames : 13344 seqs<-490nt : blastall p blastx -d CAGLrefaa.fasta -i CACANewcut.fasta -F G 11 E 1 m 8 o tCACA07.txt & mv tCACA07.txt resCACANewCF.txt :
- Variations for other sequences with other parameters:** A block starting with 'Recommandé pour les autres seqs aussi : cfair@electre~\$ pwd : /home1/Gnlev/cfair/genopole/KLBA : il y a KLBAtrim.fasta : cp KLBAtrim.fasta KLBAtrimCF.fasta : emacs KLBAtrimCF.fasta : viré seqs :
- DNA sequence extremely frequent:** A block of raw DNA sequence data starting with 'seq de seqs nome ca : ATGAGG...'

(b)

Figure 1.1: Two methods of tracking the process of exploratory data analysis. (a) A partial list of output files for different runs of the same analysis with file names reflecting the parameters of each run, from [29]. (b) A Word document describing a biologist’s analytical steps across multiple software tools, from [92].

One increasingly popular means of addressing these challenges is to conduct analyses, at least the growing share involving programming, in computational notebooks (Figure 1.2). These enable analysts to iteratively execute analytical code and interleave it with computed results and explanatory text. Whereas, before, analysts had to copy code and results from various files into a separate report (Figure 1.1(b)), computational notebooks enable them to write, run, and explain their analyses in a single document. In place of large collections of similarly named files (Figure 1.1(a)), analysts can combine all explorations in a single annotated document. Computational notebooks have seen widespread adoption in recent years [28] and been hailed by some as a replacement to academic papers as the best vehicle for sharing scientific results. [88]. *Yet, while millions of people use computational notebooks for a variety of data-driven activities, we know little about how they actually use them or how well notebooks address the challenge of tracking and sharing complex data analyses.*

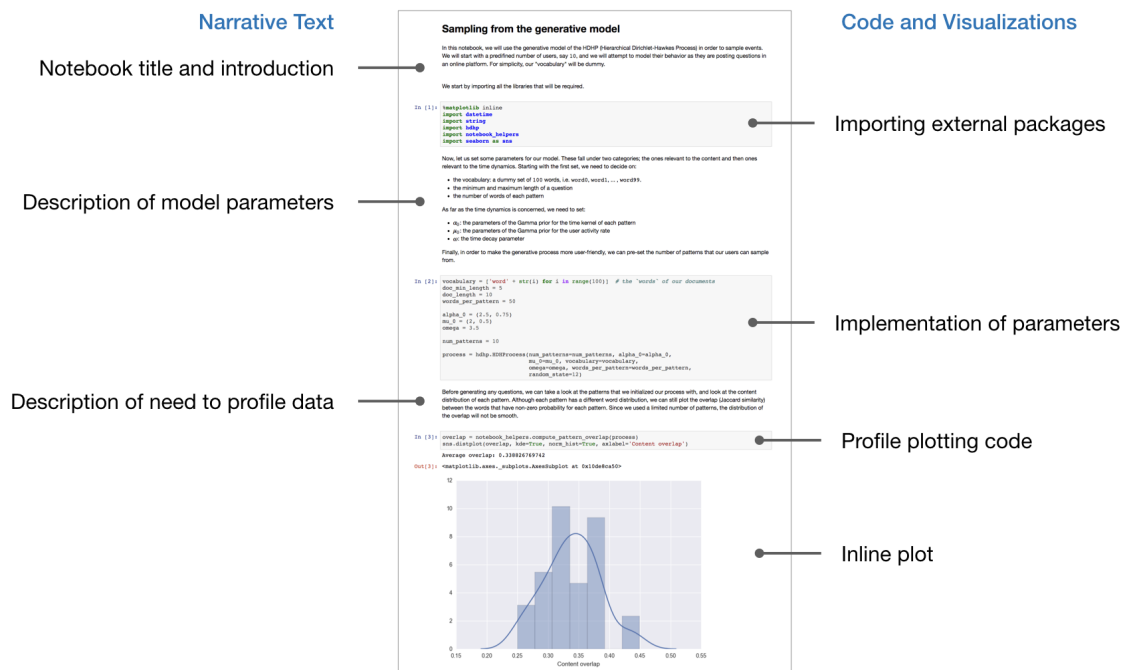


Figure 1.2: A computational notebook combining code, visualizations, text.

1.2 Thesis

This dissertation explores challenges analysts face tracking and sharing their data analyses. It focuses on how they currently use computational notebooks to do so, and how these notebooks might be designed to better support their needs. This dissertation also begins to explore how the paradigm of computational notebooks might support data-driven work in domains where programming with a general-purpose programming language is not the primary means of interacting with data, domains such as engineering, healthcare, and government. Underlying these investigations is the thesis:

Understanding how the tension between exploring data and explaining process manifests itself in practice makes it possible to design software that enables analysts to document and share their work more effectively.

1.3 Contributions

This dissertation has four types of contributions: empirical results, theoretical perspectives, prototype systems, and an open dataset. Here I summarize these contributions in the order they will be presented in the chapters that follow (Figure 1.3).

CHAPTER 3 presents three studies characterizing the use of *computational narrative* (i.e., the interleaving of code and visualizations with explanatory text) in computational notebooks. In the first study I analyze over 1 million Jupyter Notebooks hosted publicly on GitHub, finding that more than a quarter lack even a single word of explanatory text. Even those notebooks that had explanatory text were mostly loose collections of notes and scripts without a coherent structure. In a second study I systematically code over 200 notebooks supplementing academic publications, finding that even when these notebooks have explanatory text, only about a third use that text to discuss analytical reasoning or to interpret results. Even among these academic note-

Studies		Contributions
Analysis of 1 million+ Jupyter Notebooks on Github	Chapter 3	Evidence of lack of text in most computational notebooks
Systematic coding of 200+ academic Jupyter Notebooks		Theory that a tension between exploration and explanation is central to data analysis
Interviews with academic data analysts		Open dataset containing 1 million+ computational notebooks
Design workshops with academic data analysts	Chapter 4	Evidence that hierarchy helps analysts perform and communicate their work
Formative study with novice data analysts		Theory that notebooks can reduce tension between exploration and explanation by supporting flexible organization and navigation
Technology probe with experienced data analysts		Prototype notebook cleaning extension, Janus
Evaluation of free-text order entry with clinicians	Chapter 5	Evidence that clinicians can use free-text within clinical notes to place medication orders
		Theory that computational notebooks can use domain specific languages to support non-programmers' data-driven work
		Prototype clinical note editor, ActiveNotes

Figure 1.3: Contributions of this dissertation including empirical results, prototype systems, theoretical perspectives, and an open dataset.

books where we might expect authors to include detailed explanations of their work to support future research, most authors used explanatory text to simply label the steps of their analysis. In a third study I interview 15 academic data analysts, finding that this lack of narrative stems from the tendency for exploratory analyses to produce messy

notebooks that are difficult to understand. Cleaning and sharing these notebooks takes more effort than copying a final figure into an email, which may be all that analysts feel their collaborators want to see.

Together these findings support a theoretical perspective that a tension between exploration and explanation makes it difficult to track and share data analyses, regardless of the medium involved. The iterative and messy process of exploring data is fundamentally at odds with the careful, reflective process of explaining what the results of those explorations mean. In addition to these empirical findings and theoretical perspective, I also released all data from the first study, including over a million computational notebooks, as a single dataset for others to download and study (<https://doi.org/10.6075/J0JW8C39>).

In CHAPTER 4 I build on these findings by exploring how computational notebooks might be redesigned to encourage clearer communication of complex data analyses, in particular by including more explanatory text and explicit organization. Taking the point-of-view that notebooks need to provide an immediate benefit to organization and annotation activities I design and develop Janus, a Jupyter Notebook extension that enables analysts to add history and hierarchy to their notebooks. Through two studies, the first a formative study with novice analysts and the second a multi-week technology probe with expert analysts, I demonstrate that hierarchy in particular enables analysts to flexibly organize and navigate their notebooks in ways that support both the ongoing analysis and later communication of results. These findings support the theoretical perspective that flexible organization and navigation of code, visualizations, and text can help reduce the tension between exploration and explanation by providing a lightweight and fast way to tailor notebooks to the task at hand.

In CHAPTER 5 I explore how the the paradigm of computational notebooks might support data-driven work in a domain where general-purpose programming is not the primary means of interacting with data. With a team of collaborators I helped design

and test ActiveNotes, a prototype clinical note editor that enables clinicians to place medication orders while typing free-text notes. This work provides empirical results about how clinicians might use free-text order entry, and supports the theoretical perspective that notebooks mixing free-text and commands written in a domain-specific language might support data-driven work outside traditional data analysis.

Together these investigations help us understand how computational notebooks and other interactive media might help individuals and organizations think with data.

2 Related Work

This thesis builds on prior work characterizing the iterative process of data analysis and exploring the design of systems for collaborative visual analytics. It also touches on the accrual of technical debt while analyzing data and the role of narrative and active reading in comprehending analyses. In this chapter I survey relevant work from each domain and discuss how this dissertation builds on it.

2.1 Data Analysis and Sensemaking

In the introduction to his seminal book *Exploratory Data Analysis* John Tukey memorably described data analysis as “looking at data to see what it seems to say” [99]. This definition is vague, which may have been Tukey’s point. Data analysis includes a variety of activities and tools ranging from tracking virtues in a paper notebook (as Ben Franklin famously did) to tabulating financial results in Excel. A more precise definition might exclude whole communities of practice. It might also fail to convey the creative and exploratory process of data analysis. Tukey in particular stressed that exploratory analysis and plotting, preferably by hand, should almost always precede more exact statistical tests because hand-guided exploration can identify interesting trends, check assumptions, and inform selection of appropriate analytical techniques.

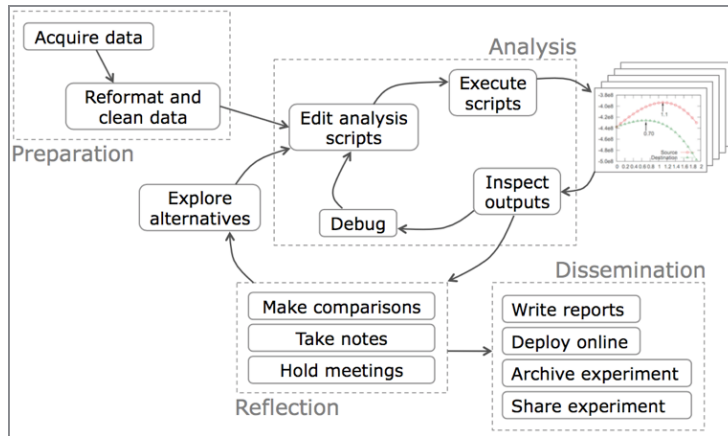
From tracking how millions of people use social media to big science initiatives collecting petabytes of data, the scale of data analyzed today routinely exceeds what

can be plotted by hand [63, 80]. Instead, programming and computerized analysis has become a primary means of interacting with data. Despite this digitization, recent accounts of data analysis echo Tukey's description of an iterative and imprecise art.

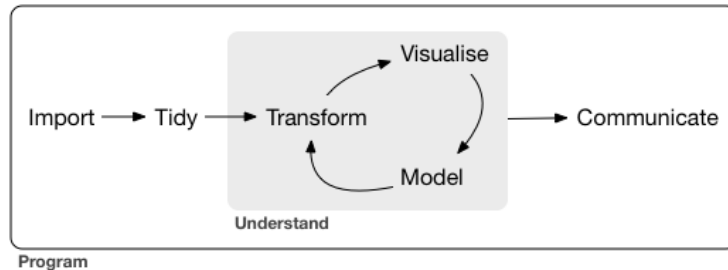
Based on interviews with 35 enterprise data analysts, Kandel et al. characterized data analysis as an iterative process with five overlapping phases (Discovery, Wrangle, Profile, Model, Report) [50]. Guo similarly characterized data analysis as an iterative process with four phases (Preparation, Analysis, Reflection, Dissemination) (Figure 2.1(a)) [29]. Hadley Wickham, core developer of the tidyverse of R packages for data analysis, prefers a six phase model (Import, Tidy, Transform, Visualize, Model, Communicate) (Figure 2.1(b)) [104].

While different in detail, these models are remarkably similar at a high level. They all cast data analysis as an iterative process where insights generated at a later stage can cause analysts to jump back to an earlier one. They all bookended the process with stages of finding and cleaning data on one end and communicating results on the other. And they all emphasize that analysts try different versions of the same analysis, slowly improve their methods over time, and hit numerous "dead ends" before finding an explanation they feel "fits" the data. In each model data analysis is not a predetermined science but an exploratory process making sense of data.

Like other forms of sensemaking [81], the process used to collect, explore, and make sense of data can have a significant impact on the sense made. Small changes to how data are collected, cleaned, and analyzed can lead to vastly different results. For example, one highly cited economics paper claimed that countries with a public debt greater than 90% of GDP average -0.1% annual economic growth [79]. Using slightly different methods and fixing an error in the original authors' Excel file, researchers at another institution placed the figure at 2.2% [40]. This is a difference between recession and stable growth which had massive implications for policy makers in both the United States and European Union.



(a)



(b)

Figure 2.1: Models of the data analysis process by a) Philip Guo and b) Hadley Wickham. Both cast data analysis as an iterative process bookended by acquiring data and communicating results.

This difference highlights how much data analysis relies on professional judgment, which has consequences for the ways analysts document and share their work. While observing analysts at the International Monetary Fund, Harper and Sellen found that the more judgment involved in producing a piece of information, the less suitable it was for sharing over asynchronous electronic media [33]. Analysts at the Fund routinely interpolated missing data or adjusted figures based on their knowledge of countries' data collection practices. Without knowing how and why these adjustments were made, others could easily misinterpret the data and insights drawn from them. And while in some cases the goal may be to produce generalizable knowledge, more

often than not the goal of data analysis may simply be to inform an individual seeking to make a decision [52].

When tracking their work analysts are often simply trying to communicate the analysis to their present or future selves. Taking the perspective of distributed cognition [46] — which views cognition as extending beyond the bounds of an individual brain to include artifacts, other people, and social practices over time — the tools analysts use to perform data analysis are integral parts of their cognitive process. They are part of a reflective conversation [84] the analyst has with their data. The challenge is designing substrates for cognition that work both for analysis in the moment and as a vehicle for collaboration over time.

As in prior work this dissertation focuses on data analysis where a general-purpose programming language is used as the primary means of collecting, cleaning, and modeling data. However, as CHAPTER 5 shows, it also has implications for other forms of analysis where such programming is not involved. To our current understanding of data analysis this dissertation contributes the theory that a tension between data exploration and process explanation lies at the heart of data analysis and hinders collaborative analysis by making it difficult to track and share complex analyses.

2.2 Computational Notebooks

The amount of exploration and professional judgment involved in data analysis necessitates clear documentation of analyses if others — or even the original analyst — are to inspect, replicate, or build on them. Leading work on reproducibility suggests that at a minimum, analysts should distribute the code used in their analyses [74]. Yet, analysts themselves may have difficulty reconstructing the exact process used to generate a result [29]. They often try the same analysis in multiple different ways, producing large collections of opaquely named and interrelated files as demonstrated in Figure

1.1(a). Moreover, the analysis may involve combining and reflecting on media from a variety of digital and paper resources that are not easily shared [92]. Even with all the code and resources in one place there is the additional challenge of making them understandable. As the organizers of the Software Carpentry workshops note, “most researchers are never taught the equivalent of basic lab skills for research computing” [108]. These include placing explanatory comments at the start of every program file, making code dependencies explicit, and separating raw from cleaned data. Much of this organization and annotation is a manual process learned through experience.

One way to address these challenges is to perform data analyses in computational notebooks. In the tradition of Knuth’s literate programming [54], computational notebooks enable analysts to mix code with manual annotations in a single document. While their history can be traced to the JOSS interactive programming language developed in 1963 [87], and notebooks have been available in proprietary software such as Mathematica since the 1980s [109], the notebook computing paradigm has seen rapid adoption only in the past decade thanks to the release of free and open source platforms such as Jupyter Notebook [53] and RStudio [95]. These have millions of users in fields as diverse as education, finance, and the sciences [28] (Figure 2.2). This new generation of notebooks is based on a linear collection of cells, each of which contains rich text or code that can be executed to compute results or generate visualizations. These cells are linearly arranged, but can be reorganized, reshuffled, and executed in any order.

The interactive notebook paradigm is spreading beyond data analysis to other development and visualization environments. ObservableHQ [68] and Iodide [12] both provide computational notebooks based on Javascript where users can both analyze data and change the operation and layout of the notebook with code. Distill, an online academic journal, uses a notebook format to explain complex machine learning research [2]. Likewise, Codestrates recently demonstrated how the notebook paradigm could be used to blur the line between development and use of an application [78].

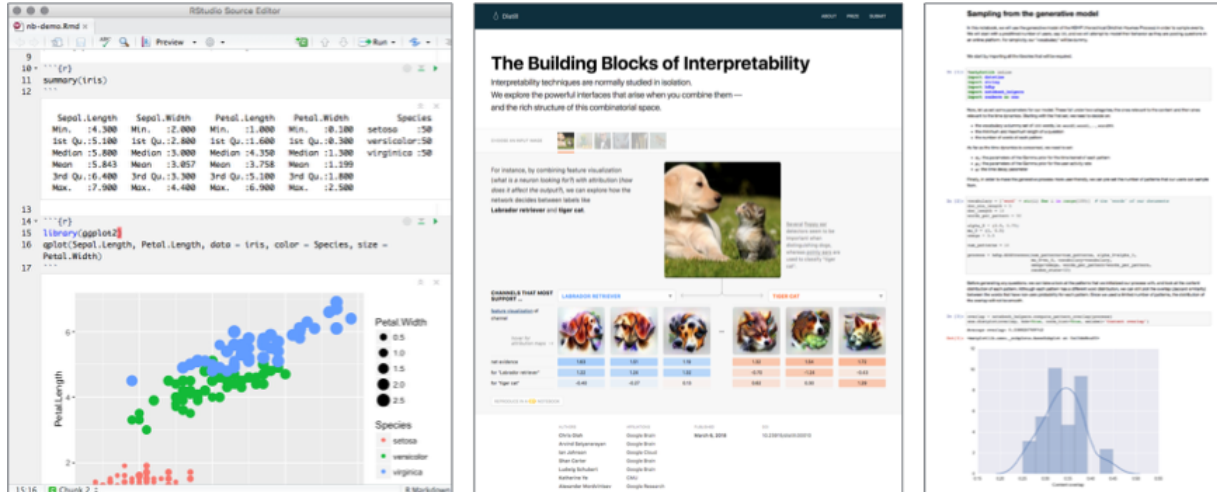


Figure 2.2: Three types of computational notebook. From left to right: an R Notebook, Distil Journal Article, and Jupyter Notebook

The notebook paradigm is clearly powerful and addresses a key challenge of performing data analysis by combining previously distributed code, visualizations, and text in a single document. This combinatorial aspect of notebooks has been studied in the field of human-computer interaction over the last decade with research platforms aiming to make it easier to automatically combine traces of an analysis from multiple sources. Burrrito, for example, instrumented analysts’ computers so that analytical steps were automatically recorded and could be mixed with manual annotations [30]. PRISM enabled computational biologists to mix and reflect on paper and digital media in a hybrid laboratory notebook [92].

But we know very little about how the current generation of computational notebooks are actually being used. There is hope that, in addition to helping analysts manage files, computational notebooks will usher in a new era of open and reproducible analyses where analysts can write clear and compelling computational narratives that both tell the story of the analysis and can be used to replicate it [88].

This dissertation contributes to our understanding of computational notebooks by presenting one of the first studies of how data analysts use and share their notebooks

(though Kery et al. [51] have simultaneously been studying computational notebook use with complimentary results). Moreover it demonstrates how small changes to how notebooks are designed can impact how analysts perform and share analyses.

2.3 Collaborative Visual Analytics

A 2005 report by the National Visualization and Analytics Center listed supporting collaboration as a grand challenge for visualization research [96]. Shortly thereafter Heer and Agrawala synthesized dozens of design decisions for systems supporting collaborative visual analytics into seven key areas: division and allocation of work; common ground and awareness; reference and deixis; incentives and engagement; identity, trust, and reputation; group dynamics; and consensus and decision making [36].

