# Aiding Collaborative Reuse of Computational Notebooks with Annotated Cell Folding

Adam Rule, Ian Drosos, Aurélien Tabard, James D Hollan

# Aiding Collaborative Reuse of Computational Notebooks with Annotated Cell Folding

ADAM RULE, Design Lab, UC San Diego, USA
IAN DROSOS, Design Lab, UC San Diego, USA
AURÉLIEN TABARD, LIRIS, Université de Lyon, CNRS, France
JAMES D. HOLLAN, Design Lab, UC San Diego, USA

Computational notebooks aim to support collaborative data analysis by combining code, visualizations, and text in a single easily shared document. Yet, as notebooks evolve and grow they often become difficult to navigate or understand, discouraging sharing and reuse. We present the design and evaluation of a Jupyter Notebook extension providing facilities for annotated cell folding. Through a lab study and multi-week deployment we find cell folding aids notebook navigation and comprehension, not only by the original author, but also by collaborators viewing the notebook in a meeting or revising it on their own. However, in some cases cell folding encouraged collaborators to overlook folded sections or spend longer reviewing a notebook before editing it. These findings extend our understanding of code folding's trade-offs to a new medium and demonstrate its benefits for everyday collaboration. We conclude by discussing how dynamic reorganization can support sharing and reuse of computational notebooks.

CCS Concepts: • **Human-centered computing → Graphical user interfaces**; *Empirical studies in HCI*;

Additional Key Words and Phrases: Computational notebook; Cell folding; Jupyter Notebook

**ACM Reference Format:**
Adam Rule, Ian Drosos, Aurélien Tabard, and James D. Hollan. 2018. Aiding Collaborative Reuse of Computational Notebooks with Annotated Cell Folding. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW, Article 150 (November 2018), 12 pages. https://doi.org/10.1145/3274419

## 1 INTRODUCTION

Exploratory data analysis is an iterative process of extracting insights from data [6, 13, 14, 26]. As the scale and scope of data expand, this process has become increasingly collaborative [7, 13]. Consider a journalist who enlists several colleagues to sort through a collection of leaked documents, or a researcher who sends tissue 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 drastically different results. But clearly communicating complex analyses is remarkably difficult, especially when the analysis involves programming, which tends to produce large collections of scripts, comments, and results that can be difficult to manage or understand [6, 24].



Authors' addresses: Adam Rule, Design Lab, UC San Diego, 9500 Gilman Dr. La Jolla, CA, 92093, USA, acrule@ucsd.edu; Ian Drosos, Design Lab, UC San Diego, 9500 Gilman Dr. La Jolla, CA, 92093, USA, idrosos@ucsd.edu; Aurélien Tabard, LIRIS, Université de Lyon, CNRS, 25 avenue Pierre de Coubertin, 69622 Villeurbanne Cedex, Lyon, France, aurelien.tabard@univ-lyon1.fr; James D. Hollan, Design Lab, UC San Diego, 9500 Gilman Dr. La Jolla, CA, 92093, USA, hollan@ucsd.edu.



Version Date: August 28, 2018

One increasingly popular means of addressing these challenges is to conduct analyses in computational notebooks which combine code, visualizations, and text in a single interactive document (Figure 1). Computational notebooks aim to support collaborative data analysis by enabling analysts to present their work as computational narratives that can be easily shared and understood [19]. However, prior work examining over a million notebooks and interviewing dozens of notebook users revealed that many notebooks are loose collections of notes and scripts that even the original analyst struggles to understand [15, 22]. Rather than share their messy notebooks, 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 perform iterative analyses they have yet to realize their anticipated collaborative potential.

