Design and Evaluation of Spatialized 2D Computational Notebooks for Data Science

Jesse Maguire Harden

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science & Applications

Chris L. North, Chair
Doug A. Bowman
Donald S. McCrickard
Yalong Yang
Mahdi Belcaid

June 25, 2025
Blacksburg, Virginia

Keywords: Computational Notebooks, Nonlinearity, Data Science, Space to Think

Design and Evaluation of Spatialized 2D Computational Notebooks for Data Science

Jesse Maguire Harden

(ABSTRACT)

Computational notebooks are a popular tool for data science due to their ability to interleave text documentation, code, and results, including visuals, into a computational narrative. However, they have certain limitations which may be caused or exacerbated by the linear, top-down, 1D organization of cells within them. This dissertation explores and evaluates the potential of nonlinear computational notebooks to address issues with current linear computational notebooks, such as inefficient use of larger display spaces, and improve on the state of computational notebooks generally for data science. In this dissertation, we performed several studies aimed at exploring and evaluating the potential of spatialized 2D computational notebooks. We found in our first study several patterns of nonlinearity in data science work with computational notebooks and evaluated how problematic each pattern is. The second study found that users would utilize 2D space to organize computational notebook cells and discovered several patterns of organization. The third study showcased how comparative analysis can be more efficient and easier in well-designed spatialized 2D computational notebooks. The final study found that freeform spatialized 2D computational notebooks were seen as more usable than linear computational notebooks. Overall, our studies helped begin the work of expanding computational notebooks beyond their current linear mold.

Design and Evaluation of Spatialized 2D Computational Notebooks for Data Science

Jesse Maguire Harden

(GENERAL AUDIENCE ABSTRACT)

Computational notebooks, such as Jupyter Notebooks, are a popular tool for data science, the science of analyzing data for insights and creating predictive models to solve real-world problems, due to their ability to interleave text documentation, code, and results, including visuals, into a computational narrative, or a narrative of the computational work done to understand patterns in data and/or predict future results. Current computational notebooks enable users to create and organize cells, which contain either code and output of code, or text to help explain thought processes and more, in a linear, top-down, 1D organization; this linear organization may cause or exacerbate issues that computational notebook users have. This dissertation explores and evaluates the potential of nonlinear computational notebooks, computational notebooks which allow organization of cells in nonlinear ways, specifically in 2D (vertical AND horizontal), to address issues with current linear computational notebooks, such as inefficient use of larger display spaces, and improve on the state of computational notebooks generally for data science. The first study found several patterns of nonlinearity in data science work with computational notebooks and evaluated how problematic each pattern is. The second study found that users would utilize 2D space to organize computational notebook cells and discovered several patterns of organization. The third study showcased how comparative analysis can be more efficient and easier in well-designed spatialized 2D computational notebooks. The final study found that freeform spatialized 2D computational notebooks were seen as more usable than linear computational notebooks.

Dedication

To my biggest supporter and motivator, my wife, Camelia.

Acknowledgments

Personal Acknowledgments

To my mentors: To my advisor, Dr. Chris North, I am grateful for your guidance, support, and patience. You helped me to grow and see that I can do research and do it well. I am thankful for your mentorship.

To my colleagues: To the InfoVis lab, thank you for being there for me, for providing interesting ideas to think about and helping me refine my work. To the SAGE3 team, thank you for your support, your help, and your guidance on this journey.

To my family: Thank you for your patience and support as I pursued this journey of research.

Funding Sources

This research was partially supported by NSF Grant 2003387, SAGE3: Smart Amplified Group Environment for Harnessing the Data Revolution for Virginia Tech, a Walts Fellowship and Nance Fellowship through the Department of Computer Science at Virginia Tech, and a Summer Cunningham Fellowship through the Graduate School at Virginia Tech.

Attribution

Chapter 3 was conducted in collaboration with Dr. John Wenskovitch, Dr. Mahdi Belcaid, Dr. Nurit Kirshenbaum, Dr. Luc Renambot, Dr. Jason Leigh, and Dr. Chris North.