Much attention has been devoted to the second and third areas of common ground and awareness and reference and deixis. For example, Sense.us, ManyEyes, and CommentSpace collectively explore how to annotate and comment on shared online visualizations as well as bookmark particular views for sharing and later access [39, 101, 106]. While informative, all of these systems rely on rich, graphical user interface (GUI) interactions with data and data visualizations. What remains unknown is how to support collaboration when code is a primary means of interacting with data, particularly when code and commentary can be interleaved in a single document.

Projects like Google’s Colaboratory [4] and JupyterLab [28] are beginning to explore this domain by adding real-time collaborative editing, commenting, code-hiding, and tables of contents to computational notebooks. However as [50] notes, collaboration at the level of scripts is rare and it remains to be seen if features such as code-hiding and tables of contents will be enough to encourage analysts to share their “messy” and “throw-away” code. This dissertation explores how to support collaborative analysis when code rather than GUIs or direct manipulation are used to analyze data.

2.4 Narrative and Storytelling

Sir Peter Medawar in his *Induction and Intuition in Scientific Thought* stated that science “begins as a story about a Possible World — a story which we invent and criticize and modify as we go along, so that it ends by being, as nearly as we can make it, a story about real life” [64]. The same could be said of data analysis.

One of the key features of computational notebooks is that they enable analysts to arrange code, visualizations and text in a computational narrative. While computers are good at producing and processing data, humans are much better at understanding stories. I am not an expert in narrative, nor can I summarize millennia of innovation in a few paragraphs. However, here I highlight a few salient aspects of narrative as it relates to data analysis and visualization.

At its core, a narrative is a series of ordered and connected events. The Oxford English Dictionary defines narrative as “An account of a series of events, facts, etc., given in order and with the establishing of connections between them; a narration, a story, an account”. As such, a series of disjointed events is not a narrative (e.g., a twitter newsfeed), nor is a collection of related events that are not in a particular order (e.g., an affinity diagram or mood board). Narratives occur in a variety of media including audio, text, and video, each of which have their own strategies for engaging the audience and moving the story along. While some techniques, such as the “flashback” can be employed across media, others like split-screen sequences in film, are unique to particular media [55].

Since the early 2000s, there has been increasing focus on narrative and storytelling in information visualization. Gershon and Page highlighted the power of narrative to engage and convey information and suggested that information visualization employ well-established narrative techniques such as continuity editing, filling gaps, and redundancy [26]. Segel & Heer built on this foundation by developing a design

space for what they called “narrative visualizations” (visualizations with a set of ordered and connected views), and identified seven distinct genres including magazine, slideshow, and comic-strip [85]. Noting the importance of the order in which data views are presented, Hullman et. al conducted multiple studies of how people sequence information visualization events, finding they tend to prefer a consistent, hierarchical structure [44, 45]. More recently, Kosara & Mackinlay highlighted the need to use different storytelling strategies in different situations with different audiences (e.g., self-running presentations, live presentations, small-group presentations) [55], and Satyanarayan & Heer demonstrated Ellipsis, a tool to support the authoring of narrative visualizations for the web [82].

This prior research demonstrates the challenge of communicating exploratory data analysis, the promise of computational notebooks, and the characteristics of narrative in information visualization. However, as noted in prior research, narrative affordances and strategies differ across media and audience [55]. It remains to be seen what forms of narrative computational notebooks afford and the distinct scenarios in which analysts use them. Moreover, tools such as Ellipsis which support the construction of narratives in interactive information visualizations may not apply when crafting narrative in computational notebooks which need to not only convey insights, but also how they were generated.

This dissertation contributes an understanding of how narrative is used to communicate complex data analyses when interleaved with code and visualizations in a notebook. In contrast to prior work which has focused on the workflow and tools used by professional storytellers such as journalists this dissertation focuses on the larger population of everyday data analysts crafting narratives.

2.5 Technical Debt

In exploring how analysts document data analyses performed with code, this dissertation touches on the topic of technical debt. In software engineering, “technical debt” refers to writing code in ways that save time or energy now but incur a “debt” of future work to align code with best practices [14]. This can include using simple but error-prone algorithms in place of more complex but robust ones, or skimping on documentation. Recent work has moved from calling for the elimination of technical debt to acknowledging its inevitability and learning to manage it [8, 11]. One way to identify sources of technical debt is to look for “code smells”, symptoms of bad design or implementation such as duplicate code, long methods, or missing comments [98].

Fixing the problems identified by code smells can include refactoring, that is changing the structure of code without changing its function, or writing comments and documentation [22]. Recent studies of refactoring highlight that programmers refactor frequently, intersperse refactoring with other program changes, and typically use manual methods to refactor code rather than tools [66]. Likewise studies of commenting and documentation note that programmers do not write or update documentation as quickly or completely as they or their managers would like [21, 59] and have explored ways of automatically generating documentation and comments [63, 89].

These studies provide some guidance for data analysis involving programming, but there are significant differences between data analysis and software engineering in goals and process. Data analysts typically write short scripts rather than production-ready code and can often tolerate more faults in their code than software developers. Moreover, analysts need to document the analytical process itself, not just the operation of a finished piece of code. This dissertation adds to our understanding of technical debt by demonstrating how it accrues in data analysis and is a key barrier to wider sharing and reuse of computational notebooks.

2.6 Active Reading

In studying how analysts navigate and read their notebooks this dissertation also touches on the topic of active reading. In contrast to the linear and often passive process of reading a novel, active reading is the combination of reading, critical thinking, and learning used to understand, compare, and ask questions of a text. It is a staple of knowledge work from students learning from textbooks to researchers keeping up to date with the latest literature.

Several studies have highlighted the affordances of paper for active reading [11], even down to micro-behaviors such as how people flip or tuck papers, or use a finger to bookmark a location [43]. Researchers in human-computer interaction have also developed a number of systems to replicate the affordances of paper in digital media, and to go beyond them. XLibris, for example, enables users to freely annotate a document, but also see clippings of important sections, even providing automatic links to relevant information when a section of text is circled or highlighted [77]. Papiercraft went further to link paper and digital interactions, letting users control digital documents by annotating physical copies of them [61]. Liquidtext explored active reading with multi-touch, enabling users to use gestures on a tablet to extract and link sections of text, or collapse intermediate text to view two disparate sections of a document side by side [94].

This dissertation extends prior work on active reading by demonstrating how more flexible means of organizing and navigating computational notebooks can ease ongoing analysis and enable analysts to quickly re-purpose notebooks for collaboration.

3 Exploration and Explanation: A Central Tension in Data Analysis

This chapter presents three studies of how data analysts use computational notebooks to document and share their work. These include an analysis of over 1 million notebooks shared online, systematically coding 200 notebooks supplementing academic publications, and interviews with 15 academic data analysts. I find a tension between exploring data and explaining process hinders analysts from clearly documenting and sharing their work, particularly when using computational notebooks that aim to support both exploratory and explanatory phases of analysis.

3.1 Introduction

Data analysis is an iterative and exploratory process of turning data into insights [50, 29]. Along the way, analysts produce numerous interrelated artifacts as they write scripts, generate graphs, and jot quick notes [92]. Working “at the speed of thought”, analysts typically view their code and outputs as throw-away and not worth annotating or organizing as many explorations lead to “dead-ends” [50]. Over time this lack of annotation and organization can produce a tangle of opaquely named and interrelated files, leaving analysts with questions such as; Was it “fig_final_200k_40_p05.png”

or “fig_final_40k_20_p05.png” that showed the significant difference? And what version of the code did I use to generate those graphs?

One increasingly popular means of addressing this challenge of organizing the byproducts of data analysis is to perform analyses in computational notebooks. Computational notebooks combine code, visualizations, and text in a single document that can be easily shared (Figure 1.2). While computational notebooks have been around for decades, they have seen rapid adoption in the past decade thanks to the release of free and open source notebook editors such as Jupyter Notebook and RStudio. Today, millions of researchers, students, journalists, and analysts use computational notebooks for a wide range of activities [28].

This new wave of computational notebooks also aims to support collaborative data analysis through the production and sharing of *computational narratives*. Many hope that notebooks will support a wave of open and reproducible research by making it easier to track and share the process of analysis, not just the results. Fernando Perez and Brian Grainger, co-Founders of the Jupyter project being Jupyter Notebook, explain the need for computational narrative in this way [76]:

Computers are good at consuming, producing and processing data. Humans, on the other hand, process the world through narratives. Thus, in order for data, and the computations that process and visualize that data, to be useful for humans, they must be embedded into a narrative — a computational narrative — that tells a story for a particular audience and context.

The research in this chapter asks to what extent computational notebooks are achieving this vision. It deals primarily with Jupyter Notebook, which is used by an estimated 6-8 million people [28], and whose creators are most explicit about their hope to support collaborative data analysis.

3.2 Study 1: Analyzing 1 Million Jupyter Notebooks

To examine the role of narrative in computational notebooks, I scraped and analyzed the 1.23 million publicly available Jupyter Notebooks on GitHub in July 2017. GitHub is a popular website for hosting, managing, and collaboratively editing software source code and in May 2015, GitHub began to natively render Jupyter Notebooks so that anyone viewing a Jupyter Notebook on the site would see the fully rendered notebook rather than its underlying JSON object. This rendering has made Github a popular site for storing and sharing Jupyter notebooks (Figure 3.1) as it enables analysts to easily share their notebooks with others who may not have a copy of the Jupyter Notebook software on their machine. While GitHub users cannot tweak and re-run notebooks on the site, they can at least see the notebook’s contents statically.

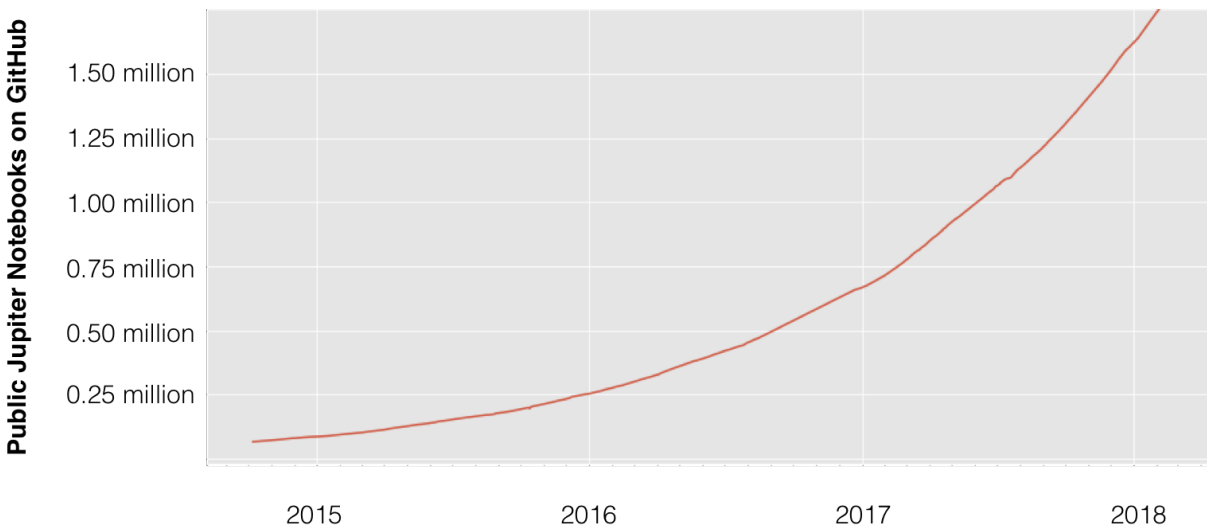


Figure 3.1: The growth of Jupyter Notebooks on Github. There are now well over 1 million Jupyter Notebooks shared publicly on GitHub according to Project Jupyter’s estimates. [71]

3.2.1 Methods

In July 2017 I searched GitHub for all publicly available Jupyter Notebooks that had not been forked (i.e., copied) from another repository (i.e., collection of code). For each notebook, I attempted to download the notebook file, metadata about the repository where it was found, and, if present, the repository's README file. Due to GitHub's rate limiting, these queries took two weeks to complete on a single machine. Of the 1,294,163 notebooks hosted on GitHub at the time, I was able to download notebook files and repository data for 1,227,573 notebooks, or roughly 95% of the public Jupyter Notebooks on GitHub at the time. The majority of the remaining 5% of notebooks I was unable to download were empty or mal-formatted files. Realizing that others may wish to explore this data, I have made the data from this study publicly available through the UC San Diego Library (<https://doi.org/10.6075/J0JW8C39>). To support further analysis I computed a wide variety of features pertaining to each notebook's content and organization.

3.2.2 Results

Users: There were 100,503 GitHub users who had publicly shared a notebook at the time of the study. This was about 0.4% of all GitHub users at the time. The number of notebooks per user followed an exponential distribution, with 24.5% of users hosting only one notebook on GitHub, and 27.4% hosting ten or more. The majority of notebooks (81.4%) belonged to users who had hosted 10 or more.

Repositories: There were 191,402 repositories on GitHub containing at least one Jupyter Notebook. The number of notebooks per repository followed an exponential distribution, with 39.1% of repositories having only one notebook and 14.6% of repositories having ten or more. The majority of notebooks (66.4%) belonged to repositories with ten or more notebooks in them.

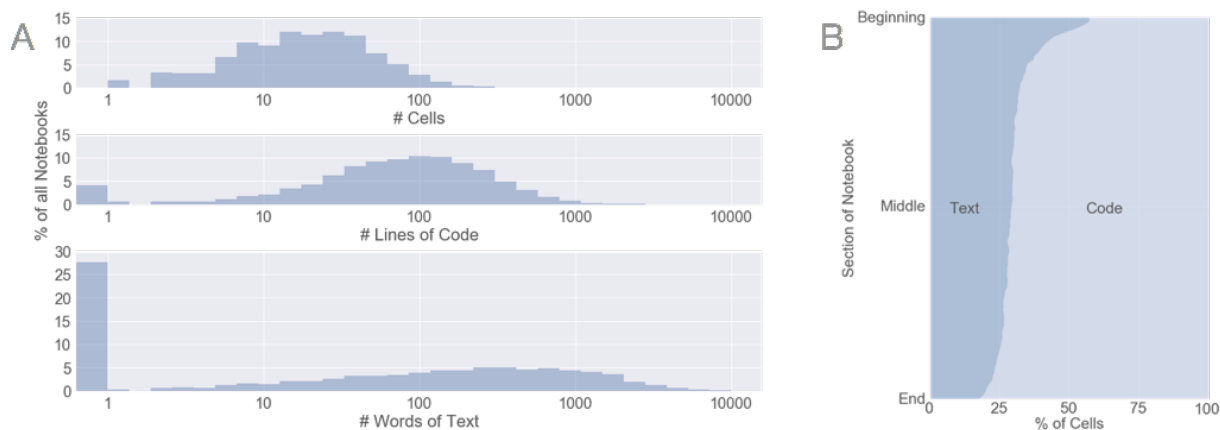


Figure 3.2: A) Notebook length as measured by cells, lines of code, and words of mark-down. While only 2.2% of all notebooks had no code, 27.6% had no text. B) Content type across the average notebook. Cells at the start of the notebook were more likely to be text and cells at the end more likely to be code.

Language & Packages: Jupyter Notebooks can execute code written in over 40 programming languages and users select a primary language when they create each notebook. Of the 85.1% of notebooks with a language specified, the vast majority were written in Python (96.3%), particularly Python 2.7 (52.5%). Notebooks written in R and Julia each accounted for about 1% of all notebooks, with all other languages accounting for less than 1% of the dataset. Of notebooks written in Python, R, or Julia, 89.1% imported external packages or modules. The most commonly imported Python packages were Numpy (67.3% of Python notebooks with imports), Matplotlib (52.1%), and Pandas (42.3%), showing a strong emphasis on data science and visualization.

Notebook Length: Jupyter Notebook cells can be any length and contain either executable code, markdown to be rendered as text, or raw content that should be rendered as is. Most notebook cells in our corpus were either markdown or code (99.8% of cells) rather than raw content. The number of cells per notebook, as well as the amount of text and the number of lines of code per notebook all followed log-normal distributions except that a significant number of notebooks (27.6%) had no text in them but consisted entirely of visualizations or code (Figure 3.2). Only 2.2% of notebooks did not have any

code but were entirely text, for example a table of contents notebook that only had text and links pointing to other notebooks in a collection. Ignoring notebooks without text, the median notebook had 218 words of text, though the longest, at 55,000 words, was longer than *The Great Gatsby*. Disregarding notebooks without code, the median notebook had 85 lines of code, though the longest had over 400,000 lines of code, more than NASA’s primary space shuttle flight software [17].

Organization: Notebooks are extremely flexible with their main organizational element being cells, which can be linearly arranged. Users can provide additional structure by deciding how to split text or code across cells, using functions, classes, and comments to structure code, and using markdown headers and links to structure text. We found that cells at the beginning of notebooks were more likely to be text, but that the majority of later cells were devoted to code (Figure 3.2). Most notebooks used headers to organize text, and comments to organize code (Figure 3.3).








	Feature	% of all Notebooks
Text	Text	72.7 
	Headers	60.2 
	URLs	31.6 
Code	Code	97.8 
	Comments	62.1 
	Functions	37.3 
	Classes	12.3 

Figure 3.3: Notebook organizational features. Most notebooks organized text with headers, and code with comments.

Execution & Outputs: While convention is to run cells linearly from top to bottom of a notebook, cells can be executed in any order. This can be useful when checking if

changes to a prior analytical step impact later computations. Jupyter Notebooks track cell execution order, so I was able to see if notebooks were run linearly or non-linearly. I found that 43.9% of notebooks with computational output had a non-linear execution order. Jupyter supports three types of output: stream (e.g., print statements), executed results (e.g., numerical results), and displayed data (e.g., rich data displays such as graphs and tables). In our corpus, 85.0% of notebooks had output in at least one cell, with 68.5% of notebooks having stream output, 58.1% having an executed result, and 45.5% having displayed data.

Description of Repositories: GitHub repositories provide a number of facilities for describing and documenting projects. These include a short description, longer README files that get rendered on the repository's homepage, and GitHub-hosted project websites. While 58.5% of notebook repositories had a description and 73.0% had a README, only 4.5% had a GitHub-hosted project website. Analyzing the descriptions gives a sense for the topics notebooks analyze and discuss. Excluding common english words such as articles or prepositions and words related to notebooks such as "notebook" or "jupyter", the ten most common words in repository descriptions were learning, project, machine, udacity, course, deep, nanodegree, neural, kaggle, and model, showing an emphasis on machine learning and education.

3.2.3 Discussion

This GitHub corpus is very diverse. Whereas some notebooks contained only a single line of code others were fully interactive textbooks spanning hundreds of pages when printed. While some stood alone, others were part of large collections of notebooks that documented a multi-step analysis. While some were homework submissions, others demonstrated software packages, or documented original research. This diversity discourages generalization, but I highlight a few broad trends related to the use of text and narrative.

First, most notebooks were not rich computational narratives but loose collections of notes and scripts. As discussed in CHAPTER 2, at a bare minimum narratives need to describe a series of events and the connections between them. However, a quarter of all notebooks in our corpus did not have even a single word of explanatory text. Disregarding these, the median notebook still had roughly half as much text as this dissertation’s abstract (Figure 3.2). And while it could be argued that the analyst’s code itself is a description of events and the ordering of cells a tacit connection between them, nearly half of the notebooks in our corpus had a non-linear execution order. From this evidence, there seems to be a general lack of intention to describe or order analytical events in the notebooks studied, other than what is expedient for analysis.

Second, descriptive text was not evenly distributed across notebooks (Figure 3.2). Text was most likely to occur at the very beginning of the notebook, steadily less prevalent as the notebook progressed, and least likely to occur at the very end. This may reflect the use of introductory text to present the goals and organization of the notebook, but not conclusion text to reiterate goals and interpret results. Alternatively, the declining use of text as the notebook progresses may demonstrate that less explanation is needed once the analysis has been set up, or that analysts tire of annotating their notebooks over time.

Third, notebooks in our corpus rarely stood alone. The vast majority were in repositories containing other notebooks, a README file, or both. A single narrative may flow across multiple notebooks, from one for data cleaning into another for profiling and modeling. Moreover README files may provide additional information about the motivations, background, and findings of the analysis. This finding sheds doubt on any claim that notebooks generally serve as self-contained descriptions of an analysis.