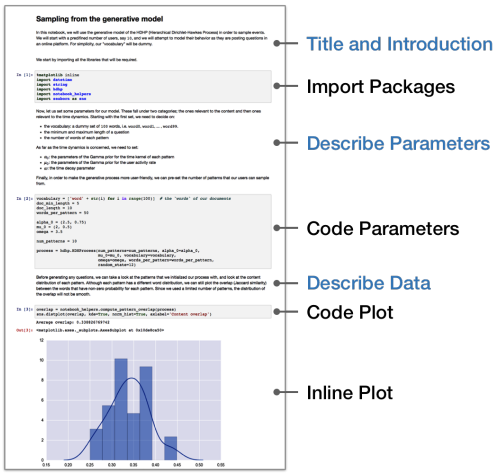


Fig. 1. Analysts can use computational notebooks to combine code, visualizations, and text into rich narratives for different audiences and contexts.

*This paper explores how to design computational notebooks to encourage clearer communication, freer sharing, and deeper engagement with complex data analyses.* It does so by asking how lightweight cell-folding and annotation might enable analysts to more easily craft notebooks that can be understood and reused in a variety of collaborative contexts. Through a lab study and multi-week deployment we find evidence that cell-folding aids both the original analysis and later reuse of notebooks for group presentations or continued analysis by a collaborator. However, as in prior work on code folding and document overviews, we find that folding cells can encourage collaborators to overlook folded sections or spend longer reviewing a notebook before editing it. These findings extend our understanding of code folding's benefits and trade-offs to a new medium while also demonstrating its collaborative benefits in both laboratory and real-world settings.

## 2 RELATED WORK

### 2.1 Data Analysis

In *Exploratory Data Analysis*, John Tukey described his subject as "looking at data to see what it seems to say" [26]. While vague, this definition captures the exploratory nature of data analysis which typically involves iterative cycles of obtaining, cleaning, profiling, analyzing, and interpreting data [6, 13]. As analysts explore data they try different versions of the same model or graph and routinely hit "dead ends" before generating an insight they feel "fits" the data.

As they iterate, analysts struggle to track which steps led to which results. Some tools help analysts track this provenance automatically [4, 21], though often in GUI-based analysis software rather than programming environments. Thankfully, research by Kery et al. is beginning to explore ways to support the types of lightweight versioning unique to programming for data analysis [14]. But much tracking remains manual and relatively few tools address how analysts organize and communicate the many byproducts of their work so others, or even their future selves, can audit, resume, or replicate it. As Guo [6] and Tabard [24] both note, complex analyses tend to produce large collections of similarly named files that can be difficult to navigate or understand. Many analysts don't want to take time to organize or annotate their code and outputs since many

explorations lead to dead ends and documenting each of these would break their train-of-thought. For these reasons complex data analyses easily devolve into poorly documented collections of files.

### 2.2 Computational Notebooks

One increasingly popular means of managing the many byproducts of programmatic data analysis is to perform analyses in computational notebooks. In the tradition of Knuth's literate programming [16], computational notebooks enable analysts to mix live code and visualizations with explanatory text. While available for decades in proprietary software, the recent emergence of several open source computational notebooks such as Jupyter Notebook and RNotebooks has enabled millions of scientists, journalists, and students to adopt them for a wide range of analytical tasks [5].

As described by the creators of Jupyter Notebook, the central aim of computational notebooks is to enable analysts to write rich computational narratives describing their analysis to a particular audience and context [19]. Indeed, some notebooks elegantly explain complex analyses and the paradigm of computational notebook is spreading. Distill uses a computational notebook format to explain complex machine learning algorithms [18], Codestrates demonstrates how computational notebooks can blur the lines between developing and using software [20], and ObservableHQ recently released a computational notebook for analyzing data on the web using JavaScript [10].

However, as two recent studies note, clear notebooks are the exception [15, 22]. Many lack explanatory text altogether, and even those that accompany academic publications rarely use text to discuss the reasoning that guided the analysis or interpret results. While notebooks provide tools for analysts to write rich computational narratives, analysts do not necessarily use them to great effect as they face a tension between continuing to explore data or pausing to explain their process. At the end of an analysis, many analysts are left with long and messy notebooks that even they struggle to understand, much less want to share with a collaborator.