Chapter 4 is based on the paper titled "Exploring organization of computational notebook cells in 2d space" which was accepted for publication at IEEE VLHCC 2022 ©2022 IEEE. Reprinted from [22]. This work was conducted in collaboration with Elizabeth Christman, Dr. Nurit Kirshenbaum, Dr. John Wenskovitch, Dr. Jason Leigh, and Dr. Chris North.

Chapter 5 is based on the paper titled ""There is no reason anybody should be using 1D anymore": Design and Evaluation of 2D Jupyter Notebooks" which was accepted for publi-

cation at ACM Graphics Interface 2023. Reprinted from [23]. This work was conducted in collaboration with Elizabeth Christman, Dr. Nurit Kirshenbaum, Dr. Mahdi Belcaid, Dr. Jason Leigh, and Dr. Chris North.

Chapter 6 was conducted in collaboration with Dr. John Wenskovitch, Dr. Nurit Kirshenbaum, Dr. Luc Renambot, Dr. Jason Leigh, and Dr. Chris North.

Contents

List of Figures **xiv**

List of Tables **xvi**

1 Introduction **1**

1.1 Background and Motivation 1

1.2 Research Questions 4

1.3 Structure of Dissertation 5

2 Review of Literature **7**

2.1 Benefits of Larger Displays 7

2.1.1 Space to Think 7

2.1.2 Physical Navigation 8

2.2 Computational Notebooks 8

2.2.1 Issues with Computational Notebooks 9

2.3 Potential of Spatial Computational Notebooks 10

3 Pinpointing Problematic Patterns of Nonlinearity in Computational Notebook Work **13**

3.1 Introduction 13

3.2 Related Works . . . 15
3.2.1 Potential & Problems of Computational Notebooks . . . 15
3.2.2 Works on Improving Computational Notebooks . . . 16
3.3 Pinpointed Patterns of Nonlinearity . . . 18
3.3.1 Comparing Visualizations & Results . . . 19
3.3.2 Branching Code Paths . . . 19
3.3.3 Sectioning a Notebook . . . 20
3.3.4 Prioritizing Items by Importance . . . 20
3.3.5 Gathering Visualizations & Results into Dashboards . . . 21
3.3.6 Interacting Nonlinearly with Interactive Visualizations . . . 21
3.3.7 Modularizing Code . . . 22
3.3.8 Controlling Cell Versions . . . 22
3.3.9 Managing Nonlinear Execution Patterns . . . 23
3.4 Survey Methodology . . . 23
3.4.1 Data Analysis Process . . . 24
3.5 Survey Results . . . 26
3.5.1 Demographic Results . . . 26
3.5.2 Likert-Scale Results . . . 27
3.5.3 Qualitative Results . . . 28
3.6 Discussion . . . 40

3.6.1 Evaluating the Patterns . . . 40
3.6.2 2D Nonlinear Notebooks & the Pinpointed Patterns . . . 41
3.6.3 Accessibility Concerns for Nonlinear Notebooks . . . 42
3.6.4 Nonlinearity & Prior Literature on Computational Notebook Issues . . . 43
3.6.5 Limitations . . . 44
3.7 Conclusion . . . 44

4 Exploring Organization of Computational Notebook Cells in 2D Space 46

4.1 Introduction . . . 46
4.2 Background and Related Works . . . 48
4.2.1 Computational Notebooks . . . 48
4.2.2 Computational Narratives . . . 49
4.2.3 Space to Think . . . 49
4.3 Methodology . . . 50
4.3.1 User Study Task . . . 50
4.3.2 Post-Task Survey and Optional Interview . . . 51
4.4 Results . . . 51
4.4.1 User Study Task Results . . . 51
4.4.2 Post-Task Survey Likert-Scale Results . . . 55
4.4.3 Qualitative Survey Questions and Interview Results . . . 56