Finally, the exploratory and iterative nature of data analysis is reflected in the fact that nearly half (43.9%) of notebooks in our corpus were uploaded to GitHub with a non-linear execution order. This means that analysts went back and re-ran earlier cells,

rather than just linearly writing and executing code. This figure should be considered as a lower-bound as many analysts may have done a clean run of their notebook before sharing it online, obscuring their non-linear analytical process.

These results demonstrate that while many notebooks are certainly used for iterative analysis, few contain lengthy explanations of their contents or much evidence of intentional ordering of scripts and results. Are analyses performed in notebooks being explained in other ways? Or might it be that particular uses of notebooks employ more narrative than others? I began to address these questions by focusing on one particular community of practice: academic data analysis.

3.3 Study 2: Narrative in Academic Notebooks

In this second study, I focused on how notebooks documenting academic data analysis employ narrative. Whereas the first study looked at the structure of notebooks at scale, this study sought to characterize the structure of notebooks in greater detail within a particular community of practice.

I chose to study academic notebooks because, relative to other communities, the collaborative nature of academic research may favor inclusion of text to explain methods and results so others can understand and build on the work. Transparency and replicability of analytical processes is also of increasing importance in the scientific community [67, 74]. To give an idea of the richness of some scientific notebooks, one highlighted by the Jupyter team [48] which supplements a Nature article [15] contains over 2000 lines of code and 7000 words of text, even as the Nature article itself is half that length at 3500 words. I explore whether this example is an outlier, or if most academic notebooks employ narrative to communicate process and results in detail.

3.3.1 Methods

Sampling: With the help of a research assistant I sampled academic computational notebooks by searching GitHub for repositories with both a Jupyter Notebook (.ipynb) file and a README linking to an academic publication. In a pilot analysis of “interesting” academic notebooks [48], we found that many notebooks were in repositories whose README had a URL pointing to a journal, conference, or pre-print publication. While many of these links lead to journal-specific websites, such as nature.com, the most common links pointed to Document Object Identifiers (DOIs) and arXiv pre-prints. To obtain a sample of academic computational notebooks, we searched GitHub for repositories containing Jupyter Notebooks and a README with a DOI or arXiv link. We purposefully sampled the resulting 858 repositories to get 52 from a range of disciplines, looking for keywords such as “chemistry”, “physics”, and “linguistics” in the READMEs. These 52 repositories contained 221 notebooks.

Coding: We iteratively coded all 221 notebooks to develop codes describing how academic notebooks employ text [90]. Specifically, we coded each notebook’s genre, organization and use of text, and the organization and use of code comments. My research assistant and I open coded 50 notebooks to develop initial codes and refined and reapplied these codes until we achieved greater than 60% inter-rater reliability (Cohen’s Kappa), which has traditionally been considered a “substantial” level of agreement [56]. We then divided and separately coded the remaining notebooks. We used a similar process to identify features of the repositories containing academic notebooks, coding for the contents of the repository as well as contents of their README files.

3.3.2 Results

Repository Content and READMEs: In 43 of the 52 repositories, notebooks made up the majority of contents, averaging 81.6% of the repository’s total bytes. In the

nine cases where notebook content was the minority, the majority of repository contents were program files that the notebook imported and called during the analysis. In addition to notebook files, the majority of repositories contained source code in program files such as .py files (40 repositories). Many contained raw data (24 repositories), figures (15), manuscript files (10) and additional documentation (7). Most repository README files described what the repository's code did (33 repositories) and the steps required to setup or install it (33). Many READMEs also described the organization of the repository's files (24) and how to execute the code or notebooks once configured (18). Few discussed analytical reasoning (7) or results (10).

Notebooks: Half of the repositories (26) contained a single notebook. The two repositories with the most notebooks (52 and 26 respectively) were largely repetitive with notebooks that tweaked one or two parameters at the top, and then ran the exact same collection of cells to get a version of a model or result. To prevent these nearly identical notebooks from skewing our data, we removed them from further analysis, leaving 50 repositories with 145 notebooks for further hand-coding (Table 3.1). These 145 notebooks were generally longer than the notebooks from our GitHub corpus in Study 1 with a median length of 31 cells (compared to 18 in Study 1) 102 lines of code (85), and 329 words of explanatory text (218).

Organization and Use of Text: Most notebooks had an introductory text cell (55%) which typically described the analysis to follow but almost none had a concluding text cell (3%). The vast majority of notebooks used headers (86%), and slightly fewer had text aside from the headers to explain the analysis (77%). Of those notebooks with non-header text, 88% used that text to describe analytical steps, but only 34% used it to explain reasoning, and just 38% to discuss results.

Organization and Use of Code Comments: We found 82% of notebooks had code comments. Of these, almost all notebooks (99%) used comments to describe what the code was doing, and half (50%) used comments at some point to control the program

flow by commenting out alternative code. Very few notebooks used comments at any point to explain the analysts’ reasoning (10%) or results (4%).

Table 3.1: Length and content of academic computational notebook by genre. Analysis notebooks employed more text, while Figure notebooks had more code.

	Analysis	Tutorial	Figure	All
# Notebooks	54	41	50	145
# Cells	38	23	17	31
Lines of Code	102	89	162	102
Words of Text	434	213	103	329
Headers	87%	78%	90%	86%
Text	89%	73%	66%	77%
Text Intro	72%	61%	30%	55%
Text Steps	94%	97%	70%	88%
Text Reasoning	46%	33%	15%	34%
Text Results	29%	37%	48%	38%
Comments	89%	66%	88%	82%
Com. Steps	98%	100%	95%	99%
Com. Reason	15%	15%	2%	10%
Com. Results	2%	7%	4%	4%
Com. Flow	58%	37%	50%	50%

Notebook Genre: Through our iterative coding we identified three broad categories of academic notebook; 54 notebooks documented a full analysis, 50 simply replicated figures, and 41 were tutorials for how to use a particular software package. The use of text varied across genre (Table 3.1) with full analysis notebooks typically having more explanatory text than figure replication notebooks and being more likely to have a textual introduction to the notebook. On the other hand, figure replication notebooks tended to use text to discuss results more than analysis notebooks. Note that due to our small sample size and significant variance between notebooks, none of these differences was statistically significant.

3.3.3 Discussion

This closer examination of academic computational notebooks revealed distinct genres, highlighting that even within the smaller community of academic users, computational notebooks serve a variety of purposes. Yet, even in the most verbose genre (i.e., notebooks that replicated the full analysis described in a paper) analytical reasoning and results were discussed in only about a third of notebooks. While a couple notebooks contained richly detailed narratives with several thousand words of text, most were simply collections of scripts with occasional notes describing the code.

Similarly, most repository README files focused on what the repository's files did and how they were organized, but did not discuss reasoning or results. This lack of explanation is not because analyses were straightforward. Even in these publicly shared notebooks, half used code comments to control program flow, demonstrating that versions of the analysis were tried, evaluated, and rejected in favor of other implementations. It seems notebooks were being used for iterative analyses, but not necessarily for constructing rich narratives. Consider that 90 of the 145 notebooks in our sample had less text than their repository's README. This suggests that analysts may be using other media to explain their analyses.

Still, the consistent use of headers, text descriptions of steps, and README files describing repository contents demonstrates that analysts are taking at least some time to annotate and explain their analyses. What audience do analysts consider when they annotate their notebooks? And why do they seem to devote more effort to describing steps but not higher-level motivations or reasoning? I began to address these questions in a third study.

3.4 Study 3: Interviewing Academic Data Analysts

The second study highlighted that, when present, text in academic computational notebooks was more often used to label steps of the analysis than to discuss the reasoning that guided the analysis or to interpret results. Seeking to better understand why these notebooks lacked the rich computational narrative they were designed to support, I interviewed 15 academic data analysts who use computational notebooks on a regular basis.

3.4.1 Methods

Participants: I recruited 15 academic data analysts (4 Female, 11 Male) from eight laboratories at UC San Diego by attending weekly lab meetings and emailing open science listservs at the university. Participants included six postdocs, five PhD students, three staff researchers, and one undergraduate student. Participants researched topics ranging from computational biology and pharmacology to astronomy and engineering science in eight different laboratories. Four of these laboratories had multiple people using computational notebooks as well as extensive infrastructure for running, storing, and sharing notebooks. In the other four labs, our participants were the only ones using computational notebooks. Five of our interviewees had authored at least one notebook from our Study 1 corpus, though we did not specifically recruit them for this reason. None was the author of a notebook included in our Study 2 corpus.

Procedure: I conducted twelve semi-structured interviews, three with pairs of analysts from the same lab and nine with individual analysts. Each interview lasted 30-45 minutes and focused on how each analyst organized, edited, and shared computational notebooks. I grounded each interview by discussing at least one notebook the analyst had been working on recently. Sample questions included:

1. Can you show us a notebook you have been working on recently?

2. Can you explain the analysis in this notebook?
3. What sections or cells have you spent the most time working on?
4. Who else has access to this notebook? Do you plan to share it further?
5. Would you need to make any changes before sharing it further?

With the help of a research assistant I transcribed each interview and iteratively generated an affinity diagram to identify themes across participants.

3.4.2 Results

Notebook Use and Reuse: Our participants used notebooks for a variety of reasons, many of which were educational. Analysts gave lectures from notebooks, assigned homework in notebooks, and used notebooks to train new lab members. While these educational uses warrant further study, we focused our interviews on the use of notebooks for research, where they were most commonly described as playgrounds for experimentation (seven participants), particularly when prototyping and debugging code. While many used notebooks to develop pipelines to automate multi-step analyses (five participants), others felt that notebooks were best for small-to-medium sized tasks and preferred language-specific development environments for larger analyses which they would run repeatedly as new data became available (two participants). Two other participants would not run analyses in the notebook but copied code into the notebook as a record of work performed elsewhere.

Analysts spoke not only of notebooks' initial use during analysis, but also their ongoing reuse. One intended reuse was reconstructing provenance, that is, retracing the exact steps used to generate a result. This provenance was useful for keeping track of what analyses had been tried, even if they led to dead ends, keeping older versions

of figures in case an advisor decided they preferred them to the new one, and helping analysts untangle exactly how they achieved a result.

While analysts can use computational notebooks to track their every step, in many applications (including Jupyter Notebook) this tracking does not happen automatically. As a result, analysts risk losing valuable history of their analysis if they re-run or delete cells which could be a source of frustration. As one participant put it:

I wanted Jupyter to be the tool that tracked what I did, and I'm sad that it's not. — P6

A second reuse of notebooks our participants mentioned was code reuse. Individual analysts might want to reuse snippets of code from their own past notebooks, or they might want to copy code from others' notebooks. This could lead to keeping code in notebooks for easy access and discourage deleting old code:

I don't necessarily want to delete that messy version of the notebook because I might not even remember if I had something in there that, like I, I might want again. — P14

A third potential reuse of notebooks was enabling full replication of a result. To support this reuse notebooks should be clean and annotated enough that another analyst could reasonably re-run the notebook on a different computer. However, as one participant noted, there are numerous barriers to making notebooks both human and machine readable, and preparing notebooks for replication requires more careful construction than preparing them as a loose history of previously written or run code. As one participant wondered:

Should sharing just be, look at their code, 'Oh they did that'? Maybe just that. Maybe it's too much to go all that way [to make it fully replicable]... it's really hard to make it runnable on somebody else's server — P13

A fourth reuse of notebooks after the original analysis was presenting results. In these notebooks analysts downplayed the role of code and added text to describe methods and results. In many cases, they even transferred outputs of the analysis to an entirely different medium (e.g., slides, word processing document) for easier review.

In some cases this re-organization was for a non-technical audience, so analysts tried to draw attention away from the code and toward the conclusions by copying results to another media for sharing. In other cases, when the audience was technical and the desired feedback was technical, analysts would focus on refactoring code in the notebook so it was easy to understand and critique.

While notebooks serve these different purposes, some of our participants felt it is difficult for them to serve more than purpose one at a time.

It's a trade-off between having a very extensive notebook where every step is documented, or only tracking the last evolved state of whatever the question is. — P1

I know I need to make a new version of it that I think will be like, "Github ready". I want my notebook to look like the examples of notebooks that I talked about from my lab mate, but those are so clean that they don't represent my normal notebooks. They're like, presentation notebooks. Like this is perfect. This has descriptions of all the stuff I did and there's no fooling around. — P14

Sharing: Analysts shared their notebooks in ways that reflected differing perspectives on appropriate uses and audiences. For some analysts, notebooks were personal artifacts, best for individual use or select sharing with other technically oriented "insiders":

A notebook is a very personal thing, so even if I would say, "Okay, here [labmate] please look into it", it wouldn't be very helpful because it's very

much reflecting my style and for sure he would do slightly different types of analysis to come to the same conclusions. — P1

I think, that notebook as a medium is sort of useful to, you know, those insiders, the people that will be interested and will, you know, tweak some parameters and then possibly, you know, redo the exact same analysis just on different data. — P2

These analysts were skeptical that collaborators wanted to see their code and instead shared results through mediums such as email, word processing documents, and slides. They would often also share the full notebook just in case their collaborator wanted to see more details, but some felt that reviewing the notebook got in the way of interpreting higher-level findings and providing feedback.

So over time I had to realize that the collaborators... have no computer science background, nor a very strong microbiome background, so I have to report on a very high level... I try to condense what I'm finding within one sentence... And I'm attaching the PDF [version of the notebook] should the person be interested in details, but typically no one is really looking into the methodology I'm applying, so they just trust me. — P1

I've got all this code and I've got my data but this is really not interesting and, you know, my collaborators should not really be worried about that. They should be worried about, like, what do these figures represent and whether this is something that they are expecting, or is this, is it likely there is something wrong with the way that we are processing their data. — P2

In contrast to this “notebooks for insiders” perspective, some analysts felt that notebooks were good for interacting with people who didn't program. However, note-

books for a general audience required careful curation to make them easily interpretable and having programming novices run the notebook could present additional challenges.

I'm trying to explain every detail unless it's like very intuitive... I wanna have chemistry people be able to read the notebook... and if there is a problem, they are going to have to look for a bioinformatician. But I just wanted them to be able to read for now. — P8

Cleaning, Layout & Annotation: Whether for personal or shared use, every analyst we interviewed felt their notebooks had to be cleaned. Analysts described their notebooks as “messy”, containing “ugly code” or “dirty tricks”, and needing “cleaning” or “polishing”. Our interviewees said their notebooks needed cleaning because they were “too lazy” to add annotation, needed to be “at their best” to produce well annotated notebooks, or simply “ran out of time”. As one participant put it:

Mine feels like a mess, mine feels like if somebody else looked at it they wouldn't have any idea what, really what order [I ran the code in], or like why I did things. — P12

Cleaning involved both organizing the notebook and adding textual annotation. Organization included adding tables of contents, sequentially numbering sections within and across notebooks, keeping scripts in individual cells under 100 lines of code, and splitting analyses that were “bulky” or “crowded” into multiple notebooks.

I like to break apart my analyses into what I consider to be notebooks that cover all the work you would do up to a stopping point where a human has to evaluate it. — P7

For me [the biggest challenge is] organization, I don't know if I should do things chronologically or if I should do things by type of data... If I run

something and I run it four different times do I just make a note up here of the four parameters I used, or do I do four different cells where I ran it each time?... at some point [I end up] just getting frustrated and I'll make a new notebook. — P12

Analysts annotated their notebooks both for personal use and easing interpretation by collaborators. Personal documentation was added to prevent “getting lost” in the notebook, to remember what was done previously, and visually differentiate sections of the analysis to aid scanning.

So I try to document what I'm doing, or at least what the tasks are because it's so easy to get lost in all the different specific questions. — P1

When the notebook was to be shared, annotation focused instead on presenting the analysis at a high level, providing background information and interpreting results.

The thing that I usually end up having to put in that's tedious but it's kind of the whole point, is, you know, okay I generated these beautiful visualizations and then what are the conclusions that I drew from them, because, in our role, we're supposed to be the experts who are saying not just, “This is the visualization”, but “If you look at this visualization the conclusion that you should draw”... the interpretation — P6

Social Practice: Several analysts felt that there was not sufficient social expectation or practice to make widespread sharing and detailed annotation of notebooks feasible. This could be due to supervisors not wanting to see the details of an analysis, or lacking formal training in how to document data analyses in notebooks.

Does your PI [Primary Investigator] care about the code or not? And I think that most of the time, from my experience, it's been no. You know

they just want to see the plots... That's, I think, driven by what, you know, what our old kind of standard kind of pathway was... it was a lot of just scripts that you couldn't really port and couldn't really make available. — P15

With like paper lab notebooks in the wet lab you get really heavy training... you should write like your name, your date, kind of the hypothesis that you're doing, a little bit of the intro and then like your materials what your steps were and then some sort of conclusion like and your data... the notebook isn't physically set up that way... But you just do it that way because that's how you're trained to do it. — P12

Publication and Reproducibility: Despite this lack of training or pressure to share notebooks in some labs, many analysts expressed a desire, even an obligation to document their notebooks in such a way as they would be reproducible, that is, that they could be run by another analyst on their own computer. However even these participants mentioned several barriers to making their notebooks truly reproducible. One was deciding when the analysis was ready for publication. Another was receiving pushback from collaborators when preparing to publish a notebook publicly.

So to define the point in time when a publication is finalized is very complicated. Is it when you first submit? Is it when it's in review? Or when you only have to do some format editing? So there is no hard deadline unto which you have to finalize your notebook, and therefore it's very easy to not do it... it's a lot of additional work, and you also have a todo list of more pressing issues — P1

The couple times I've mentioned it [publishing a notebook] I've gotten people, like, they're worried that it like opens them up to more criticism than it's worth for them. — P7

3.5 Discussion

These results highlight the effort involved in organizing, annotating, and sharing computational notebooks. In particular, they highlight the tension between exploration, in which iterative experimentation tends to produce “messy” notebooks, and explanation, in which these notebooks are “cleaned” for a particular purpose (e.g., tracking provenance, code reuse, replication, presentation).

When tailoring notebooks for each of these uses, there is a tradeoff in how the notebook is annotated and organized that reflects the tension between exploration and explanation. Notebooks that track provenance focus on faithfully tracking the exploratory process of data analysis but, given the interactivity of notebooks, analysts seeking to track provenance need to be careful to not overwrite past actions. Alternatively, notebooks for presentation may obscure almost the entire exploratory process of data analysis in an effort to make it easy to review and provide feedback on the results. Notebooks to be shared publicly have to meet an even higher standard of cleanliness that one participant noted removed all the exploratory “sandbox” material. Because of these trade-offs, it is difficult for notebooks to effectively serve more than one purpose at a time.

My interviews also highlight that while notebooks technically enable analysts to wrap computational code and results with explanatory text, they do not necessarily prompt more frequent reflection or annotation. Social practices like presenting at lab meeting and writing papers may still be stronger triggers for these explanatory and sensemaking activities. As P12 noted:

...it’s mostly lab meetings and then actually writing the paper that are the only times, or like the initial planning, that are the only times where you have to sit and be like “Why am I doing this? What am I gonna do? What am I finding? What do I think it means?”

3.6 Conclusion

Computational notebooks address many fundamental challenges with performing, documenting, and sharing data analyses. They support incremental and iterative analyses, enabling users to edit, arrange, and execute small blocks of code in any order. They enable explanation of thought processes by allowing analysts to intersperse code with richly formatted textual explanations. They facilitate sharing by combining code, visualizations, and text in a single document that can be posted online or emailed. Some computational notebooks are truly remarkable in the way they elegantly explain complex analyses [48].

Yet, the three studies in this chapter demonstrate a tension between exploration and explanation that complicates construction and sharing of computational notebooks. The exploratory process of data analysis tends to produce “messy” notebooks with alternative code and duplicate cells. These notebooks need to be cleaned before they can clearly explain the analysis to a particular audience (e.g., the analyst’s future self, a technical colleague, a manager, or the public) for a particular purpose (e.g., tracking provenance, supporting code reuse, enabling replication, presenting results). Cleaning notebooks is often tedious, manual work, and it is difficult to craft notebooks that serve more than one purpose or address more than one audience at a time. Many analysts simply choose to explain and share their analyses using other, more established media, or only provide a link, “for the curious”, to the notebook where they performed the analysis in the first place.

The issues of notebook “cleanliness” and intelligibility resonate with the discussion of refactoring and “technical debt” in software engineering [14]. Rather than calling for the elimination of technical debt, recent work acknowledges its inevitability and suggests better ways to manage it. While some lessons from this literature may apply to data analysis, there are significant differences in the process of iteratively writing

scripts to analyze data and writing robust source code for enterprise applications.