### 2.3 Multi-Scale Interfaces for Working with Long Documents

Several interaction paradigms have been developed to aid navigation and comprehension of long documents by enabling users to simultaneously work at multiple levels of detail [3]. Two of the most common are overview + detail interfaces that present separate views of the same document at different levels of detail (e.g., slide navigators, mini-maps) and focus + context interfaces that mix different levels of detail in the same view (e.g., code folding, fisheye views).

Prior work has explored how these paradigms support reading and writing code or text. Several studies have shown that focus + context views help programmers perform some tasks more quickly, though not all, by aiding navigation [3, 12]. Other research on academic writing directly compared overview + detail, focus + context, and standard linear interfaces [8]. This work found that subjects wrote better essays when using an overview + detail interface but were slower answering direct questions, possibly because they spent additional time exploring the document overview even after finding the relevant answer. By contrast users of the focus + context interface read documents faster, but rarely viewed parts of the document that are not initially visible. Consequently they demonstrated less learning from the document. Overall these studies show that multi-level interfaces can aid completion of certain tasks, but may distract from the task at hand or discourage in-depth reading. It remains to be seen how they might support use of computational notebooks that combine executable code, explanatory text, and graphical results in a hybrid of previously studied media.

In the sections that follow we present the design and evaluation of a multi-scale interface for working with computational notebooks. This interface leverages manual cell-folding and annotation to aid and encourage notebook annotation and organization. In the studies that follow we seek to understand not only if this paradigm helps analysts navigate their own notebooks, but if it also encourages them to share notebooks and helps collaborators understand and reuse them.

## 3 DESIGN OF CELL-FOLDING FOR COMPUTATIONAL NOTEBOOKS

Prior work found that computational notebooks tend to become long, poorly formatted, and difficult to understand, hindering sharing and reuse [15, 22]. With this challenge in mind, we conducted two design workshops with seven graduate students (recruited in a convenience sample from a large public university) to brainstorm how we might help analysts craft notebooks that are easier to share, understand, and reuse. We recruited participants who had experience with both computational notebooks and design thinking [1] to ground brainstorming in prior experience and avoid common design pitfalls such as prematurely focusing on a single solution.

In an initial workshop we had participants brainstorm ways to make it easier for analysts to annotate, organize, and reflect on analyses in their notebooks. In a second workshop we worked with participants to paper prototype a subset of these ideas. In some cases, participants' ideas reflected features available in other writing and development environments (e.g., table of contents, file mini-maps, commenting), but other ideas were more specific to notebooks (e.g., stable and sequential cell numbering to aid navigation, parallel but linked presentation and scratch-pad notebooks). Based on participant feedback and prior research suggesting the value of code folding and document overviews, we chose to design and test a form of code-folding specific to computational notebooks that would encourage simultaneous code organization and textual annotation.

Jupyter Notebooks are linear collections of cells, that is, independent blocks of code or text. We developed Janus (Fig. 2), an extension to Jupyter Notebook, that enables analysts to fold cells in named sections. When users fold a cell or several consecutive cells, Janus renders a section header in their place. This header can be named and shows how many lines of code the folded