4.5 Discussion . . . 57
4.5.1 Organizing Notebooks in 2D . . . 58
4.5.2 Enabling Strengths of 2D Notebooks . . . 58
4.5.3 Addressing Tradeoffs of 2D Notebooks . . . 59
4.5.4 Assessing Interest in 2D Notebooks . . . 60
4.5.5 Limitations . . . 60
4.6 Conclusion . . . 61

5 Design and Evaluation of 2D Jupyter Notebooks with the Multi-Column Organizational Pattern 62

5.1 Introduction . . . 62
5.2 Background and Related Works . . . 64
5.2.1 Computational Notebooks . . . 65
5.2.2 Space to Think . . . 66
5.3 Design of 2D Jupyter Notebook Extension . . . 67
5.4 Study 1 Methodology . . . 68
5.4.1 Recruitment and Screening . . . 69
5.4.2 Hardware for User Study . . . 69
5.4.3 Task Designs and Rationales . . . 69
5.4.4 Survey Questions Design . . . 74
5.4.5 Data Analysis Process . . . 75

5.5 Study 1 Results . . . 76
5.5.1 User Interaction Strategies . . . 77
5.5.2 Efficiency Measurements . . . 79
5.5.3 Survey Questions . . . 80
5.5.4 Scrolling Time . . . 82
5.6 Discussion . . . 82
5.6.1 Task Efficiency Benefits . . . 82
5.6.2 Usability Benefits . . . 83
5.6.3 Design Challenges & Opportunities . . . 83
5.6.4 Limitations . . . 84
5.7 Conclusion . . . 85

6 Exploring the Potential of 2D Freeform Computational Notebooks with SAGECells 92

6.1 Introduction . . . 92
6.2 Overview of SAGECells in SAGE3 . . . 94
6.3 Study Methodology . . . 95
6.3.1 Recruitment and Screening . . . 95
6.3.2 Hardware for User Study . . . 96
6.3.3 Task Design and Rationale . . . 97
6.3.4 Survey Questions Design . . . 99

6.3.5 Data Analysis Process . . . 100
6.4 Results . . . 103
6.4.1 Strategies and Layouts Used . . . 103
6.4.2 Performance Metrics . . . 106
6.4.3 Survey Questions . . . 109
6.5 Discussion . . . 110
6.5.1 How Affordances Affect Layouts . . . 111
6.5.2 Efficiency in Linear vs. Nonlinear Notebooks . . . 112
6.5.3 Usability Benefits of Nonlinear Notebooks . . . 112
6.5.4 Design Challenges and Opportunities . . . 113
6.5.5 Limitations . . . 114
6.5.6 Future Work . . . 115
6.6 Conclusion . . . 116

7 Conclusion **118**

7.1 Conclusions . . . 118
7.1.1 RQ 1: What problems with current computational notebooks are due to their linear structure, which could feasibly be addressed with spatialized nonlinear computational notebook designs? . . . 118
7.1.2 RQ 2: How would users organize computational notebook cells in spatialized 2D notebooks? . . . 119

7.1.3 RQ 3: What potential benefits could spatialized nonlinear computational notebooks which implement a multi-column structure provide for data science over current linear notebooks? . . . 120

7.1.4 RQ 4: What potential benefits could spatialized freeform nonlinear computational notebooks provide for data science over current linear notebooks? . . . 121