The tension between exploration and explanation demonstrated in this research is also echoed in the literature on design rationale [58, 65]. When in the middle of a design process, designers often do not want to pause to document emergent requirements or reasons for a particular design direction as this can disrupt their train of thought [13]. Moreover, any documentation they create may quickly go out of date or be contradicted by realizations later in the design process. To encourage more capture of design rationale, prior research as suggested providing short term payoffs to design rationale documentation [13]. Taking a similar approach, the next chapter will explore how to providing an immediate benefit to crafting well organized and annotated notebooks.

3.7 Synthesis

This chapter presents three studies of how a tension between exploration and explanation hinders analysts from using computational notebooks to track and share their work. The contributions include:

1. Evidence for a lack of narrative in most computational notebooks either through a lack of explanatory text altogether, or failure to use that text to describing reasoning or results
2. Theory that a tension between exploration and explanation is central to data analysis and hinders tracking and sharing data analyses
3. A open dataset for future research containing over 1 million computational notebooks (<https://doi.org/10.6075/J0JW8C39>)

This chapter, in part, includes portions of material as it appears in *Exploration and Explanation in Computational Notebooks* by Adam Rule, Aurelien Tabard, Jim Hollan in the 2018 proceedings of the ACM international conference on Human Factors in

Computing Systems. The dissertation author was the primary investigator and author of this paper. The research in the chapter was also supported by the work of Regina Cheng and Nathan Hassansadeh who assisted with the second and third studies of this chapter respectively.

4 Aiding Communication of Data Analyses through Flexible Organization and Navigation

This chapter presents the design and testing of Janus, a Jupyter Notebook extension enabling analysts to add history and hierarchy to their notebooks. Through two studies, the first a formative study with novice analysts and the second a multi-week technology probe with expert analysts, I demonstrate that hierarchy in particular enables analysts to flexibly organize and navigate their notebooks in ways that support both the ongoing analysis and later communication of results. These findings support the perspective that flexible and lightweight organization and navigation can help reduce the tension between exploration and explanation, and in turn make complex analyses easier to track and share by enabling analysts to tailor their notebooks for varied uses.

4.1 Introduction

Exploratory data analysis is a process of extracting insights from data and communicating those insights to others [30, 50, 92, 99]. As the scale and scope of data expand, this process has become increasingly collaborative [36, 50]. Consider a jour-

nalist who enlists several colleagues to sort through a collection of leaked documents, or a researcher who sends biological samples across the country as part of a multi-site study; in both cases collaborators need to discuss details of the analysis, not only to coordinate efforts but also because small changes to how data are collected, cleaned, or analyzed can lead to vastly different results. But clearly communicating complex analyses is remarkably difficult, especially when the analysis involves programming.

This dissertation has focused on how analysts use computational notebooks combining code, visualizations, and text to communicate their work, either to their future selves or their collaborators. While computational notebooks were explicitly designed to help analysts craft compelling narratives to share with others [76], as the previous chapter showed, more often than not analysts' notebooks are loose collections of notes and scripts that even they have difficulty understanding at a later time. Rather than share these "messy" notebooks with collaborators, most analysts share simplified results through media such as email, slide decks, or paper printouts that lack a notebook's interactivity, reproducibility, or context.

While computational notebooks help analysts manage their files they have yet to realize their anticipated collaborative potential. The prior chapter argued this gap is in part due to a fundamental tension between the conflicting processes of exploration and explanation in data analysis. Data exploration involves working roughly and quickly to answer a set of evolving questions. Explanation involves a slower process of stepping back to reflect on what the results mean. Further exploration can break or complicate current explanations, and pausing to add explanatory text can put further exploration on hold. Many analysts prioritize data exploration and simply wait until the next deadline to clean and organize their scripts, even when using a notebook. As one analyst in the final study of the last chapter noted:

...it's mostly lab meetings and then actually writing the paper [and] initial planning, that are the only times where you have to sit and be like "Why

am I doing this? What am I gonna do? What am I finding? What do I think it means?”

Even when analysts do take the time to clean and share their notebooks, they report being skeptical that anyone looks at them because the notebooks may seem too detailed, or collaborators may have difficulty tracking the operation of code.

This chapter explores how to design computational notebooks to encourage clearer communication, freer sharing, and deeper engagement with complex data analyses. Through two design workshops with data analysts I explore ways to provide an immediate benefit to notebook annotation and organization activities so analysts have increased incentive to write clear notebooks. Based on these workshops I developed Janus (Figure 4.1), an extension to Jupyter Notebook that enables analysts to add history and hierarchy to their notebooks, giving them more control over how different sections of their notebook are displayed and how prior explorations are tracked. Through two studies of Janus I find that adding hierarchy in particular enables analysts to navigate and manipulate their notebooks in ways that support both ongoing analysis and later communication of process and results. I find that analysts and their collaborators do not passively read notebooks from top to bottom but seek to manipulate, organize, and navigate them in ways that support the task at hand. These findings support the perspective that richer forms of navigation and organization might help analysts use notebooks and other computational media to more clearly communicate complex analyses by providing a lightweight means to tailor the notebook to task at hand.

4.2 Workshops: Identifying Design Opportunities

There are likely many ways to ease the tension between exploring data and explaining process that prevents computational notebooks from being a more effective medium for collaborative data analysis. Past research on managing technical debt has

demonstrated that, in the related domain of software development, developers often know they should be annotating and cleaning their code, but feel they have little time to do so [6]. Moreover they typically rely on manual methods to clean their code rather than relying on tools [66]. Related work on documenting design rationale has suggested providing a short-term benefit to documentation which may appear to only have a long-term payoff [13]. Based on these findings I took the approach of trying to provide an immediate benefit to manual notebook organization or annotation activity, incentivising notebook cleaning rather than automating it or encouraging analysts to clean their notebooks in a separate phase after analysis. Adopting this point of view, I ran two workshops with the help of two research assistants to identify opportunities for the design of computational notebooks to help analysts write clearer notebooks in-the-moment.

4.2.1 Brainstorming Workshop

We recruited seven graduate students from a large public university for the first workshop. Each participant had experience with Jupyter Notebook as well as design thinking [8]. We recruited design-savvy analysts to avoid brainstorming pitfalls such as prematurely focusing on a single solution or discouraging wild ideas. At the start of the 90-minute workshop we briefed participants on common barriers to clear communication in computational notebooks and gave them the following prompt: *How might we make it easier for analysts to annotate, organize, and reflect on analyses in their notebooks? How might we provide an immediate benefit to organization and annotation activities so analysts get into the habit of doing them in-the-moment?* We asked participants to brainstorm ideas individually and then build on these ideas in teams of 2-3 participants. In the final portion of the workshop each team sketched and presented one or two of their most promising ideas to the larger group. Participants generated 78 unique ideas before forming three teams to refine them. The teams' final sketches embodied fourteen design ideas such as

providing templates for common analytical tasks, generating rich tables-of-contents to aid navigation, and providing a second notebook panel as a scratchpad or presentation view. Together these fourteen ideas demonstrated five immediate benefits that notebooks could provide to incentivise in-the-moment annotation or organization. These include:

1. Navigation - Helping navigate long, repetitive, or complex notebooks
2. Version Control - Helping track and compare versions of their code
3. State Inspection- Helping to see the state and evolution of variables and processes
4. Debugging - Helping proactive detection and fixing of code errors
5. Architecting - Helping split code and text across cells, functions, and files

The remainder of this chapter will focus on encouraging analysts to annotate and organize their notebook by my making it easier to navigate their notebooks and track versions of their code. However, the three other benefits are worth exploring, and in some cases are being leveraged in existing computational notebooks.

Participants' desire for assistance with state inspection stemmed from their experience writing numerous small snippets of code just to recall the name, structure, and value of existing variables before acting on them. These snippets of code tend to clutter the notebook as they are often used once and forgotten until they get in the way of some other task. Notebooks such as Iodide and RStudio, for example, provide panels for looking at which variables are currently in the environment, though not necessarily their provenance.

Participants' desire for help with debugging stemmed from similar experience writing small snippets of code to debug a portion of their analysis. To my knowledge no computational notebook explicitly supports debugging as many traditional integrated

development environments do, though the release of JupyterLab with its support for a broader range of development tasks might lend itself to a debugger more than other more focused notebook interfaces

Finally participants' desire for help with architecting stemmed from their experience with the fuzzy boundaries between what should be in a single notebook, multiple notebooks, or a separate script to be imported into multiple notebooks. JupyterLab was in part motivated by the same observation that analysts often worked across multiple notebooks and would often save portions of their notebook code as separate program files and needed better tools for editing these akin to those available for software development.

4.2.2 Paper Prototyping Workshop

Based on the brainstorming workshop, we developed paper prototypes of three notebook extensions that leverage some of the benefits above to address a challenge observed in prior research. These included a rich variable inspector to aid state inspection, cell and notebook-level histories to aid version control, and a two-panel notebook to aid simultaneous navigation of high-level narrative and implementation details. In a second workshop we had six of the seven brainstorming workshop participants use, critique, and revise one or more of these paper prototypes for up to 30 minutes.

Participant feedback on the paper prototypes highlighted a design opportunity at the intersection of cell-level histories and a two-panel notebook. Participants were intrigued by the idea of a rich variable inspector, but wanted assistance not only with viewing but also manipulating variables. One promising step in this direction would be to apply the principles of Wrangler [49], a direct manipulation data cleaning tool, to the notebook environment where users could interact with data through both a graphical-user interface and code. However, more participants saw potential in cell-level histories and a two-panel workspace. They were particularly interested in finding ways to have

more control over (or good defaults for) the naming and layout of different sections of their notebooks or versions of individual cells. For example, perhaps cell versions could derive their names from the values of parameters that changed between them.

In the remainder of this chapter I present Janus, an extension to Jupyter Notebook motivated by these insights and designed to help analysts write clearer notebooks through the addition of history and hierarchy that give them more control over how the notebook is laid out. I then discuss two studies of Janus' impact on how analysts document, share, and build on one another's work.

4.3 Janus: History and Hierarchy for Computational Notebooks

Jupyter Notebooks are linear collections of cells or independent blocks of code or markdown text. Janus (Figure 4.1) is an extension to Jupyter Notebook that enables analysts to view notebook and cell-level histories as well as add hierarchy by hiding groups of cells in named sections.

4.3.1 History: Cell and Notebook Versions

Janus tracks changes to the notebook and presents these as cell and notebook-level histories. Prior observations revealed that analysts often try variants of the same model or graph using slightly different parameters but face a trade-off between either deleting these experiments, or cluttering the notebook with them. Janus supports a different presentation of alternate and historical code by letting users either view and name past versions of individual cells, or see a full history of their notebook's evolution over time, using a slider to move between notebook versions. Janus saves any notebook changes to an SQLite database on the user's machine. When the user names a particular

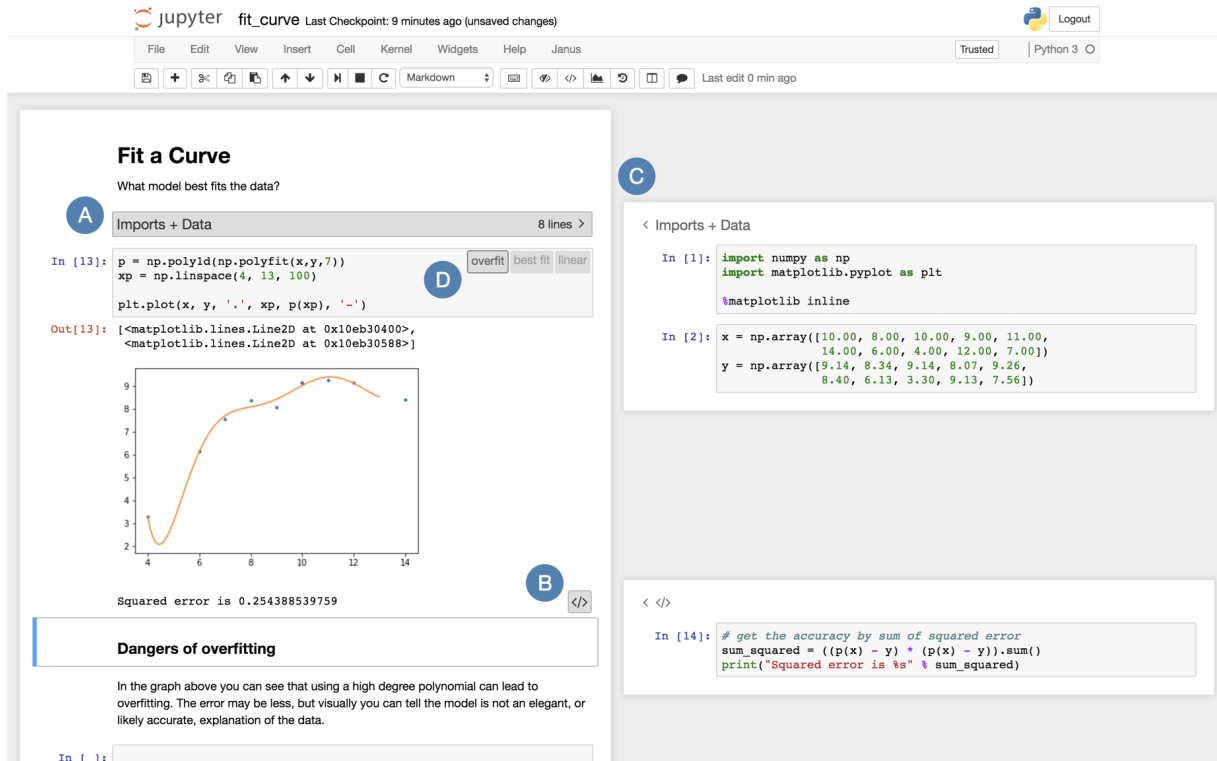


Figure 4.1: The Janus Jupyter Notebook extension. We developed Janus to explore the implications of adding history and hierarchy to computational notebooks. Janus lets users hide groups of cells in named sections (A), or an individual cell’s code or outputs (B). These hidden cells can be selectively shown in a sidebar (C). Janus also lets users track and name different versions of individual cells (D).

version of a cell, that version is extracted from the database and saved in the notebook file itself, enabling targeted sharing. A collaborator with Janus would be able to see named cell versions, but not unnamed versions or the notebook’s full history.

4.3.2 Hierarchy: Hiding Cells, Inputs, and Outputs

Janus also lets analysts hide cells from the main flow of the notebook. When a user hides a cell or several consecutive cells, Janus renders a section header representing the hidden cells in their place. This header can be named and provides basic information about the hidden cells (i.e., how many lines of code they contain). Hovering over the section header shows a miniaturized view of the hidden cells, making it easier to vi-

sually search for particular cells without fully showing them. When the section header is clicked, it reveals a second notebook panel to the right of the main notebook containing the hidden cells. Janus also let's notebook users hide only a cell's source code or outputs, which can also be shown in the sidebar. Originally Janus' sidebar only showed one section of hidden cells at a time, but we modified it to show multiple sections at a time based on feedback from the formative study described in the next section.

Janus stores information about which cells are hidden in metadata attached to each cell. This metadata is saved in the notebook file itself, so other Janus users viewing the notebook would also see a version of the notebook with hidden cells. Notebook users without Janus would simply see the entire notebook in it's normal fully-visible state.

These two features, lightweight history and hierarchy, change how analysts can interact with their notebooks without breaking the underlying metaphor of a linear collection of cells. While simple, these changes have potential to support new ways of performing and sharing data analysis. They may also support a new mental model of notebooks that separates messy implementation details from the final presentation, affording new forms of sharing without requiring major changes to the underlying notebook. Moreover, being able to hide and name groups of cells may make it easier for collaborators, and even the original analyst, to navigate and understand notebooks. We investigated these possibilities through two studies, finding that notebook hierarchy in particular supports a range of navigation and manipulation behaviors that aid both ongoing analysis and later communication.

4.4 Study 1: Formative Study with Novice Analysts

In this first study I wanted to see if notebook users had trouble adapting to a new paradigm of having hidden sections. I also wanted to see if adding this hierarchy en-

abled analysts to more easily understand, navigate, and build on a collaborator's notebook. To address these questions I conducted a formative study with 32 undergraduate data science students under the scenario that they were extending a lab-mate's analysis. Notebooks are often used to share code within labs. For example, a senior researcher may share a notebook with a junior colleague to introduce them to a particular method or line of research. However, even when sharing within a lab, analysts spend substantial time cleaning their notebooks to make them easier for colleagues to follow.

4.4.1 Methods

With the help of a research assistant I recruited 34 students from an introductory undergraduate data science course at a large public university in the United States and gave them a \$25 gift card for participating. Two participants were unable to complete even the first study task in the allotted time and were excluded from our analysis, leaving 32 participants who had completed one or more of the three study tasks.

In this between-subjects study we asked participants to continue an analysis which a fictional lab-mate had begun in a Jupyter Notebook. At the start of the study we gave each participant a 13" laptop with a notebook that compared housing and rental prices in five major cities on the United States' west coast. We then asked participants to perform three tasks that compared housing and rental prices in these cities to those in San Jose, CA. The three tasks were:

1. Get the unique name/id that the data uses to identify San Jose, CA
2. Plot rental and home prices in San Jose over time
3. Correlate changes in rental and homes prices in San Jose with changes in rental and home prices in the other five cities.

Sixteen participants used standard Jupyter Notebook in the control condition

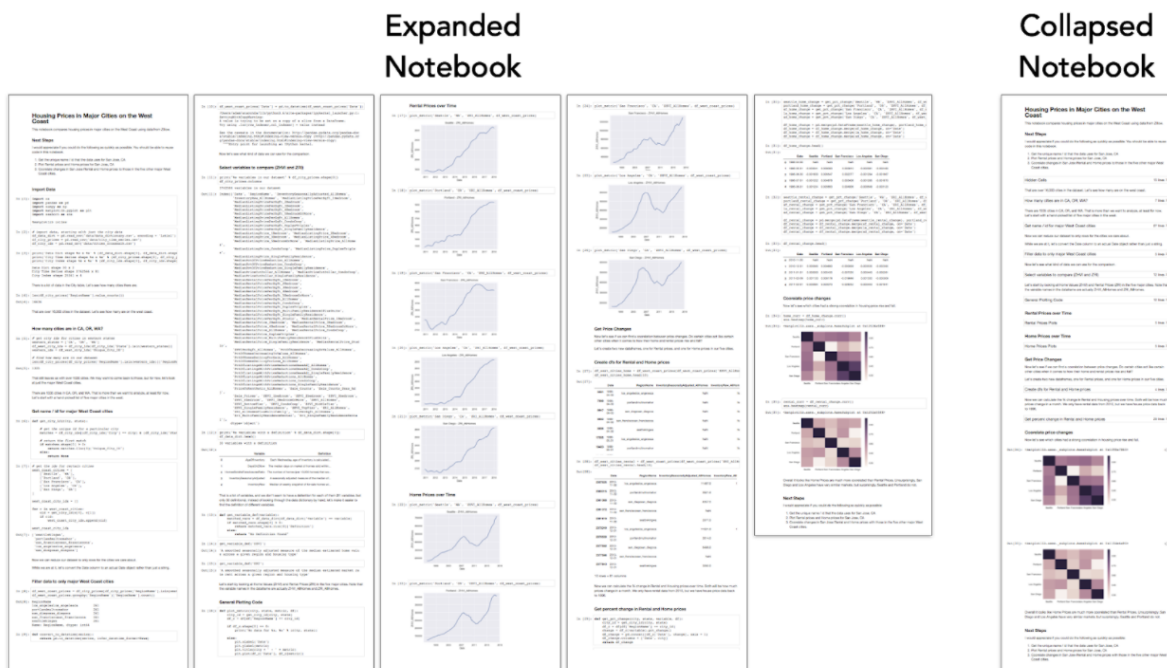


Figure 4.2: The example notebook comparing housing prices used in Study 1. Participants in the baseline condition used the fully expanded notebook on the left whereas participants in the experimental condition used the notebook collapsed with Janus on the right. While the collapsed notebook was initially much shorter than the expanded one, all hidden cells could still be accessed by clicking a header for each section of hidden cells.

and sixteen used Jupyter Notebook extended with Janus in the experimental condition. Participants in the experimental condition were first shown Janus’ features in an example notebook, and then worked with a collapsed version of the study notebook in a version of Jupyter Notebook extended with Janus. Hiding most of the code in named sections made this notebook about 80% shorter (Figure 4.2) than the control notebook, though all hidden cells were accessible by clicking on a labeled section header, which would open the cells in Janus’ sidebar. In the control condition these sections were labeled with inline markdown headers. To encourage extensive navigation and engagement with the notebook, the tasks were designed to be completed by reusing code already in the notebook. We did not test Janus’ history features in this study as I felt

they would be more beneficial for analyses spanning days or weeks than those lasting an hour or less.