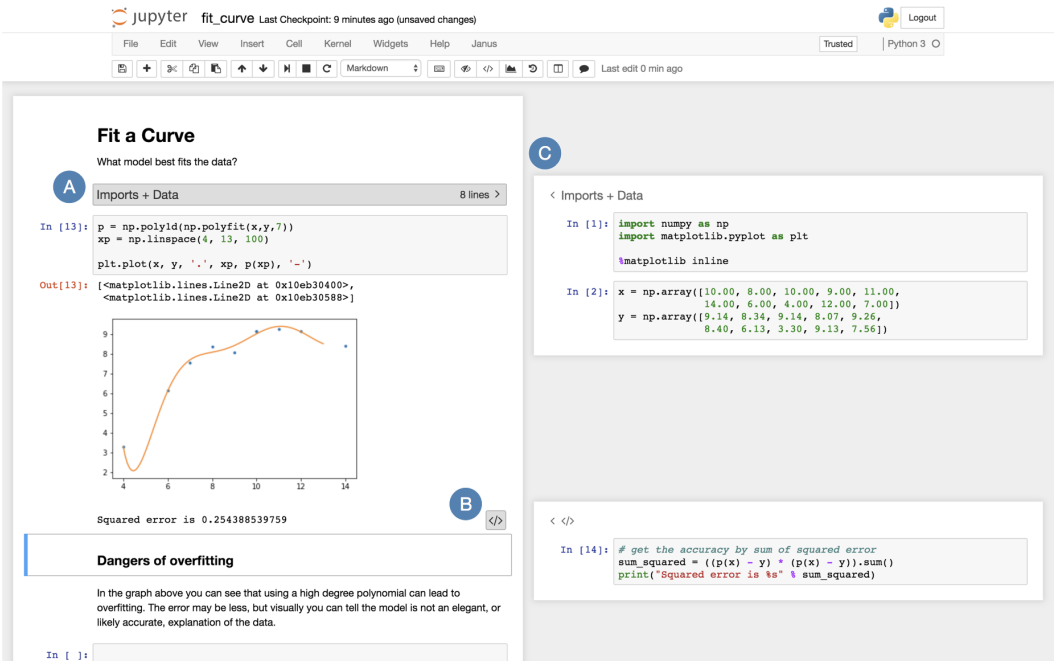


Fig. 2. We developed Janus, an extension to Jupyter Notebook, to explore the implications of adding cell folding to computational notebooks. Janus lets users fold groups of cells in named sections (A), or an individual cell's code or outputs (B). These hidden cells can be selectively shown in a sidebar (C).

cells contain. Hovering over the section header shows a miniaturized view of the hidden cells, making it easier to visually scan cells without unfolding them. When the section header is clicked, it reveals a second notebook panel to the right of the main notebook containing the hidden cells. We chose to unfold cells in a sidebar rather than inline so the main notebook could act as a stable notebook overview while the sidebar provided focused details, mixing the overview + detail and focus + context paradigms. Janus additionally lets notebook users fold just a code cell's input code or outputs so, for example, a graph can remain in the main notebook but the code used to generate it can remain hidden until needed.

Prior work suggests this cell folding should help analysts navigate long notebook files and complete certain analytical tasks more quickly [3, 8, 12]. We sought to understand if it would also aid sharing and reuse of computational notebooks.

## 4 STUDY 1: LAB STUDY WITH NOVICE ANALYSTS

With this first study we wanted to see if folding cells helps novice analysts navigate, understand, and extend a collaborator's notebook. When analysts share their notebooks, they often share them with less experienced programmers. A senior researcher may share a notebook with a junior colleague to introduce them to a line of research, or to share results with a colleague whose expertise is in biology, rather than computation. Due to this difference in experience, even when sharing within a lab or collaboration analysts can spend substantial time cleaning their notebooks to make them easier to understand [15]. To explore if cell folding might aid notebook navigation, comprehension, and reuse we conducted a study with 32 undergraduate data science students under the scenario that they were extending a lab-mate's analysis.

### 4.1 Participants and Methods

We recruited 34 students from an introductory undergraduate data science course at a large public university in the United States and gave each 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. To start, 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 extended the analysis to include San Jose, CA. The three tasks included 1) finding the unique ID that the dataset used to identify San Jose, CA, 2) plotting rental and home prices in San Jose over time, and 3) correlating changes in rental and homes prices in San Jose and the five original cities.

Sixteen participants used standard Jupyter Notebook in the control condition and sixteen used a version of Jupyter Notebook extended with Janus in the experimental condition. Participants in the experimental condition were first shown Janus' cell-folding features in an example notebook, and then worked with a folded version of the study notebook that was initially about 80% shorter than the control notebook due to folding 35 cells into 12 sections with names such as "Select Variables to Compare" and "Plot Rental Prices Over Time". The control notebook had the same sections marked with markdown headers. To encourage extensive navigation and engagement with the notebook, the study tasks were designed to be completed by reusing code already in the notebook.

We gave participants 30 minutes to complete as many of the 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. All materials and data for this and the next study are online at https://github.com/activityhistory/cscw2018.

### 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 current major. 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.

### 4.3 Results

We compare 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 we were more interested in how cell-folding changed how participants used notebooks than statistical significance.

*4.3.1 Proficiency:* All participants had experience with Python and Jupyter Notebook from their data analysis course, but there were significant differences between computing and non-computing majors 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 were robust enough to suggest computing and non-computing majors might use cell folding differently based on prior experience.

*4.3.2 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 post-study comprehension quiz associated with using Janus, pursuing a computing degree, or the interaction of these factors ( $p > 0.1$ in all cases) (Figure 3).

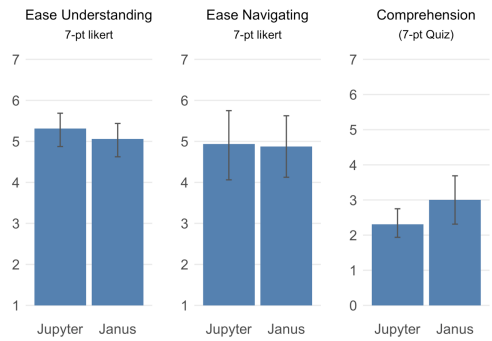


Fig. 3. There was no significant difference between conditions in self-rated ease of understanding or navigating the notebook, nor in notebook comprehension.

*4.3.3 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). A two-way ANOVA revealed that non-computing majors using Janus tended to take longer to review their notebook before starting the tasks than did other participants ($p = 0.09$, 95% CI [-69, 986] seconds longer). Note in Figure 4a that this difference seems largely driven by three outliers.

However there was also a reverse trend for non-computing majors using Janus to surprisingly take less time to complete the first two tasks (Figure 4b). We compare time on the first two tasks as

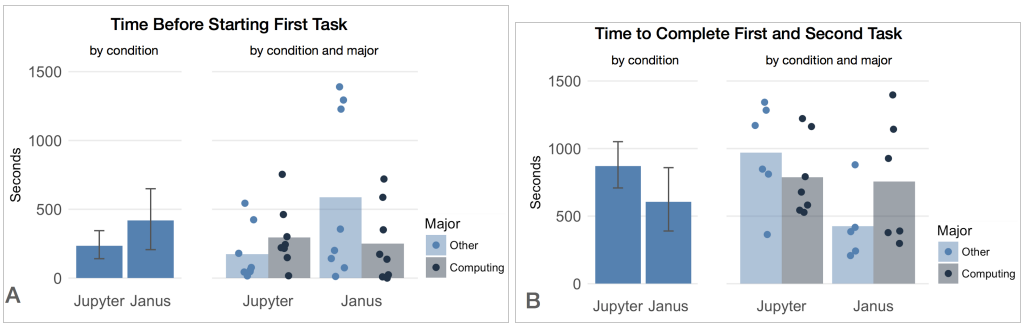


Fig. 4. (A) Non-computing majors using Janus tended to spend more time reviewing the notebook before starting the first task, though this was mostly due to three outliers. (B) However, Janus users also tended to take less time to complete Tasks 1 and 2, particularly if they were non-computing majors. Note how seven of the eight fastest times completing Tasks 1 and 2 were from Janus users.

only 13 of 32 participants were able to complete the third task in time, limiting any comparison across all tasks. 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.

*4.3.4 Qualitative Feedback:* Ten of the sixteen Janus users 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

However, five Janus users felt the folding hindered navigation at times or took getting used to:

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

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

Participants in the Janus condition also mentioned various ways to improve the extension. Four wanted the sidebar to be less "cramped", using a larger screen or showing folded cells inline with the main notebook. Three wanted the sidebar to follow them as they scrolled so they could simultanously view code from the top and bottom of the notebook. Finally, eight participants wanted better support for code or browser search (e.g., CMD - F) to aid navigating the notebook.

## 4.4 Discussion

Most participants who used Janus liked being able to fold cells. Conversely, many who used the standard notebook interface complained that the notebook was long and hard to navigate,

suggesting there are clear benefits to cell folding. Still, there may be times when analysts want to see the entire notebook or disparate sections side-by-side.