We gave participants 30 minutes to complete as many of the three tasks as they could. At the end of the study we interviewed each participant about their experience using their version of Jupyter Notebook, and had them fill out a post-study questionnaire testing their notebook comprehension.

4.4.2 Measures

Proficiency: Before the study began, we asked participants to rate how proficient they were analyzing data using Python and working with Jupyter Notebook on 7-point likert scales. We also asked participants how many years programming experience they had and their major. Due to their systematic differences in self-rated data analysis proficiency, we divided participants into computing majors (e.g., Computer Science, Computer Engineering, Electrical and Computer Engineering), and non-computing majors (e.g., Cognitive Science, Bioengineering, Chemical Engineering) and balanced major type across conditions.

Navigation and Comprehension: At the end of the study we asked participants to rate how easy it was to navigate and understand the notebook on 7-point likert scales. We also tested their comprehension of the notebook using a 7 question quiz which asked about methods and high-level results. For example, one question asked: *Which cities did your colleague originally compare?*

Productivity: We measured how productive participants were by tracking how many of the three tasks they completed and how long it took them to complete each task.

4.4.3 Results

We compared programming proficiency between computing and non-computing majors using t-tests. For all other measures we use two-way ANOVAs to estimate differences associated with using Janus, the participant's major, or the interaction of these factors. We report 95% confidence intervals for effects with a p-value less than 0.10, though for this formative study we were more concerned with users' perceptions and interactions with Janus than statistical significance.

Proficiency: Sixteen participants were pursuing a computing degree and sixteen a non-computing degree. While there was a range of programming experience within each group, there were significant differences between them in self-reported proficiency using Python for data analysis (4.1 vs 3.1 out of 7, $p < 0.01$), self-rated proficiency with Jupyter Notebooks (4.6 vs 3.6 out of 7, $p < 0.05$), and years of programming experience (3.3 vs 2.3 years, $p < 0.01$). These differences are robust enough to suggest that computing and non-computing majors might use Jupyter differently based on different levels of programming experience.

Navigation and Comprehension: There was no significant difference in self-rated ease of navigating the notebook, self-rated ease of understanding the notebook, or scores on the poststudy comprehension quiz associated with using Janus, pursuing a computing degree, or the interaction of these factors ($p > 0.1$ in all cases).

Productivity: There was no significant difference in the number of tasks participants completed associated with using Janus, pursuing a computing degree, or the interaction of these factors ($p > 0.1$ in all cases). However, there were differences in how long participants took to complete each task (Figure 4.3(a)). A two-way ANOVA revealed that non-computing majors using Janus tended to take longer reviewing their collaborator's notebooks before starting the tasks than did other participants ($p = 0.09$, 95% CI [-69, 986] seconds longer). Note in Figure Figure 4.3(a) that this difference seems largely driven by three outliers. We return to this explanation in the discussion.

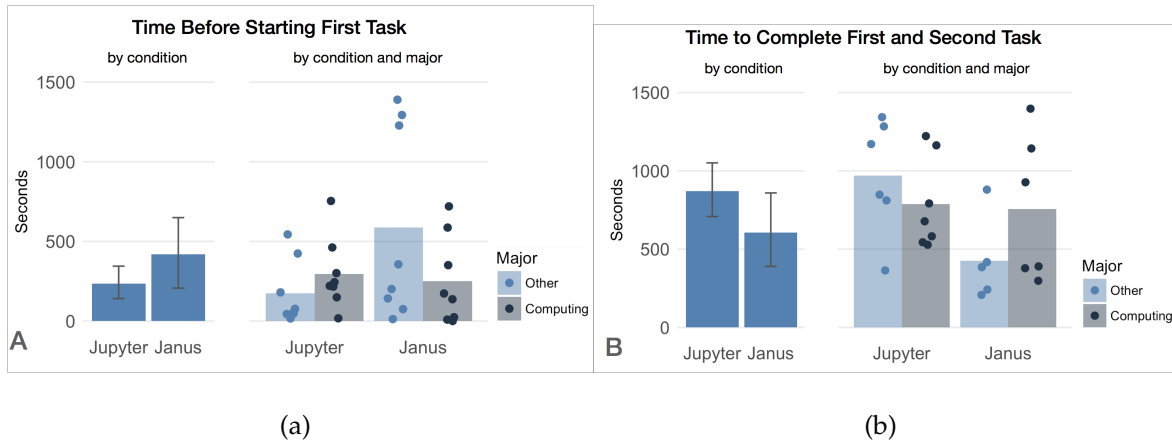


Figure 4.3: Task performance with Janus. (a) Non-computing majors using Janus tended to spend significantly more time reviewing the notebook before starting the first task, though this was mostly due to three outliers. (b) However, Janus users tended to take less time to complete Tasks 1 and 2, particularly if they were non-computing majors. Note how seven of the eight fastest times on Tasks 1 and 2 were from Janus users.

Despite potentially taking longer to review the notebook, there was also a reverse trend for non-computing majors using Janus to surprisingly take less time to complete the first two tasks (Figure 4.3(b)). We compare time spent on the first two tasks as 24 of 32 participants were able to complete Tasks 1 and 2 in the allotted time but only 13 were able to complete all three, limiting our ability to make a comparison. A two-way ANOVA represented this difference as a combination of a weak main effect for all Janus users to take less time to complete Tasks 1 and 2 ($p = 0.09$, 95% CI [92, 996] fewer seconds), but also a weak interaction effect for computing majors using Janus to take longer on Tasks 1 and 2 than non-computing majors using Janus ($p = 0.10$, 95% CI [-102, 1126] more seconds). This combination of effects nullified any time gain from using Janus for computing majors, but preserved it for non-computing majors. Analyzing Tasks 1 and 2 separately revealed a similar trend with non-computing majors using Janus taking less time.

Qualitative Feedback: In the baseline condition with Jupyter Notebook, seven of sixteen participants mentioned the length of the notebook as a problem when asked

about ease of navigation. Some even proposed new ways of making the notebook easier to navigate:

I wish there was some type of compressing tool. I did not have to see all these plots to be able to understand the next step. So if there is anything that hides the plots and expands it only when I want to see it. I feel tools like that would make it easier to navigate. — P32

In terms of difficulty, definitely the length. It would be really nice if there was some sort of breaking, [or] pages. You know? Each of these sections, if you could while you're making the notebook be like, "new page", but still be in the same notebook. Because you know, a real notebook has pages. That's an ongoing gripe for me with Jupyter Notebooks. — P21

Of the sixteen participants who used Janus, ten spoke favorably of the extension.

I like the hide cells thing. I never really explored that. If that's not already a thing in Jupyter, I hope it's a thing... When I'm trying to find a certain portion of the notebook it would be easier to just hide the portions I don't need currently. — P29

I liked how it was, how the lines were condensed. And [how] I could have it on a separate side, [how] it was divided, or split screen. I get to see [the code] as I'm looking at the whole file... So I can see what steps go after and I can get the summary of it. — P23

However, five Janus of the sixteen users felt the hierarchy hindered navigation at times or at least took some getting used to:

It's difficult... In a regular notebook you can just scroll up and down and see the code but here if you want to see you have to actually click it, so it's not as efficient as it used to be. — P11

For the most part I'd say it worked fine. It's just small things personally that I'd say that I wasn't used to. I definitely enjoyed the hide output [feature]. — P25

Participants in the Janus condition also discussed various ways to improve the extension. Four mentioned wanting the sidebar to be less “cramped” either by using a larger screen or showing hidden cells inline with the main notebook. Three wanted the sidebar to follow them as they scrolled so they could view code from the top of the notebook while making edits to a cell at the bottom of the notebook. Finally, eight participants mentioned using browser search (e.g., CMD - F) to navigate the notebook and wanting better support for code search in Janus.

4.4.4 Discussion

Many of the sixteen participants who used Janus quickly adapted to and liked being able to hide and navigate between sections of the notebook. On the other hand, many of the participants who used the current version of Jupyter Notebook complained that the notebook was long and hard to navigate. Together these results suggest there are clear benefits to adding hierarchy to computational notebooks. However, there may be times when analysts want to see the entire notebook at once, or to rapidly scroll between and compare code in multiple sections. Based on this feedback I redesigned Janus to allow users to look at multiple sections of hidden cells in the sidebar at a time.

Remarkably, using Janus also seemed to help non-computing majors take less time to find and reuse code in the notebook, though it may have also encouraged them to take longer reviewing the notebook in the first place. Examining how participants edited their notebooks gives some clues as to why this was the case. In short, using Janus seems to have encouraged participants to adopt different programming strategies than standard Jupyter users.

First, as shown in Figure 4.3(a), the trend for non-computing majors using Janus to take longer to review the notebook is largely due to three outliers. These participants all wanted to see the entire notebook unfolded before they began their analysis and took time to show and re-run each section of hidden cells in the main notebook, retracing their colleagues' entire analysis before beginning their own. Since Janus lacked a "Show All Hidden Cells" button, this was a tedious process. Alternatively, it may be that by breaking long analyses into sections, Janus makes the analysis seem more approachable and encourages users to retrace each section of the analysis rather than just gloss over them.

As for Janus enabling users, particularly non-computing majors, to take less time to complete their tasks, consider the bottom third of Figure 4.3(b) where seven of the eight completion times under 500 seconds were from Janus users. Reviewing screen recordings revealed that these faster times may be due to adopting a strategy of making small edits and re-running cells in the middle of the notebook rather than copying and pasting code to the end of the notebook. Five of these eight participants with the fastest times adopted this in-place editing strategy whereas only three of the remaining sixteen participants who completed Tasks 1 and 2 did so. This effect was particularly pronounced for non-computing majors because many computing majors were still able to complete Tasks 1 and 2 relatively quickly in the baseline Jupyter condition by falling back on well-established strategies for navigating long files such as using CMD-F to search for specific function names. The four Janus users at the top third of Figure 5 who took more than 900 seconds to complete Tasks 1 and 2 all took a third approach: writing entirely new code to complete the tasks rather than reuse code already in the notebook as was intended.

Using Janus then seemed to have a polarizing effect on programming style, either encouraging participants to edit and re-run code in place or to ignore previous code and start their analysis from scratch. In the baseline Jupyter condition more participants

took the strategy of reusing code already in the notebook by copying and pasting it to the end of the notebook where they could modify it without overwriting previous analytical steps. Future work will need to explore why adding a simple hierarchy may influence programming strategy. It may be that viewing code in smaller chunks gave participants confidence that they understood the code and the consequences of modifying it, so they were more likely to edit code in place than their peers in the baseline Jupyter condition who saw the entire analysis at once. In other cases, hiding sections of cells may also have encouraged participants to overlook aspects of prior work and thus recreate it. Either way, using Janus helped novice analysts skim and access sections of a collaborators' notebook without having to constantly scroll through sections unrelated to their current task.

4.5 Study 2: Technology Probe with Expert Analysts

The first study in this chapter demonstrated that adding hierarchy can help novice analysts navigate and extend a collaborator's computational notebook. However, extending an existing analysis is just one form of collaboration, and analysts might use history and hierarchy differently in their own notebooks, especially if they have substantial experience with notebooks. To better understand how analysts use computational notebooks to support collaborative data analysis, and how history and hierarchy might support these uses, I had three expert analysts use Janus for two to three weeks during their everyday analyses. I had three goals for this technology probe [47]: understanding analysts needs and desires in a real-world setting, field-testing Janus, and inspiring analysts (and ourselves) to think about new ways of interacting with computational notebooks.

4.5.1 Methods

With the help of a research assistant I recruited three analysts (2 PhD students and 1 undergraduate student) via email from three different laboratories at a large public university in the United States. All three had extensive experience with Jupyter Notebook, using it to model antibiotic resistance, neural spiking behavior, and the structure of proteins.

We demonstrated Janus to each participant at the start of the study using an example notebook and then asked each to clean the housing price comparison notebook from Study 1 (Figure 4.2) as if they were going to share it with a colleague. Afterwards we asked them how they might have cleaned the notebook differently for themselves or a manager. After the cleaning task, each participant installed Janus on their primary work computer and used it for a period of 2-4 weeks during their everyday analyses. Participants could use a commenting tool built into Janus to leave in-situ feedback if they encountered a bug or an interesting use case. We interviewed each participant about their experience with Janus after each week of use.

Our primary data came from weekly interviews. Sample questions included:

1. Can you open a notebook where you used Janus this week? How did you use, or try to use, Janus in this notebook?
2. Were there specific moments where you wanted Janus to help you do something, that it could not?
3. Did you show a notebook organized with Janus to anyone this week? If so, how did they respond?

We transcribed each interview and iteratively extracted and grouped key quotes to find themes.

4.5.2 Results

I organize results by the three goals of technology probes: Social, Technological, and Design.

Social: Communicating with Self and Others.

Whether they were cleaning our example notebook or their own notebooks, our participants expressed the feeling that they were their own closest collaborators, and that other analysts, especially their advisors (sometimes referred to as Principle Investigators or PIs), were not interested in their code, though a fellow student or post-doc might want to see portions of their code from time to time.

Oftentimes I am the colleague in the sense that I'm looking at [the notebook] a week after and I have no idea what I was thinking at the time. — P1

My PI is not a coding person... he wants to know what's going on and he wants to see the graphs. — P3

From most of my conversations with people that's pretty consistent across PIs. They don't want to see the code, they just want to see the high level idea. — P1

As a result, our participants primarily considered their own needs when cleaning notebooks and used other media to share results, though often reluctantly. Our participants focused on the difficulty of using notebooks to present results during weekly lab meetings.

When I've presented versions of this notebook in our weekly meetings, I'm always scrolling through and it takes a while to scroll through something and I might think, "Oh I want to go back to this plot above", and

I scroll scroll, scroll, scroll and people are getting distracted by various plots. — P1

My notebooks are slightly markdown lite. I don't have much deep analysis of what I'm doing. If someone else walked into this notebook they'd probably have a pretty hard time figuring out exactly what's going on and why this graph is interesting. — P2

Its all slide decks now. We've tried... I haven't had much success using notebooks as a presentation tool. I've just kind of given up on that. — P3

However, using Janus opened up new possibilities, both to aid an ongoing analysis, and to aid communication with collaborators. When working on an analysis, Janus helped our participants keep old code without it getting in the way of the current task. Keeping this code could even help them resume the analysis later:

The reason I got excited about [Janus] is we have these huge notebooks all the time, and you don't want delete stuff that you did cause you want to come back and try to remember what you did later on. But, when you're working on it, it's just like, after a while you have to start scrolling through all this stuff and... it just gets tedious — P3

What I would have normally have done was cut it out, just delete that cell completely, and just think in my head, "Oh that line was super easy to type, if I want to see that selection again, I'll just type that same line again." But... I'll forget to do that later.. [with Janus] instead of having to retype it I could just kind of see "Oh, what section of the data-frame was I looking at before?" That will kind of jog my memory. — P2

Another participant mentioned that using Janus helped him make sure he was editing the correct plot. Whereas before he would often mistake two similar plots, now he could hide all but the current plot he was working on. In his words:

The feature of hiding the output is, I love that feature and obviously that has become a central piece of my notebook. — P1

Our participants also felt that using Janus helped them focus their own and collaborators' attention on the most important parts of the analysis and encouraged them to think about expected results by initially hiding all results so they didn't absentmindedly scan for results without reading explanatory text first.

I can just kind of quickly scroll through [the notebook] and know that every cell that is still left is a cell that I wanted to show for some reason — P2

I feel like when you have these notebooks with all these figures its really tempting to just scroll down until the next figure and just look at it and scroll down... its a nice experience [with hidden outputs in Janus] for me or a collaborator to say, "Okay, what is it that I'm looking for, what do I expect to see?" — P1

Beyond fostering deeper engagement with the analysis, our participants felt that using Janus also made it easier to reuse their notebook in multiple contexts without much additional work. This contrasted with needing to make slide decks to summarize results for lab meetings or spending significant time cleaning notebooks before sharing them publicly online.

I'm actually going to be presenting this notebook today to the group that I'm working with and I've been very excited about this feature because... when I'll be presenting it on the projector it gives me a lot more control over what people are seeing. — P1

I can just throw [incomplete/buggy code] to the side, that way I don't need to have a development notebook and a presentation notebook. I can

kind of just have it all mixed into one and throw stuff to the side when I don't want people to see it. — P2

Surprisingly, none of our participants used the cell or notebook history tools. It may be that these tools are most useful for longer-term analyses, that they do not present a useful representation of past analytical history, or were poorly named. Two participants thought the cell history feature was most useful not for passively tracking history but for purposefully saving interesting model or graph variants, especially if those models or graphs took a long time to create.

I think [the cell history] could be named a little better and maybe that would help people be encouraged to use it... I think that it's a really handy tool. — P2

Technology: Sidebar Unnecessary Technical Overhead.

Janus was robust enough to support daily use, though users did encounter a few bugs which we corrected during the study. As in Study 1, one of our participants mentioned that the sidebar felt small. He rarely viewed hidden cells there, choosing instead to hide and show them in the main flow of his notebooks. While we expected more experienced analysts would use external monitors, giving more space for the sidebar, our participants still frequently edited notebooks on their smaller laptop screens as they moved around campus to attend different meetings. This same participant also mentioned that the animations used to open and close the sidebar were distracting whereas those used when hiding and showing cells in the main notebook felt more natural.

While we intended for the main notebook to provide an overview of the analysis, and the sidebar to provide details-on-demand, analysts' repeated critique suggests that hiding and showing cells in-line with the rest of the notebook may better match their mental models and be less distracting. Removing the sidebar would also have made

Janus easier to implement. Jupyter Notebook updates its underlying JSON data structure based on cells rendered in the main notebook DOM element. Since the sidebar cells were rendered outside this element, much of Janus' code was devoted to linking and synchronizing sidebar cells with duplicate cells hidden in the main notebook's DOM.

Design: Support for Navigation and Manipulation.

Participants were generally happy with Janus and mainly suggested minor tweaks rather than complete re-imaginings of the notebook metaphor. However, our interviews, as well as observations from Study 1, highlight several use cases that computational notebooks could better support. Many of these use cases are similar to those explored in the active reading [61, 77, 94] or collaborative visual analytics literatures [36], though they reflect the unique needs of analysts and their collaborators as they read, review, and reflect on analytical code and commentary in notebooks.

Richer Visualization of Hidden Cells: In the current version of Janus, sections of hidden cells are summarized by a user-defined labels and the number of lines of code they contain. When users hover over the hidden section, they see a small preview of the hidden cells. While analysts sometimes know the exact cell they are looking for (e.g., where was that one with the plotting code?) at other times they are looking for all cells where a particular data object was manipulated and trying to understand what happened to that object as a result of the code. Hidden section markers could provide information about the objects created, modified, or deleted in them and provide compact visualizations of those operations.

Inline Search and Notebook Summarization: Analysts often looked for all instances of a variable or method, or the history of how an object was created, used, and modified. This process could be accelerated by letting notebook users search for a data object, and then hiding all cells or lines of code except those pertaining to the searched object.

Showing Notebooks Sections Next to Each Other: Analysts often compared disparate

sections of the same notebook, for example when copying code or comparing results of two different steps. Currently this requires precise and repetitive scrolling. JupyterLab, the next version of Jupyter Notebook addresses this issue by letting users place two versions of the same notebook side by side. Alternatively, notebooks could let users collapse intermediate sections of a notebook (as in LiquidText [94]) to show two disparate sections one right after the other.

Saving Views: Analysts wanted to view different parts of their notebooks at different times, and to save configurations to show collaborators. Similar to how some collaborative visual analytics systems enable users to save configurations of the visualization [39, 101], notebooks could enable users to save configurations of hidden cells so they can prepare and easily revert to specific configurations.