Remarkably, cell folding seemed to particularly help non-computing majors take less time to find and reuse code in the notebook, though it may have also encouraged some of them to take longer reviewing the notebook in the first place. As shown in Figure 4a, three non-computing majors using Janus took substantially longer to review the notebook before editing. These participants unfolded and re-ran each section of folded cells before they began their tasks. They may have found the folding unhelpful or, by breaking long analyses into sections, cell folding may have made the analysis seem more approachable and encouraged them to retrace each step. Either way, this result accords with prior work which found that people may continue reviewing a document after finding what they need if it has an overview pane, as the main notebook panel provided [8].

As for Janus enabling some users, particularly non-computing majors, to complete their tasks more quickly, screen recordings revealed that this may be due to adopting the strategy of making small edits to code in the middle of the notebook rather than copying it to the end of the notebook before editing. Five of the eight fastest participants (seven of whom were in the Janus condition) adopted this in-place editing strategy whereas only three of the remaining sixteen participants who completed Tasks 1 and 2 did so. Viewing code in smaller chunks may have given participants confidence that they understood the code and the effects of modifying it, so they were more likely to edit code in place. Computing majors may not have benefited as much from using Janus' code folding because even without it they may have been able to fall 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 of Figure 4 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. This result accords with prior work that found that fisheye or folding views can discourage readers from viewing parts of a document that are initially hidden [8].

Using Janus 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. While we found no significant increase in notebook comprehension or number of tasks completed, cell folding did help some novice analysts navigate and extend a collaborators' notebook more quickly, showing a benefit of cell folding for certain types of collaboration.

## 5 STUDY 2: TECHNOLOGY PROBE WITH EXPERT ANALYSTS

Our first study demonstrated that cell folding 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 cell folding differently in their own notebooks. To better understand how cell folding might support everyday collaboration we asked three expert analysts to use Janus for 2-4 weeks during their normal analytical work. We had three goals for this technology probe [9]: 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 notebooks.

### 5.1 Participants and Method

We recruited three experienced 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 and used it as their primary tool for data analysis.

We demonstrated Janus' cell folding to each participant using an example notebook and then asked each to organize the housing price comparison notebook from Study 1 as if they were going to share it with a colleague. Afterwards we asked them how they might have organized the notebook differently for themselves or a manager. Next, each participant installed Janus on their primary

work computer and used it for 2-4 weeks during their everyday analyses. We interviewed each participant about their experience with Janus after each week of use. Sample questions included:

- Can you open a notebook where you used Janus this week?
- How did you use, or try to use, Janus in this notebook?
- Did you show a notebook organized with Janus to anyone this week? How did they respond?

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

### 5.2 Results

*5.2.1 Social: Communicating with Self and Others.* Whether they were organizing 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:

> 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

> 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 organizing notebooks. By the time they were ready to share results with their lab, their notebooks were often too detailed and messy to present, so they would have to, reluctantly, create summary slide decks.

> 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

> 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, aiding both the ongoing analysis and group presentation. When working on an analysis, Janus helped our participants keep old code without it getting in the way of the current task. Participant 1 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. Participant 2 noted:

> What I would have normally have done was cut it out... 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 as to what analysis I was doing... so I can keep more of my thoughts, my own thoughts in there. - P2

Our participants also felt using Janus helped them reuse notebooks for group presentations, reducing the time spent preparing their notebook for sharing, helping them remember what they wanted to talk about, and giving them greater control over what was visible at any one time.

> 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 folded outputs] for me or a collaborator to say, "Okay, what is it that I'm looking for, what do I expect to see?". [And then unfold them] - P1

*5.2.2 Technology: Sidebar Unnecessary Technical Overhead.* As in Study 1, one of our participants mentioned that the sidebar felt small. He rarely viewed folded cells there, choosing instead to fold and unfold them in the main flow of his notebooks. While we expected more experienced analysts to use external monitors with more space for the sidebar, our participants still frequently edited notebooks on smaller laptop screens as they moved around campus to attend different meetings. 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.