4.6 Discussion

The findings from these two studies with Janus suggest it would be fruitful to consider how computational notebooks can support richer forms of navigation and manipulation. As opposed to the linear and often passive process of reading a novel, analysts and their collaborators want to actively skim, cross-reference, bookmark, and jump around computational notebooks as they do other physical and digital documents. However, in their current form, computational notebooks require extensive scrolling to navigate and are difficult to skim. While participants had some workarounds (e.g., using third-party extensions to render a table of contents at the top of their notebook, or using an ALL-CAPS header to mark where they left-off), notebooks could do more to support richer navigation and manipulation. The primary benefit from using Janus seemed to be providing a lightweight means to dynamically reorganize and navigate notebooks in ways that supported the task at hand.

Some computational notebooks such as Observable, Iodide, and JupyterLab are

experimenting with allowing users to hide all code cells at once, leaving just textual description and results. While this may provide a quick and easy way to “clean” notebooks for certain uses, many use cases are not supported. For example, participants in our technology probe selectively hid all but a few code cells to help them focus during a development task, hid only cells with experimental or broken code during a notebook code review with a colleague, and then hid all code cells when presenting work in their weekly lab meeting. Future work might explore analysts’ varied motivations and strategies when cleaning their notebooks to support different activities.

So far Janus supports a limited form of navigation, mainly helping analysts and their collaborators jump around long notebooks by hiding sections they do not need to see at the moment. However, as Tashman and Edwards note in related literature on active reading, richer forms of interaction can also include annotation, content extraction, and dynamic layout of documents [94]. As with systems such as XLibris [77] and Papiercraft [61], computational notebooks have potential to support these activities in ways that paper cannot, for example automatically searching for all portions of the notebook that depend on a highlighted block of code.

Support for navigation and manipulation can also extend beyond a single document. As Chen et al. note, there are several levels of interaction to support beyond an individual document including multi-document workspaces and support for multi-session reading [11]. In addition to supporting interactions with a single notebook, computational notebook software could also support multi-document sorting, layout, and information extraction as well as saving and restoring multi-document workspaces to support reading across sessions and even between collaborators.

4.7 Conclusion

The rapid adoption of computational notebooks demonstrates their usefulness for analysis and their potential to encourage open science with increased sharing of data and analyses. Key to fulfilling this potential is enabling and encouraging clear communication about the goals, methods, and results of analyses. Although computational notebooks are enticing vehicles for open science, much work remains to enable them to realize their full potential as a medium for effective communication and sharing of interactive and reproducible analyses.

In this chapter I explore how computational notebooks might be designed to encourage clearer communication by developing and testing Janus, an extension to Jupyter Notebook that adds history and hierarchy to the notebook. Through a formative study with 32 undergraduate data science students I demonstrate that adding hierarchy to computational notebooks can help novice analysts more easily navigate and extend an analysis in an existing notebook. In a second study, a multi-week technology probe with three expert analysts, I found that although analysts infrequently used Janus' notebook history tools, the ability to hide content and add hierarchy to notebooks aided both the ongoing analysis and later communication by supporting varied navigation and manipulation behaviors.

Future research should explore how computational notebooks and other forms of computational media can support richer forms of navigation and manipulation to help analysts and their collaborators deeply engage with complex data analyses.

4.8 Synthesis

This chapter presents the design and testing of Janus, a Jupyter Notebook extension enabling analysts to add history and hierarchy to their notebooks. Its main

contributions are:

1. Evidence suggesting that adding hierarchy to notebooks enables less experienced programmers to extend a colleague's analysis more quickly
2. Evidence that adding hierarchy to analysts notebooks supports both the ongoing analysis and later communication of experienced analysts
3. Theoretical perspective that providing a lightweight means to dynamically reorganize and navigate notebooks reduces the tension between exploration and explanation by making it easier to tailor notebooks to the task at hand
4. A prototype Jupyter Notebook cleaning extension, Janus

This chapter, in part, includes portions of material from *Aiding Communication of Complex Data Analyses in Computational Notebooks* by Adam Rule, Ian Drosos, Aurelien Tabard, and Jim Hollan, as it was submitted to the 2018 ACM international conference on Computer Supported Cooperative Work. The dissertation author was the primary investigator and author of this paper. Esan Hassanzedah and Evan Schmitz assisted with running the design workshops and Ian Drosos assisted with running the formative study and technology probe.

5 ActiveNotes: Interactive Notes for Clinicians

Whereas prior chapters explored the design and use of computational notebooks for programming-based data analysis, this chapter investigates a broader paradigm of using computational notebooks to support data-driven activities without using a general-purpose programming language. Specifically, this chapter presents an exploratory analysis of how clinicians might use free-text to place medication orders from within a clinical note. This work takes the approach of using a domain-specific language to enable clinicians to perform actions within a note.

5.1 Introduction

The previous two chapters explored how the design of computational notebooks might encourage analysts to document and share their work so others can understand, reproduce, and extend it. Computational notebooks such as Jupyter Notebook and R Notebooks, while powerful, require users to be familiar with a general-purpose programming language such as Julia, Python, or R. Despite the millions of data analysts and end-user programmers who might use computational notebooks in this way [83], there are even more professionals performing data-driven activities who have little or no programming experience. These include nurses and physicians, city planners,

lawyers, engineers, and educators. How might computational notebooks support their work? This chapter begins to address this question by examining the specific scenario of how computational notebooks might help clinicians place medication orders from within a clinical note. This work demonstrates how computational notebooks can support data-driven work for a broader range of people by using a domain-specific language to specify or interact with data.

5.2 Background: Fragmented Medical Records

Medicine has experienced an influx of digital data in recent years due in large part to government incentives to adopt Electronic Health Records (EHRs) [102]. Radiology images once stored on film now live on hard drives and medication orders once written on paper are now routinely placed via computers. One consequence of this mass digitization has been a rapid rise in the amount of information clinicians and are expected to manage. While the field of biomedical informatics is dedicated to organizing data so clinicians and computers can process and act on it [86], recent studies suggest that front-line clinicians still lack adequate tools to manage, make sense of, and act on the growing flood of patient data [24, 73].

This is partially due to the current separation of structured and unstructured data in Electronic Health Records (EHRs). With paper records clinicians often wrote short notes summarizing a patient visit such as “ ‘feeling tired’, Depressed, Librium (30) (5mg)”. These concisely conveyed a patient’s symptoms, the physician’s diagnosis, and treatment plan [35]. Today, many EHRs organize patient data by information source (e.g., labs, radiology, medications, notes) rather than by problem (e.g., diabetes, hypertension, glaucoma) [103] or encounter. Under this scheme symptoms (e.g., feeling tired) are displayed on a separate screen from diagnosis or treatment. This fragmentation makes it difficult for clinicians to make inferences across data types without jump-

ing around the record to extract information from disparate sections [110], a process some refer to as “chart biopsy” [41].

Clinicians also struggle to make sense of and act on patient data due to “note bloat” [34, 42]. Clinical notes were originally intended to concisely summarize a patient’s care. While clinical notes still serve this purpose — and are where clinicians spend the majority of their EHR time [9] — most are bloated with irreverent and duplicate information included to fulfill billing and legal requirements. In many notes the majority of text was auto-populated from a template, or copy-pasted from another note rather than reflecting a concise and up-to-date summary of the clinicians’ findings, reasoning, and plan. Consequently clinicians spend much of their time scrolling through long, poorly formatted notes looking for a sentence or two of interest [110].

Clinicians also struggle to make sense of and act on patient data due to the way current EHRs separate documentation from action. While clinicians typically document their treatment plan in a clinical note, they need to go to an entirely different part of the EHR to actually place the orders to carry out that plan. If they want to reference a lab value while writing a note, they either need to go look up that value on a separate screen, or use a “smart phrase” to auto-populate that value into the note, often along with a host of boilerplate text and irrelevant values. This separation of data entry, access, and action means physicians are often jumping around the record as they work, documenting information in multiple places, or delaying documentation. Hospitalist physicians often place orders for their patient’s care immediately after morning rounds, but wait hours or days before writing the progress note explaining why they chose that line of care [10, 97], potentially delaying future care.

Poor EHR usability is a major cause of physician burnout [31] and dissatisfaction [24] and contributes to many physicians spending more time in front of their computers than in front of their patients [93]. The major EHR vendors have largely addressed these issues with EHR usability by making it easier to pull structured information into

clinical notes, or letting clinicians split their screen to show two portions of the same record side by side [1, 3]. Improvements to EHR usability can allow clinicians to spend less time on the computer and more with patients [31], but prior research has largely focused on auto-populating critical information into clinical notes or helping clinicians copy information across the record [32], though in some cases with intelligent linking and alerting of clinicians when critical values change [105].

However, little attention has been paid to the opposite conversion of creating structured data from note text in real-time or of letting clinicians take actions from within a note [16], though this concept is intimately connected to previous work on information retrieval. For example in [105], one participant remarked after using a feature to search for and copy patient information into a note that they would “like to place orders for medications and tests” from within the note. This note-centric vision of EHRs is in many ways motivated by the same principles that led to the rise of computational notebooks: a desire to unify the many byproducts of a data-driven activity and a need to more closely tie explanations of that activity to the artifacts used to carry it out.

This chapter explores how the divide between documentation and action in EHRs might be reduced by enabling clinicians to mix the two in their clinical notes. The ultimate vision of this line work is to develop an EHR paradigm using interactive progress notes to unify entry, access, and retrieval of structured and unstructured patient information. However this chapter focuses on a much smaller part of this note-centric vision: enabling clinicians to place medication orders from within a clinical note using free-text. The particular research questions I address in this chapter are:

1. What shorthand do clinicians use when placing free-text medication orders?
2. What functionality do clinicians expect from computerized free-text order entry?
3. Do clinicians see free-text order entry as a useful addition to their clinical notes?

Beyond addressing these specific questions, this chapter also illustrates a broader vision of using computational notebooks to support users with little or no programming experience. While there may be numerous approaches to developing computational notebooks for non-programmers, this chapter takes the approach of leveraging a domain-specific language (DSL) to parse text or perform actions in the context of a free-text note. In this research the computational nature of the notebook only involves parsing free-text to create a structured medication order to be passed to another part of the medical record and the DSL involved in mainly a markup language. However, a similar interaction of parsing free-text could be used to perform other computations, such as retrieving and plotting a patient’s recent vital signs directly in the note [105]. Moreover this chapter demonstrates an approach to mixing explanatory and actionable text that does not rely on the cell-based organization of current computational notebooks but relies on “tagging” actionable text within a stream of explanatory text.

5.3 ActiveNotes Prototype

To explore the feasibility of free-text order entry, I, along with a team of collaborators, helped develop ActiveNotes (Figure 5.1). My efforts focused on interaction design and testing while my collaborators set requirements, developed application infrastructure, and helped run the study described in this chapter. ActiveNotes is a clinical note editor that lets clinicians create, modify, and delete notes. Like most EHR note editors, there are no facilities for rich-text formatting such as bolding or font selection.

ActiveNotes’ main strength is in its ability to parse semi-structured text into full medication orders. To start an order, users type the text “#med” anywhere in their note which opens a small order specification dialog on top of the note (Figure 5.1A). This window provides a search bar where clinicians can type their order. Following medication ordering convention ActiveNotes expects the medication and dose information

The screenshot shows the 'Active Notes' interface for Jane Smith at PrimaryCare Clinic, La Jolla. The patient's information includes DOB: Jan 02, 1974 and SSN: 123-45-6789. The note content is as follows:

Active Problems
 HTN
 Hyperlipidemia
 Hypothyroidism
 Meniscal tear
 Vitamin D deficiency
 Urinary Tract Infection

Active Medications
 Atenolol 50 po qd
 Simvastatin 20 po qd
 Levothyroxine 0.075 po qd
 Naproxen 500 po bid

Lab Orders
 HgBA1c in 6 months
 Lipid Profile in 6 months

S: Patient is a 89M with hx HTN, HPL, PTSD, OA here for annual exam. He is taking tamulosin and requests refills. Also c/o occasional urinary incontinence.

O:
Labs:
 Cholesterol panel: normal

A/P:
 1. HTN: well-controlled
 2. HPL: simvastatin 20mg bedtime #30 w 3 refills
 3. OA: A
 4. BPH:

A dropdown menu is open for item 4, showing a table with columns: Med, Strength, Form, Route, Schedule, Days, Take, Refills, Qty, Pickup. The selected item is: simvastatin 20mg bedtime #30 w 3 refills.

At the bottom, there are buttons for 'Delete Note', 'Sign Note', and 'Save Note'. The note was created and modified on March 3, 2015.

(B) Search box showing 'sirtv' and 'simvastatin' with a dropdown menu listing 'SIMVASTATIN'.

(C) Dropdown menu showing 'simvastatin 20mg bedtime #30 w 3 refills' with a table header: Med Strength Form Route Schedule Days Take Refills Qty Pickup.

(D) Medication order form for SIMVASTATIN. Fields include: Medication (SIMVASTATIN), Strength (20MG), Dose Form, Route, Schedule (Bedtime), Days Supply (30), Take, Refills (3), and Quantity. An 'Apply' button is at the bottom.

Figure 5.1: ActiveNotes in use. (A) The main note editor after an order has been started. A search box appears wherever a user types “#med” in their note. (B) The box provides auto-completion for drug names. (C) Medication order components are highlighted after ActiveNotes’ recognizes them. (D) Users can use an order template if needed.

to be entered first and provides auto-completion for these fields (Figure 5.1B) based on an underlying medication database built around RxNorm, a medication ontology [5]. As text is entered in the search field, it also appears in the note wherever the #med tag was invoked. After entering the medication and dose fields, the order is complete

enough for checkout and clinicians can hit their “Enter” key to finish the order, which closes the order specification dialog. Alternatively, users can continue to specify other aspects of the order such as form, route, schedule, refills, etc. As each part of the order is recognized, the dialog highlights the components’ corresponding label in blue in the popover window (Figure 5.1C). Users can also access an order specification drop-down menu by clicking the triangle to the right of the search box (Figure 5.1D).

After a medication is specified through the dialog, it can be edited or deleted through an action menu that appears when users right-click or hover over the order text. Anticipating future uses of ActiveNotes, we have developed order specific tasks including refill, reorder, and discontinue that can all be accomplished in a single click.

Once finished with their note, users can click the “Rx” symbol above the note to go to the “Order Checkout” page to review and sign any orders they placed (Figure 5.2). This page is pre-populated with medication orders from the note. Here, ActiveNotes asks for any additional information needed to fully specify the order and also fills in default values for some fields that were not specified in the note, such as pickup location or number of refills. ActiveNotes currently supports only medication orders but could be expanded to support orders for lab tests, imaging, and consultations.

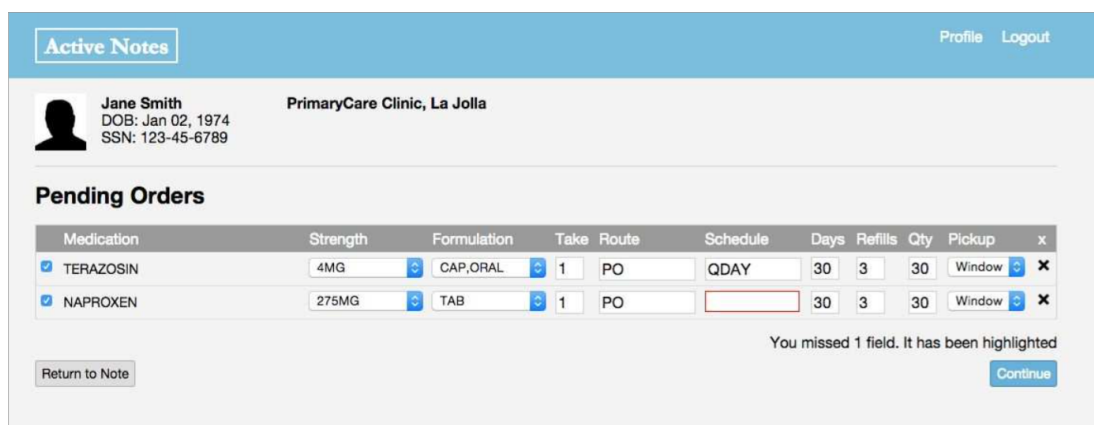


Figure 5.2: ActiveNotes checkout screen where orders can be reviewed and edited before being signed. The red box highlights missing information.

5.4 Evaluation

We tested ActiveNotes with eight clinicians from San Diego’s Veterans Affairs hospital and UC San Diego Medical Center. Clinicians’ specializations included internal medicine (3), infectious diseases (4), and surgery (1). Our participant pool included both inpatient and outpatient practitioners, ranging from fellows to senior clinicians. Each test lasted 30 minutes or less. Clinicians were first shown a short video explaining ActiveNotes’ features. They were then given two short descriptions of canonical outpatient visits and asked to create a progress note for each visit in ActiveNotes. Each scenario gave basic demographic and assessment information and asked the clinicians to place four new medication orders from within their note. The orders were largely phrased without standard shorthand (e.g. PO, BID) so we could observe the shorthand clinicians used naturally. After completing both progress notes, we asked the clinicians to comment on their experience with ActiveNotes. We tracked how long it took clinicians to successfully place each order, the shorthand they used while placing each order, any unsupported attempted uses of the system, and participants’ comments on desired features as well as the overall usefulness of ActiveNotes.

5.5 Results

Time to Order: Clinicians took, on average, 59 seconds to place each order, though this time varied greatly between clinicians and orders. For example, one clinician averaged just 30 seconds to place each order whereas another averaged 1 minute 56 seconds. Also, it took clinicians an average of just 39 seconds to place an order for “Simvastatin 20mg PO at bedtime daily 30 tablets to be dispensed with 3 refills” whereas it took an average of 1 minute 17 seconds to place an order for “Aspirin 81mg once daily 100 tablets to be dispensed with 3 refills”. Time to order also depended on whether the

clinician initially used shorthand that ActiveNotes recognized. For example, whereas ActiveNotes could parse “#30” as quantity information, it could not parse “Dispense: 30”. The orders that took the longest to complete were those in which clinicians tried to use several different shorthands that ActiveNotes did not recognize and then resorted to using the drop-down order specification menu (accessible by clicking the triangle to the right of the search box). Since one of our main objectives was to identify the shorthand clinicians naturally use, we did not tell clinicians what type of shorthand ActiveNotes was programmed to recognize and let them decide when they wanted to fall-back on the drop-down menu.

Shorthand: Clinicians used a variety of shorthand when placing free-text orders. Representative examples are shown in Table 5.1. Whereas the medication, dose, form, route, and days fields saw little variation, schedule, quantity, refills, and pickup information was entered in a number of different ways.

Table 5.1: Shorthand used when placing orders

Information	Shorthand
Medication	Aspirin, asa (abbreviation)
Dose	100mg, 75mcg
Form	tab, enteric coated
Route	po, oral
Schedule	bid, at bedtime, once a day, 1x/day, prn for dizziness
Days	30 days
Quantity	#30, qty 30, Dispense: 30
Refills	rf, no refills, refills 3, one refill
Pickup	pick, mail, to be mailed, pickup at clinic, pickup window

Functionality: As is common with prototypes, our participants wanted ActiveNotes to support additional functionality. We particularly valued this feedback since expected functionality can guide future development. There were six recurring function requests; the first three regarded time saving assistance while the last three address broader integration with the EHR. These included:

1. **Default Values:** Five clinicians wanted ActiveNotes to have default values for fields like schedule, refills, and pickup. While ActiveNotes' checkout window defaulted to "3 refills" and "Window" pickup, it did not have a default value for schedule. Moreover, there was no way to see, when placing an order in the note, that ActiveNotes would default to these values in the checkout window.
2. **Auto-populated Values:** Four clinicians wanted ActiveNotes to automatically populate form and quantity information whenever possible. For example, ActiveNotes could automatically mark 'Tab' if the specified medication and dose could only come in a tablet form. Based on a schedule and duration of "BID for 30 days", ActiveNotes could also automatically assign a quantity of 60 tablets.
3. **Order Entry Invocation:** Four clinicians wanted a simpler way to invoke the order entry window than the "#med" syntax. Clinicians either wanted to type out the full order and then hit a hotkey to tell ActiveNotes to parse the preceding text, or only wanted to invoke the "#med" command once per note and then fill out multiple orders in a row.
4. **Additional Orders:** Three clinicians expected ActiveNotes to handle additional order types including radiology, labs, and consults.
5. **Additional Parsing:** One clinician wanted ActiveNotes to be able to parse other note text, such as active problems, and save it to the appropriate part of the EHR.
6. **Information Retrieval:** Two clinicians wanted ActiveNotes to support rich information retrieval such as "retrieve most recent colonoscopy".

Usefulness: Clinicians saw the current implementation of ActiveNotes as useful in two distinct ways. First, they thought it was useful to not have to switch between sections of the EHR to document and place orders. As one clinician remarked:

I love not having to go out of the screen [to place orders] — P1

Secondly, clinicians saw directly linking orders to documentation as a way to avoid overlooking orders:

The thing that I really like about this is... having the order directly tied to the documentation of that order in the note. The issue I run into sometimes is that I write my note and I'm waiting to do my orders and then I have to... make sure... all the things I said I was going to do in the note I actually order. — P3

More broadly clinicians saw value in using interactive notes to populate the EHR with structured information:

If this had the ability to take everything in the note and just automatically download it... if you just did like #activeproblems from the note it would just put it all into that section... that would dramatically improve efficiency because there's a lot of that duplication that we're currently doing. — P7

This is opposed to the current model of importing structured information into the note:

As it currently stands, it's the opposite. You have to first put in everything in the active problem list or medication list, and then you can populate it into the note by using smart keystrokes, but it would be nice when you're initially seeing someone to not have to write the whole note and then repeat everything — P7

5.6 Discussion

Our first research question asked what types of shorthand clinicians use while placing free-text medication orders. Whereas shorthand for some fields was consistent (e.g., ‘asa’ for aspirin, ‘mg’ for milligrams) shorthand for fields such as schedule and number of refills was more varied. It will be important for interactive notes to embrace a rich set of terminology that goes beyond current ontologies, such as RxNorm, to include a variety of terms for fields such as schedule, refills, and pickup.

Secondly, we asked what types of functionality clinicians expect from a computerized free-text order entry system. The six recurring functionality requests fell into two categories: time saving assistance and broader integration with the EHR. Along the lines of time saving assistance, clinicians wanted to specify their order with as little typing or clicking as possible. This principle can be seen in their desire to invoke the “#med” dialog only once and place multiple orders in a row. It can also be seen in participants’ desire for ActiveNotes to recognize when a particular dose of a drug only comes in one form, or to automatically calculate quantity given a prescription’s schedule and duration. Clinicians also wanted ActiveNotes to be more broadly integrated with the EHR. This included both extending the types of orders it recognized to include labs, imaging, and consults as well as extending ActiveNotes’ parsing to cover other types of structured information such as active problems and family history.

Finally, we asked if clinicians saw free-text order entry as a useful addition to their note editor. Many saw free-text order entry as useful in a number of ways including being a potential time saver, less distracting than form-based input, and requiring less navigation compared to current EHRs. These clinicians thought a system like ActiveNotes could save them from needing to enter information twice, once in a structured format and then in an unstructured format and may also require less attention than form-based or drop-down based entry, letting providers focus more on their

patients if they choose to document during the patient encounter.

There are additional potential technical benefits to having clinicians tag note content as structured data (e.g., invoking #med when placing a medication order) as compared with post-hoc Natural Language Processing (NLP). First, it enables the parser, whether it is looking for orders or a problem list, to use a more targeted domain specific language for recognizing terms. Second, it adds structure to the note by marking which parts of the note refer to medications, conditions, and so on. This tagging could assist with later NLP or the development of richer note interactions related to targeted search, filtering, or highlighting of specific content.

Looking towards future designs, free-text entry of structured information was not as familiar an interaction paradigm for our participants as menu-driven interfaces. One participant did not grasp that ActiveNotes would let him document and place orders at the same time. Instead, he wrote each order in the note twice, first with fairly standard shorthand (e.g. po, #30) while “documenting” and then with less standard shorthand when “ordering” (e.g. Dispense: 30). Beyond developing a robust interactive note, it will take time before some clinicians are comfortable with a new paradigm.

It will also take further iteration to investigate how best to make the features of free-text entry discoverable and provide adequate feedback to users about the results of their actions. What keys users press to complete an order, how they select an auto-completed phrase, and how the system shows that it recognizes part of the order all need to be carefully designed to fit clinicians’ expectations.

5.7 Conclusion

This study takes a step towards a vision of developing an interactive clinical note that unify entry and access of patient information with the execution of treatment plans. We tested one critical feature of such a note, free-text order entry, and found that (i) clin-

icians use a variety of shorthand when placing free-text medication orders, particularly when specifying schedule, pickup location, and number of refills, (ii) clinicians want to specify a minimum number of fields and rely on default or auto-populated values to fill in routine information, and (iii) clinicians see free-text order entry as a useful addition to their progress notes with the potential to save time and ensure orders are not overlooked.

While ActiveNotes is too rough to be deployed in a live EHR, its free-text order entry approach was validated by representative end-users. By considering the note as the central element of the EHR and incorporating interactions and operations that typically span multiple parts of EHRs, we have an opportunity to transform documentation from being a complicated and time-consuming clerical task to a more natural interaction with the patient data.

The results of this study also have implications for other domains of data-driven work by demonstrating that computational notebooks can combine documentation and action on data without requiring use of a general-purpose programming language. Instead, interfaces can use domain-specific languages to help users specify data, perform actions, or analyze data in the same document used to explain their thought process. These domain specific languages will need to be carefully crafted for each use-case and perhaps even tailored to specific individuals. Designers will need to grapple with how precise to make the languages. Should users have only one way to invoke a command, or is the domain and list of possible actions limited enough that multiple synonymous textual commands will be recognized? Or might non-programmers be better served by using interactive drop-downs or menus to specify commands rather than text?

ActiveNotes is a rather limited example of a domain-specific computational notebooks for non-programmers. It primarily uses a domain-specific markup language to enable clinicians to specify medication orders with free text that is human readable and can be embedded within a clinical note. Future interfaces will need to explore aspects

of domain-specific computational notebooks beyond how users invoke commands or specify data including how to plot data within a notebook, how to make possible actions discoverable, and how to provide feedback about the status of pending actions or data analysis steps. While future systems could mimic current computational notebooks, they do not have to. For example Wilcox et al.'s clinical notebook provided an exploratory side-panel where information related to selected text in the note could be rendered and then manually copied into the note [105]. In their system, if the text 'ABP' was highlighted in a note, the side panel would plot the patient's arterial blood pressure readings for the last few days with a line graph that could be copied into the main note. This paradigm of separating the data-driven working space from the final notebook deserves further exploration.

Future computational notebooks also need not rely on textual interactions with an underlying domain-specific language. Menu and form-based interactions could be used to select, analyze, and act on data in ways that can still be embedded within a textual narrative. However there is an elegance in mixing textual commands with explanatory text that allows a tight coupling of commentary and action.

5.8 Synthesis

This chapter presents an exploratory study of how clinicians might use free-text order entry to place medication orders from within a clinical progress note. The contributions of this chapter are:

1. Evidence that clinicians can place and find it useful to place free-text medication orders within a clinical note
2. Theoretical perspective that computational notebooks can support a broader range of data-driven activities by leveraging domain-specific languages

3. A prototype clinical note editor, ActiveNotes, built in collaboration with the co-authors on the following paper

This chapter, in part, includes portions of material as it appears in *Validating free-text order entry for a note-centric EHR* by Adam Rule, Steven Rick, Michael Chiu, Phillip Rios, Shazia Ashfaq, Alan Calvitti, Wesley Chan, Nadir Weibel, and Zia Agha published in proceedings of the 2015 annual symposium of the American Medical Informatics Association. The dissertation author was the primary author of this paper.

6 Conclusion

This chapter revisits the contributions of this dissertation, explores their implications, and highlights directions for future research. It suggests the tension between exploration and explanation is fundamental to data analysis but that with careful design, computational notebooks have potential to reduce this tension and support a range of data-driven activities, even in domains where general-purpose programming is not the primary means of interacting with data.

Data analysis is increasingly vital to the work of many individuals and organizations. Yet, tracking and sharing analytical steps, reasoning, and results remains a challenge, making it difficult to review, resume, replicate, or build on existing analyses. The recent rise of computational notebooks — interactive documents combining code, visualizations, and text in a single file — has been hailed as a sea-change in data analysis, ushering in a new era of openness and reproducibility, particularly in the sciences. But there has been little evaluation of how computational notebooks are being used, the benefits they provide, or how they might be improved. This dissertation leverages the rising popularity of computational notebooks as a unique opportunity to study the challenges of performing, documenting, and sharing data analyses. It characterizes how computational notebooks are currently addressing these challenges, and how they might be designed to do so more effectively.

6.1 Contributions

This dissertation has four types of contribution: empirical results, theoretical perspectives, prototype systems, and an open dataset (Figure 1.3).

CHAPTER 3 presented three studies characterizing the use of *computational narrative* in computational notebooks. Through them I find that many notebooks lack a basic prerequisite for computational narrative: a single word of explanatory text. Even those notebooks that had explanatory text were mostly loose collections of notes and scripts without a coherent structure. And even in notebooks supplementing academic publications only about a third used explanatory text to discuss analytical reasoning or interpret results. Instead, most used explanatory text to simply label steps of the analysis. Speaking to analysts revealed that this lack of annotation stems from the tendency for exploratory analyses to produce messy notebooks that seem personal and are tedious to clean. Together these findings support a theory that the tension between exploration and explanation makes it difficult to track and share data analyses. In addition to these empirical findings and theoretical perspective, with the help of the UC San Diego Library I released all data from the first study, including over a million computational notebooks, online for others to study (<https://doi.org/10.6075/J0JW8C39>).

In CHAPTER 4 I built on these findings by exploring how computational notebooks might be redesigned to encourage clearer communication of complex data analyses, in particular by including more explanatory text and explicit organization. I designed and developed Janus, a Jupyter Notebook extension that enabled analysts to add history and hierarchy to their notebooks. Through two studies I demonstrated that hierarchy in particular enables analysts to flexibly navigate and organize their notebooks in ways that support both the ongoing analysis and later communication. These findings support the theoretical perspective that flexible organization and navigation of code, visualizations, and text can help reduce the tension between exploration and

explanation by providing a lightweight means to tailor notebooks to the task at hand.

In CHAPTER 5 I further explored how the the paradigm of computational notebooks might support data-driven work in a domain where general-purpose programming is not the primary means of interacting with data. With a team of collaborators I helped design and test ActiveNotes, a prototype clinical note editor that enables clinicians to place medication orders within free-text notes. This work provided both empirical results about how clinicians might use a free-text order entry, and the theoretical perspective that notebooks might leverage domain-specific languages to support data-driven work in domains outside traditional data analysis.

6.2 Implications

These findings have implications both for the design of current computational notebooks as well as a broader class of interfaces supporting data-driven work. They also have implications for open science and policies meant to encourage transparency in data analysis. This dissertation demonstrates that while computational notebooks allow analysts to craft rich computational narratives, significant barriers still prevent them from doing so. If interactive media, social practices, and policy are to encourage clearer communication of everyday analyses, they need to incentivise, ease, or routinize the process of turning messy exploratory code and results into compelling descriptions of analyses.

6.2.1 Technical Interventions

This encouragement could take the form of notebooks extensions like Janus (CHAPTER 4) that help analysts annotate and organize their notebooks in-the-moment. For example, notebooks could provide linters that detect computational or narrative “debt” and guide users through adding more informative explanations or refactoring their code.

These aids might be as subtle as a “computational spell-check” that underlines code or explanations that might be improved. Alternatively, notebooks might change their default new cell type to be text rather than code [75], encouraging analysts to articulate their next step before trying to implement it in code. Except in educational contexts where more explicit scaffolding may be required, it is likely these subtler interventions that will be most successful as they are less likely to derail data exploration.

6.2.2 Social Interventions

However, interventions need not be technical. The most impactful change might come from individual labs establishing new norms for how analyses are documented and shared. As one interviewee from the third study in CHAPTER 3 noted, many of his colleagues seemed “put-off” when he presented results from a notebook during lab meeting, feeling that he did not take any time to prepare. What if labs expected weekly presentations from notebooks and encouraged researchers to spend time cleaning them rather than assembling slide decks?

More likely there will need to be a broader movement toward recognizing and incentivizing the work that goes into clearly presenting data analyses. Methods of measuring academic output will need to be revised to track not just the number of papers citing an analysis, but also the number of projects reusing code or data from that analysis. Academic journals will need to adopt new forms of publication that elevate datasets and computational notebooks to first class-citizens on their websites rather than simply supplemental material for the curious. Companies will need to go beyond establishing guidelines [72, 23] to providing stronger incentives such as bonuses for those who take the time to address narrative and technical debt in their analytical records.

6.2.3 Wider Use of Notebooks

This research hints at the variety of notebook users and uses. Whereas some analysts see notebooks primarily as an experimental playground for code, some managers see them mainly as a vehicle for sharing polished results. Some users have extensive programming experience while others just know how to tweak a few parameters in an existing notebook.

Computational notebooks could do more to support these varied uses by presenting different views to different users and making it easier to manually, automatically, or semi-automatically construct these views. For example, notebooks could use execution histories and program slicing to help analysts decide which portions of the notebook they need to view at any one time to accomplish a particular task. They could ease the process of turning a hard-coded analytical step into a flexible one that encourages exploration by allowing readers to vary parameters using interactive widgets rather than code.

Beyond their current form, there is potential for notebooks to support data-driven work in domains where writing code is not the primary means of interacting with data. Outside of healthcare (CHAPTER 5), lawyers may benefit from being able to query, analyze, and annotate large collections of legal documents within a single notebook. Law enforcement or government employees may similarly benefit from being able to sift through and annotate crime or civic data. Supply chain managers could benefit from compiling interactive reports tracking their explorations of alternative product sourcing schemes. Over time computational notebooks might attain the status of a core computational medium like spreadsheets or word processing documents that are used by children and elders, car mechanics and scientists for a variety of data-driven activities. However, it is likely that a one-notebook-fits-all approach will not work, but that notebooks will need to provide facilities specific to each domain and skill-set, even if they share an underlying computational infrastructure.

6.2.4 Beyond Notebooks

This dissertation also has implications for the design of a broader class of current and future interfaces that support data analysis. These include current data analysis tools such as Tableau and Excel or more experimental systems such as Many Eyes or sense.us. Each might benefit from incentivizing, easing, or routinizing the explanation of data analyses through facilities that embed explanatory text alongside data transformation and visualization.

While powerful, the linear structure of notebooks limits how they can be used to explore data and present results. More could be done to embed computation and annotation in form factors such as slide decks and 2-D graphics that afford different ways of presenting analyses. This dissertation suggests that whatever the medium, there must be an immediate benefit to any organization or annotation activity, particularly when those using the medium are not professional storytellers and journalists. These populations have been the focus of prior work on creative narrative visualizations [85], but have different incentives and goals than the everyday analyst.

6.2.5 Education

This work also has implications for education. As one interviewee in CHAPTER 3 noted, when starting her chemistry and biology labs as an undergraduate she was trained in a specific, opinionated way of organizing her notebooks. Her paper notebooks did not constrain her to use them in this way, but social practice did. Currently there is little of this formal training for using computational notebooks beyond workshops like those provided by Software Carpentry and Data Carpentry [107]. Undergraduate and graduate education should focus on helping students learn to annotate, update, and revise their analyses in ways that they and others can understand. In this way analytical education should look more like writing education.

6.3 Future Work

6.3.1 Characterizing Data Analysis

Future work could better characterize how data analyses are performed in computational notebooks. Janus supports detailed tracking of every edit made to a computational notebook and could be used in longitudinal studies to examine how notebooks evolve over time. For example, Figure 6.1 shows a visualization of one notebook’s evolution in the style of History Flow [100] generated with notebook revision data from Janus. Visualizations like this reveal when analysts add commentary to their notebook, when they get stuck debugging a particular cell, and when they merge or delete cells to clean their notebooks. Other tracking software could bolster this form of study by tracking interactions with programs and websites outside the notebook. Such data could help us better understand the tools and practices analysts employ in everyday analyses that are not apparent in the final state of a notebook or interviews [29, 50]. Other studies of current use could compare how notebooks are used in different domains of practice across broad domains such as enterprise, education, and academic research, or smaller ones such as comparing use of notebooks in different academic disciplines.

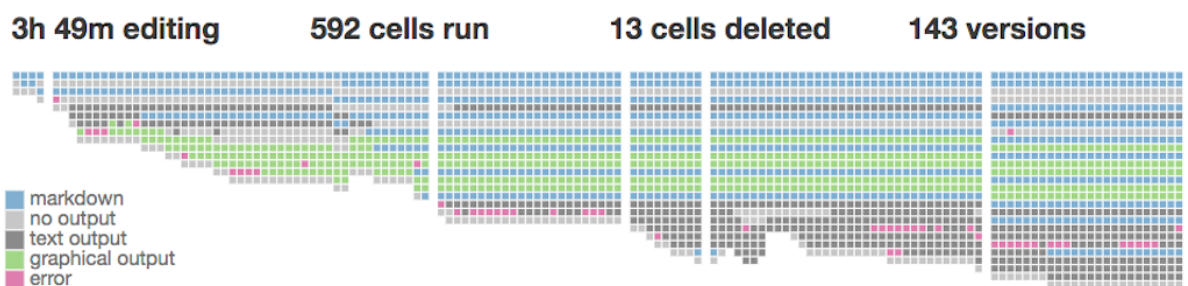


Figure 6.1: Example visualization of how one notebook evolved over time using data collected with Janus. Each column is a subsequent version of the same notebook, each square is one cell of the notebook with the topmost cell being the first cell in the notebook, and gaps between columns represent moments where more than 15 minutes elapsed between notebook edits.

6.3.2 Redesigning Computational Notebooks

Other work could continue to explore how to redesign notebooks to encourage and enable users to write more compelling narratives and be more willing to share their notebooks. One approach would be to compare and systematically test features from the growing array of computational notebooks on the market (e.g., Jupyter Notebook, R Notebooks, Mathematica, SageMath, Google Collaboratory, Mozilla Iodide). Some include extensive history tools, some have rich variable inspectors, and many take different approaches to hiding particular cells in an effort to support creation of “presentation” notebooks. Which of these features are most effective in supporting analysis or later communications, and how are they currently being used in practice?

Another approach would be to continue incrementing the design of notebooks such as Jupyter Notebook and Jupyter Lab by building novel extensions to explicitly encourage annotation and organization. As was the case with this dissertation these tools could aim to do so by encouraging flexible organization and navigation in-the-moment. However, there is also fertile ground for developing tools that more explicitly encourage best practices either during the analysis or post-hoc during a dedicated notebook cleaning phase. For example, imagine students in an introductory data analysis course using a version of Jupyter Notebook with a built-in linter that identified and coached them through eliminating technical and narrative debt. This could include identifying poorly named variables, duplicate code, or results without an explanation.

6.3.3 Computational Notebooks for Non-Programmers

Research could also explore developing notebooks for domains such as health-care, government, and engineering where a number of practitioners do not program. Here research could focus on the development of domain specific languages [37], to support interactions such as those used in ActiveNotes or Tableau that enable people

to analyze data without using a general-purpose programming language. Fast et al.'s Iris conversational agent [20] is a promising step in this direction, enabling people to analyze data through chat with a data analysis agent. One can imagine the development of different agents for different types of analysis; one that is expert in analyzing financial transactions, another with crime data, but that all keep a record of the steps and reasoning involved. These notebooks do not necessarily need to follow the same linear-collection-of-cells paradigm employed by current notebooks, but could experiment with different ways of embedding computation in narrative text. More research will be needed to develop the interactions and domain specific languages appropriate for each domain. Of particular importance is how to enable non-programmers in each domain to develop the language and interactions appropriate for their field, just as data analysts who program currently develop libraries to aid their kind of analyses.

6.3.4 Reading and Reuse of Computational Notebooks

This dissertation has focused on how analysts use computational notebooks to generate computational narratives but future research should also explore how computational narratives are consumed, particularly by those with little or no programming experience. Managers and non-technical colleagues are often the primary audience of data analyses, so how might notebooks better support their needs? Which displays work best for comprehension? How might notebooks be designed to encourage authors to organize their notebooks in these ways?

Future research could also explore how to make analyses in computational notebooks reusable at scale. For example, in announcing the first Public Library of Science (PLOS) journal in 2003, the journal's editors claimed that [7]:

Freeing the information in the scientific literature from the fixed sequence of pages and the arbitrary boundaries drawn by journals or publishers

— the electronic vestiges of paper publication — opens up myriad new possibilities for navigating, integrating, mining, annotating, and mapping connections in the high-dimensional space of scientific knowledge.