*5.2.3 Design: Support for Navigation and Manipulation.* Participants were generally happy with Janus and mainly suggested minor tweaks rather than major changes to notebook software. However, our interviews, as well as observations from Study 1, highlight several use cases that computational notebooks could better support. Many of these echo issues explored in the active reading [17, 23, 25] or collaborative visual analytics literatures [7, 11], though they reflect the unique needs of analysts and their collaborators as they read and edit code and text in notebooks.

*Richer Visualization of Folded Cells*: Currently in Janus, folded cells are summarized by a user-defined labels, the number of lines of code they contain, and a small preview of the hidden cells when a user hovers over the section header. 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. 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 Notebook 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 views of the same notebook side by side. Alternatively, notebooks could let users collapse intermediate sections of a notebook (as in LiquidText [25]) 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, notebooks could enable users to save configurations of hidden cells so they can prepare and easily revert to specific configurations.

### 5.3 Discussion: Richer Navigation and Manipulation of Computational Notebooks

These findings suggest that 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, discouraging sharing and reuse. 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.

So far Janus supports a limited form of navigation, mainly helping analysts and their collaborators jump around long notebooks by folding sections they do not need 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 [25]. As with systems such as XLibris [23] and Papiercraft [17] 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 [2]. 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.

## 6 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 interactive reproducible analyses.

In this paper we explore how computational notebooks might be designed to encourage clearer communication by developing and testing Janus, an extension to Jupyter Notebook that adds cell folding and annotation to the notebook. Through a lab study with 32 undergraduate data science students and a multi-week field deployment with three more experienced analysts we find that cell folding supports not only the original analysis but also later reuse of notebooks both for group presentations and having a collaborator extend the analysis. However, there are tradeoffs involved with cell folding as it can encourage collaborators to overlook folded cells or dwell on exploring the document overview. These findings extend prior work on code-folding and document overviews by replicating previously identified benefits and trade-offs in a new medium (i.e., computational notebooks that mix code and text), demonstrating the collaborative benefits of cell-folding, and observing how it is used by individuals and teams in a real-world setting outside the lab.

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.

## ACKNOWLEDGMENTS

This research was funded by NSF grants #1319829 and #1735234.

## REFERENCES

[1] Richard Buchanan. 1992. Wicked Problems in Design Thinking. *Design Issues* 8, 2 (1992), 5. https://doi.org/10.2307/1511637

[2] Nicholas Chen, Francois Guimbretiere, and Abigail Sellen. 2012. Designing a multi-slate reading environment to support active reading activities. *ACM Transactions on Computer-Human Interaction* 19, 3 (Oct. 2012), 1–35. https://doi.org/10.1145/2362364.2362366

[3] Andy Cockburn, Amy Karlson, and Benjamin Bederson. 2009. A review of overview+ detail, zooming, and focus+ context interfaces. *ACM Computing Surveys (CSUR)* 41, 1 (2009), 2. https://doi.org/10.1145/1456650.1456652

[4] Susan B. Davidson and Juliana Freire. 2008. Provenance and scientific workflows: challenges and opportunities. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 1345–1350. https:

//doi.org/10.1145/1376616.1376772
[5] Brian Granger, Chris Colbert, and Ian Rose. 2017. JupyterLab: The next generation jupyter frontend. (2017).
[6] Philip Guo. 2012. *Software tools to facilitate research programming.* Ph.D. Dissertation. Stanford University.
[7] Jeffrey Heer and Maneesh Agrawala. 2008. Design Considerations for Collaborative Visual Analytics. *Information Visualization* 7, 1 (March 2008), 49–62. https://doi.org/10.1057/palgrave.ivs.9500167
[8] Kasper Hornbæk and Erik Frøkjær. 2003. Reading patterns and usability in visualizations of electronic documents. *ACM Transactions on Computer-Human Interaction (TOCHI)* 10, 2 (2003), 119–149.
[9] Hilary Hutchinson, Wendy Mackay, Bo Westerlund, Benjamin B. Bederson, Allison Druin, Catherine Plaisant, Michel Beaudouin-Lafon, Stéphane Conversy, Helen Evans, and Heiko Hansen. 2003. Technology probes: inspiring design for and with families. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 17–24. https://doi.org/10.1145/642611.642616
[10] Observable Inc. 2018. Observable. (2018). https://beta.observablehq.com/
[11] Petra Isenberg, Niklas Elmqvist, Jean Scholtz, Daniel Cernea, Kwan-Liu Ma, and Hans Hagen. 2011. Collaborative visualization: Definition, challenges, and research agenda. *Information Visualization* 10, 4 (Oct. 2011), 310–326. https://doi.org/10.1177/1473871611412817
[12] Mikkel R Jakobsen and Kasper Hornbæk. 2006. Evaluating a fisheye view of source code. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM, 377–386. https://doi.org/10.1145/1124772.1124830
[13] Sean Kandel, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. 2012. Enterprise Data Analysis and Visualization: An Interview Study. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (Dec. 2012), 2917–2926. https://doi.org/10.1109/TVCG.2012.219
[14] Mary Beth Kery, Amber Horvath, and Brad Myers. 2017. Variolite: Supporting Exploratory Programming by Data Scientists. ACM Press, 1265–1276. https://doi.org/10.1145/3025453.3025626
[15] Mary Beth Kery, Marissa Radensky, Mahima Arya, Bonnie E John, and Brad A Myers. 2018. The Story in the Notebook: Exploratory Data Science using a Literate Programming Tool. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 174. https://doi.org/10.1145/3173574.3173748
[16] Donald E. Knuth. 1984. Literate Programming. *Comput. J.* 27, 2 (Feb. 1984), 97–111. https://doi.org/10.1093/comjnl/27.2.97
[17] Chunyuan Liao, FranÃğois GuimbretiÃÍre, Ken Hinckley, and Jim Hollan. 2008. Papiercraft: A gesture-based command system for interactive paper. *ACM Transactions on Computer-Human Interaction* 14, 4 (Jan. 2008), 1–27. https://doi.org/10.1145/1314683.1314686
[18] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. 2018. The Building Blocks of Interpretability. *Distill* 3, 3 (March 2018). https://doi.org/10.23915/distill.00010
[19] Fernando Perez and Brian Granger. 2015. Project Jupyter: Computational Narratives as the Engine of Collaborative Data Science. (July 2015). https://blog.jupyter.org/project-jupyter-computational-narratives-as-the-engine-of-collaborative-data-science-2b5fb94c3c58
[20] Roman Rädle, Midas Nouwens, Kristian Antonsen, James R. Eagan, and Clemens N. Klokmose. 2017. Codestrates: Literate Computing with Webstrates. ACM Press, 715–725. https://doi.org/10.1145/3126594.3126642
[21] Eric D. Ragan, Alex Endert, Jibonananda Sanyal, and Jian Chen. 2016. Characterizing provenance in visualization and data analysis: an organizational framework of provenance types and purposes. *IEEE transactions on visualization and computer graphics* 22, 1 (2016), 31–40. https://doi.org/10.1109/TVCG.2015.2467551
[22] Adam Rule, Aurélien Tabard, and James D. Hollan. 2018. Exploration and Explanation in Computational Notebooks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.
[23] Bill Schilit, Gene Golovchinsky, and Morgan N. Price. 1998. Beyond paper: supporting active reading with free form digital ink annotations. ACM Press, 249–256. https://doi.org/10.1145/274644.274680
[24] Aurélien Tabard, Wendy E. Mackay, and Evelyn Eastmond. 2008. From individual to collaborative: the evolution of prism, a hybrid laboratory notebook. ACM Press, 569. https://doi.org/10.1145/1460563.1460653
[25] Craig Tashman. 2010. LiquidText: active reading through multitouch document manipulation. ACM Press, 2959. https://doi.org/10.1145/1753846.1753895
[26] John Wilder Tukey. 1977. *Exploratory data analysis.* Addison-Wesley Pub. Co, Reading, Mass.

Received April 2018; revised June 2018; accepted August 2018