How might computational notebooks leverage their interactivity and computability to support support such “navigating, integrating, ‘mining’, annotating, and mapping” in ways that are “far more useful than the electronic equivalent of millions of individual articles in rows of journals on library shelves” [7]?

6.3.5 Other Barriers to Sharing Data Analyses

Finally there are a host of barriers to reproducibility and open science that this research did not address but that will still have a substantial impact on the future of computational notebooks. These include technical challenges such as providing appropriate access to data as well as supporting interoperability, dependency declaration, and containerization so analyses can be rerun on any computer.

More challenging will be navigating the myriad social issues surrounding sharing of data analyses. For example, what are the ethical implications of tracking and sharing more of the messy process of data analysis? Clinicians, for example, are already careful what they document about patient care to defend against legal action. How might an expectation of greater sharing of analyses actually encourage less sharing of a process that is necessarily messy and can lead to wrong conclusions at times?

And how might greater sharing of data be allowed without loss of privacy when human data is involved [18]? Computational notebooks are most useful when they are interactive and users have access to the data underlying the analysis so they can tweak, re-run, and reuse portions of it. Yet, as more and more data is shared, there is greater chance of re-identifying individuals by combining deidentified records [91]. Explanations of analyses are not likely to be shared if the underlying data cannot.

6.4 Final Thoughts

This dissertation has characterized one of the core challenges of data analysis: managing the tension between data exploration and process explanation to generate insights that are open to inspection, reproducible, and sound. It has also explored how redesigning computational notebooks might reduce this tension. My hope is that it has highlighted a way forward that helps make widespread use and sharing of computational narratives less of a far off dream and more of a present reality.

Bibliography

- [1] Cerner. <https://cerner.com/>, 2018. [Online; accessed 9-May-2018].
- [2] Distil. <https://distill.pub/>, 2018. [Online; accessed 9-May-2018].
- [3] Epic. <https://epic.com/>, 2018. [Online; accessed 9-May-2018].
- [4] Hello, colaboratory. <https://colab.research.google.com/>, 2018. [Online; accessed 9-May-2018].
- [5] Rxnorm. <https://www.nlm.nih.gov/research/umls/rxnorm/>, 2018. [Online; accessed 9-May-2018].
- [6] Nanette Brown, Ipek Ozkaya, Raghvinder Sangwan, Carolyn Seaman, Kevin Sullivan, Nico Zazworka, Yuanfang Cai, Yuepu Guo, Rick Kazman, Miryung Kim, Philippe Kruchten, Erin Lim, Alan MacCormack, and Robert Nord. Managing technical debt in software-reliant systems. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, pages 47–52. ACM, 2010.
- [7] Patrick Brown, Michael Eisen, and Harold Varmus. Why plos became a publisher. *PLoS biology*, 1(1):e36, 2003.
- [8] Richard Buchanan. Wicked Problems in Design Thinking. *Design Issues*, 8(2):5–21, 1992.
- [9] Alan Calvitti, Neal Farber, Yunan Chen, Danielle Zuest, Lin Liu, Kristin Bell, Barbara Gray, and Zia Agha. Temporal analysis of physicians’ ehr workflow during outpatient visits. In *Healthcare Informatics, Imaging and Systems Biology (HISB), 2012 IEEE Second International Conference on*, pages 140–140. IEEE, 2012.
- [10] Kathryn Carlson, Sara McFadden, and Shari Barkin. Improving documentation timeliness: A brighter future for the electronic medical record in resident clinics. *Academic Medicine*, 90(12):1641–1645, 2015.
- [11] Nicholas Chen, Francois Guimbretiere, and Abigail Sellen. Designing a multi-slate reading environment to support active reading activities. *ACM Transactions on Computer-Human Interaction*, 19(3):1–35, October 2012.

- [12] Brian Colloron and Hamilton Ulmer. The iodide notebook. <https://github.com/iodide-project/iodide>, 2018.
- [13] Jeffrey Conklin and KC Yakemovic. A process-oriented approach to design rationale. *Human-Computer Interaction*, 6(3):357–391, 1991.
- [14] Ward Cunningham. The wycash portfolio management system. *ACM SIGPLAN OOPS Messenger*, 4(2):29–30, 1993.
- [15] Tao Ding and Patrick Schloss. Dynamics and associations of microbial community types across the human body. *Nature*, 509(7500):357, 2014.
- [16] Jon Duke, Justin Morea, Burke Mamlin, Douglas Martin, Linas Simonaitis, Blaine Takesue, Brian Dixon, and Paul Dexter. Regenstrief institute’s medical gopher: A next-generation homegrown electronic medical record system. *International journal of medical informatics*, 83(3):170–179, 2014.
- [17] Brian Dunbar. Nasa - shuttle computers navigate record of reliability. https://www.nasa.gov/mission_pages/shuttle/flyout/flyfeature_shuttlecomputers.html, 2010. [Online; accessed 9-May-2018].
- [18] Cynthia Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.
- [19] Daniel Epstein, An Ping, James Fogarty, and Sean Munson. A lived informatics model of personal informatics. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 731–742. ACM, 2015.
- [20] Ethan Fast, Binbin Chen, Julia Mendelsohn, Jonathan Bassen, and Michael Bernstein. Iris: A conversational agent for complex tasks. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI ’18, pages 473:1–473:12, New York, NY, USA, 2018. ACM.
- [21] Andrew Forward and Timothy Lethbridge. The relevance of software documentation, tools and technologies: A survey. In *Proceedings of the 2002 ACM Symposium on Document Engineering*, DocEng ’02, pages 26–33, New York, NY, USA, 2002. ACM.
- [22] Martin Fowler, Kent Beck, John Brant, William Opdyke, and Don Roberts. *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 1999.
- [23] Dan Frank. Reproducible research: Stripes approach to data science. <https://stripe.com/blog/reproducible-research>, November 2016. [Online; accessed 9-May-2018].

- [24] Mark Friedberg. *Factors affecting physician professional satisfaction and their implications for patient care, health systems, and health policy*. Rand Corporation, 2013.
- [25] Peter Galison. *Image and logic: A material culture of microphysics*. University of Chicago Press, 1997.
- [26] Nahum Gershon and Ward Page. What storytelling can do for information visualization. *Communications of the ACM*, 44(8):31–37, 2001.
- [27] David Gotz and Michelle Zhou. Characterizing users’ visual analytic activity for insight provenance. *Information Visualization*, 8(1):42–55, 2009.
- [28] Brian Granger, Chris Colbert, and Ian Rose. Jupyterlab:the next generation jupyter frontend. JupyterCon, 2017.
- [29] Philip Guo. *Software tools to facilitate research programming*. PhD thesis, Stanford University, 2012.
- [30] Philip Guo and Margo Seltzer. Burrito: Wrapping your lab notebook in computational infrastructure. In *Proceedings of the 4th USENIX Workshop on the Theory and Practice of Provenance, TaPP’12*, Berkeley, CA, USA, 2012. USENIX Association.
- [31] Uta Guo, Lu Chen, and Parag Mehta. Electronic health record innovations: Helping physicians—one less click at a time. *Health Information Management Journal*, 46(3):140–144, 2017.
- [32] Janet Haas, Suzanne Bakken, Tiffani Bright, Genevieve Melton, Peter Stetson, and Stephen Johnson. Clinicians perceptions of usability of enote. In *AMIA Annual Symposium Proceedings*, volume 2005, page 973. American Medical Informatics Association, 2005.
- [33] Richard Harper and Abigail Sellen. Collaborative tools and the practicalities of professional work at the international monetary fund. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 122–129. ACM Press / Addison-Wesley Publishing Co., 1995.
- [34] Pamela Hartzband and Jerome Groopman. Off the record—avoiding the pitfalls of going electronic. *New England Journal of Medicine*, 358(16):1656–1657, 2008.
- [35] Christian Heath and Paul Luff. Documents and professional practice:bad organizational reasons for good clinical records. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work*, pages 354–363. ACM, 1996.
- [36] Jeffrey Heer and Maneesh Agrawala. Design Considerations for Collaborative Visual Analytics. *Information Visualization*, 7(1):49–62, March 2008.

- [37] Jeffrey Heer and Michael Bostock. Declarative language design for interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1149–1156, 2010.
- [38] Jeffrey Heer and Ben Shneiderman. Interactive dynamics for visual analysis. *Queue*, 10(2):30, 2012.
- [39] Jeffrey Heer, Fernanda Viégas, and Martin Wattenberg. Voyagers and voyeurs: Supporting asynchronous collaborative information visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 1029–1038, New York, NY, USA, 2007. ACM.
- [40] Thomas Herndon, Michael Ash, and Robert Pollin. Does high public debt consistently stifle economic growth? a critique of reinhart and rogoff. *Cambridge journal of economics*, 38(2):257–279, 2014.
- [41] Brian Hilligoss and Kai Zheng. Chart biopsy: an emerging medical practice enabled by electronic health records and its impacts on emergency department-inpatient admission handoffs. *Journal of the American Medical Informatics Association*, 20(2):260–267, 2012.
- [42] Robert Hirschtick. Copy-and-paste. *Jama*, 295(20):2335–2336, 2006.
- [43] Matthew Hong, Anne Marie Piper, Nadir Weibel, Simon Olberding, and James Hollan. Microanalysis of active reading behavior to inform design of interactive desktop workspaces. In *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces*, pages 215–224. ACM, 2012.
- [44] Jessica Hullman and Nick Diakopoulos. Visualization rhetoric: Framing effects in narrative visualization. *IEEE transactions on visualization and computer graphics*, 17(12):2231–2240, 2011.
- [45] Jessica Hullman, Steven Drucker, Nathalie Henry Riche, Bongshin Lee, Danyel Fisher, and Eytan Adar. A deeper understanding of sequence in narrative visualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2406–2415, 2013.
- [46] Edwin Hutchins. *Cognition in the Wild*. MIT press, 1995.
- [47] Hilary Hutchinson, Wendy Mackay, Bo Westerlund, Benjamin B. Bederson, Allison Druin, Catherine Plaisant, Michel Beaudouin-Lafon, Stphane Conversy, Helen Evans, and Heiko Hansen. Technology probes: inspiring design for and with families. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 17–24. ACM, 2003.
- [48] Project Jupyter. A gallery of interesting jupyter notebooks. <https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks>, 2018.

- [49] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3363–3372. ACM, 2011.
- [50] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Enterprise data analysis and visualization: An interview study. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2917–2926, 2012.
- [51] Mary Beth Kery, Marissa Radensky, Mahima Arya, Bonnie John, and Brad Myers. 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*, CHI '18, pages 174:1–174:11, New York, NY, USA, 2018. ACM.
- [52] Alison Kidd. The marks are on the knowledge worker. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 186–191. ACM, 1994.
- [53] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, and Sylvain Corlay. Jupyter notebooks—a publishing format for reproducible computational workflows. In *ELPUB*, pages 87–90, 2016.
- [54] Donald Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984.
- [55] Robert Kosara and Jock Mackinlay. Storytelling: The next step for visualization. *Computer*, 46(5):44–50, 2013.
- [56] J Richard Landis and Gary Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.
- [57] Christopher Le Dantec, Mariam Asad, Aditi Misra, and Kari Watkins. Planning with crowdsourced data: rhetoric and representation in transportation planning. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 1717–1727. ACM, 2015.
- [58] Jintae Lee and Kum-Yew Lai. What’s in design rationale? *Human-Computer Interaction*, 6(3):251–280, 1991.
- [59] Timothy Lethbridge, Janice Singer, and Andrew Forward. How software engineers use documentation: the state of the practice. *IEEE Software*, 20(6):35–39, November 2003.
- [60] Ian Li, Anind Dey, and Jodi Forlizzi. A stage-based model of personal informatics systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 557–566. ACM, 2010.

- [61] Chunyuan Liao, Francois Guimbretire, Ken Hinckley, and Jim Hollan. Papiercraft: A gesture-based command system for interactive paper. *ACM Transactions on Computer-Human Interaction*, 14(4):1–27, January 2008.
- [62] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela Byers. Big data: The next frontier for innovation, competition, and productivity. 2011.
- [63] Paul McBurney and Collin McMillan. Automatic documentation generation via source code summarization of method context. In *Proceedings of the 22nd International Conference on Program Comprehension*, pages 279–290. ACM, 2014.
- [64] Peter Brian Medawar. *Induction and intuition in scientific thought*, volume 22. Routledge, 2013.
- [65] Thomas Moran and John Carroll. *Design rationale: Concepts, techniques, and use*. L. Erlbaum Associates Inc., 1996.
- [66] Emerson Murphy-Hill, Chris Parnin, and Andrew Black. How we refactor, and how we know it. *IEEE Transactions on Software Engineering*, 38(1):5–18, 2012.
- [67] Nature. Announcement: Transparency upgrade for nature journals. <https://www.nature.com/news/announcement-transparency-upgrade-for-nature-journals-1.21627>, 2017.
- [68] Observable. Observable. <https://beta.observablehq.com/>, 2018. [Online; accessed 9-May-2018].
- [69] City of Chicago. City of chicago data portal. <https://data.cityofchicago.org/>, 2018. [Online; accessed 9-May-2018].
- [70] City of New York. Nyc open data. <https://opendata.cityofnewyork.us/>, 2018. [Online; accessed 9-May-2018].
- [71] Peter Parente. Estimate of public jupyter notebooks on github. <http://nbviewer.jupyter.org/github/parente/nbestimate/blob/master/estimate.ipynb>, 2018. [Online; accessed 9-May-2018].
- [72] Hilary Parker. Opinionated analysis development. *PeerJ PrePrints*, 2017.
- [73] Thomas Payne, Sarah Corley, Theresa Cullen, Tejal Gandhi, Linda Harrington, Gilad Kuperman, John Mattison, David McCallie, Clement McDonald, Tierney William Tang, Paul, Charlotte Weaver, Charlene Weir, and Michael Zaroukian. Report of the amia ehr-2020 task force on the status and future direction of ehRs. *Journal of the American Medical Informatics Association*, 22(5):1102–1110, 2015.

- [74] Roger Peng. Reproducible research in computational science. *Science*, 334(6060):1226–1227, 2011.
- [75] Fernando Perez. Personal communication. [From conversation on 7-Dec-2017].
- [76] Fernando Perez and Brian Granger. Project Jupyter: Computational Narratives as the Engine of Collaborative Data Science, July 2015.
- [77] Morgan Price, Bill Schilit, and Gene Golovchinsky. Xlibris: The active reading machine. In *CHI 98 conference summary on Human factors in computing systems*, pages 22–23. ACM, 1998.
- [78] Roman Rädle, Midas Nouwens, Kristian Antonsen, James Eagan, and Clemens Klokmoose. Codestrates: Literate computing with webstrates. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pages 715–725. ACM, 2017.
- [79] Carmen Reinhart and Kenneth Rogoff. Growth in a time of debt. *American Economic Review*, 100(2):573–78, 2010.
- [80] David Reinsel, John Gantz, and John Rydning. Data age 2025: The evolution of data to life-critical. *Dont Focus on Big Data*, 2017.
- [81] Daniel Russell, Mark Stefik, Peter Pirolli, and Stuart Card. The cost structure of sensemaking. In *Proceedings of the INTERACT’93 and CHI’93 conference on Human factors in computing systems*, pages 269–276. ACM, 1993.
- [82] Arvind Satyanarayan and Jeffrey Heer. Authoring narrative visualizations with ellipsis. In *Computer Graphics Forum*, volume 33, pages 361–370. Wiley Online Library, 2014.
- [83] Christopher Scaffidi, Mary Shaw, and Brad Myers. Estimating the numbers of end users and end user programmers. In *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*, pages 207–214. IEEE, 2005.
- [84] Donald Schon. *The reflective practitioner: how professionals think in action*, volume 1. Basic books New York, 1983.
- [85] Edward Segel and Jeffrey Heer. Narrative visualization: Telling stories with data. *IEEE transactions on visualization and computer graphics*, 16(6):1139–1148, 2010.
- [86] Edward Shortliffe and James Cimino. *Biomedical informatics*. Springer, 2006.
- [87] JW Smith. Joss: central processing routines. Technical report, RAND CORP SANTA MONICA CALIF, 1967.
- [88] James Somers. The scientific paper is obsolete. *The Atlantic*, 2018.

- [89] Giriprasad Sridhara, Emily Hill, Divya Muppaneni, Lori Pollock, and K Vijay-Shanker. Towards automatically generating summary comments for java methods. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*, pages 43–52. ACM, 2010.
- [90] Anselm Strauss and Juliet Corbin. *Basics of qualitative research: Grounded theory procedures and techniques*. Sage Publications, Inc, 1990.
- [91] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.
- [92] Aurélien Tabard, Wendy E Mackay, and Evelyn Eastmond. From individual to collaborative: the evolution of prism, a hybrid laboratory notebook. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pages 569–578. ACM, 2008.
- [93] Ming Tai-Seale, Cliff Olson, Jinnan Li, Albert Chan, Criss Morikawa, Meg Durbin, Wei Wang, and Harold Luft. Electronic health record logs indicate that physicians split time evenly between seeing patients and desktop medicine. *Health Affairs*, 36(4):655–662, 2017.
- [94] Craig Tashman. LiquidText: active reading through multitouch document manipulation. pages 2959–2962. ACM Press, 2010.
- [95] RStudio Team. Rstudio: integrated development for r. *RStudio, Inc., Boston, MA* URL <http://www.rstudio.com>, 2015.
- [96] James Thomas, editor. *Illuminating the path: the research and development agenda for visual analytics*. IEEE Computer Soc, Los Alamitos, Calif, 2005. OCLC: 750859580.
- [97] Matthew Tipping, Victoria Forth, Kevin J O’leary, David Malkenson, David Magill, Kate Englert, and Mark Williams. Where did the day go? a time-motion study of hospitalists. *Journal of hospital medicine*, 5(6):323–328, 2010.
- [98] Michele Tufano, Fabio Palomba, Gabriele Bavota, Rocco Oliveto, Massimiliano Di Penta, Andrea De Lucia, and Denys Poshyvanyk. When and why your code starts to smell bad. In *Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on*, volume 1, pages 403–414. IEEE, 2015.
- [99] John Tukey. *Exploratory data analysis*, volume 2. Reading, Mass., 1977.
- [100] Fernanda Viégas, Martin Wattenberg, and Kushal Dave. Studying cooperation and conflict between authors with history flow visualizations. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 575–582. ACM, 2004.

- [101] Fernanda Viegas, Martin Wattenberg, Frank van Ham, Jesse Kriss, and Matt McKeon. ManyEyes: a Site for Visualization at Internet Scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1121–1128, November 2007.
- [102] Robert Wachter. The digital doctor. *Hope, Hype and at the Dawn of Medicines Computer Age*, page 2015, 2015.
- [103] Lawrence Weed. Medical records that guide and teach. *N Engl J Med*, 278(11):593–600, 1968.
- [104] Hadley Wickham and Garrett Grolemund. *R for data science: import, tidy, transform, visualize, and model data.* " O'Reilly Media, Inc.", 2016.
- [105] Lauren Wilcox, Jie Lu, Jennifer Lai, Steven Feiner, and Desmond Jordan. Physician-driven management of patient progress notes in an intensive care unit. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1879–1888. ACM, 2010.
- [106] Wesley Willett, Jeffrey Heer, Joseph Hellerstein, and Maneesh Agrawala. Commentspace: Structured support for collaborative visual analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 3131–3140, New York, NY, USA, 2011. ACM.
- [107] Greg Wilson, D. A. Aruliah, Titus Brown, Neil Chue Hong, Matt Davis, Richard Guy, Steven Haddock, Kathryn Huff, Ian Mitchell, Mark Plumbley, Ben Waugh, Ethan White, and Paul Wilson. Best Practices for Scientific Computing. *PLoS Biology*, 12(1):e1001745, January 2014.
- [108] Greg Wilson, Jennifer Bryan, Karen Cranston, Justin Kitzes, Lex Nederbragt, and Tracy Teal. Good enough practices in scientific computing. *PLoS computational biology*, 13(6):e1005510, 2017.
- [109] Stephen Wolfram. *Mathematica: A System for Doing Mathematics by Computer.* Addison-Wesley, 1989.
- [110] Kai Zheng, Rema Padman, Michael P Johnson, and Herbert Diamond. An interface-driven analysis of user interactions with an electronic health records system. *Journal of the American Medical Informatics Association*, 16(2):228–237, 2009.