7.2 Future Work Directions . . . 122

7.2.1 Scalability . . . 122

7.2.2 Creating and Managing Run Order . . . 123

7.2.3 User Studies with Experts in Data Science . . . 125

7.2.4 Collaborative Scenarios . . . 126

7.2.5 Utilizing More Objective Measures . . . 126

7.2.6 Delving Deeper with Qualitative Analysis . . . 127

7.2.7 Exploring Whether and How Users Think About Organizing Cells in 2D Notebooks . . . 128

7.2.8 Effects of Available Screen Space and its Usage . . . 129

7.2.9 Human-Machine Teaming . . . 130

Bibliography **132**

List of Figures

3.1 Visual Depiction of the Patterns 18

3.2 Example Nonlinear Notebook from the Survey 24

3.3 Heatmap showcasing responses, including mean and median response, to how problematic each pattern is. 25

4.1 Minimaps of User Study Task Results 52

4.2 Example Linear design pattern with 3 split cells. 52

4.3 Example Multi-Column design pattern with parallel counties. 53

4.4 Example Workboard approach using a Directed Acyclic Graph. 54

5.1 Finding & Comparing Results 2D Notebook from Study 1 62

5.2 Notebook Controls for 2D Jupyter extension 63

5.3 Parameter Tuning 2D Notebook 72

5.4 Code Comparison 2D Notebook Clip 87

5.5 A heatmap comparing the ratings for the Post-1D and Post-2D questions. . 88

5.6 A bar chart showing average time to completion by task and layout in seconds. 88

5.7 A bar chart comparing the mean ratings for the Post-1D and Post-2D questions; positive values indicate agreement with the sentiment, while negative values indicate disagreement. 89

5.8 A heatmap visualizing the ratings for the Post-Tasks questions. 89

6.1 SAGECell Application containing the code section, the output section, the kernel and font size selection, and the execution control on the side. 95

6.2 Starting Layout for the 2D SAGECells condition; the top row is data preprocessing and the column is the first branch. 96

6.3 Example Multi-Column layout. 104

6.4 Example Grouped Combinations layout. 105

6.5 Heatmap of Preferred Layout for Various Data Science Tasks. 107

6.6 Heatmap of Agreement Level with Statements (Pro-1D statements first and then Pro-2D statements). 107

6.7 Heatmap of Preferred Layout for Various Data Science Project Types. 107

6.8 Boxplots of subjective usability measure results by layout (1D or 2D). 108

List of Tables

3.1 Can 2D Notebooks Address the Patterns? 27

4.1 2D Notebook Survey results with Likert-scale frequencies 56

4.2 Qualitative Themes in Survey & Interview Results regarding 2D Notebooks . 57

5.1 P-values for Scheirer-Ray-Hare by Task and Effect 79

5.2 Post-2D minus Post-1D Average Differences in Rating 80

5.3 Qualitative Themes in Study 1 Survey 90

5.4 Scroll Event Analysis Totals Across All Participants 91

6.1 Strategies and Layouts Used in 2D Notebook 103

6.2 Strategies and Layouts Used in 1D Notebook 104

6.3 Pre-Filtering Time to Completion Results 106

6.4 Post-Filtering Time to Completion Results 106

6.5 One-Tail Paired T-Test on Accuracy Results 106

6.6 Post-2D - Post-1D *p-values* and **Differences** 109

6.7 Qualitative Themes in Study Surveys 117

List of Abbreviations

1D 1-dimensional

2D 2-dimensional

1D means 1-dimensional, such as a program that organizes blocks or cells in either a horizontal or vertical direction, but not both.

2D means 2-dimensional, such as a program that organizes blocks or cells in both horizontal and vertical directions.

Chapter 1

Introduction

1.1 Background and Motivation

Computational notebooks, such as Jupyter Notebooks, are a popular tool for data science [55]. With code cells, which generally have an input portion for writing code and an output portion for showing results like printed values and visualizations, and markdown cells for writing text, such as explaining one's exploratory processes, noteworthy findings, and conclusions, computational notebooks provide tools for both exploratory work and explanatory presentation. Through these 2 cell types, users can interleave code, text, and results to create computational narratives that can not only be read, but also reproduced.

Current computational notebooks, such as Jupyter Notebooks, restrict the arrangement of cells to a top-down, 1D format. While this linear arrangement can work well for linear analyses and narratives, many narratives and analyses are nonlinear. A nonlinear analysis or narrative, as opposed to a linear analysis or narrative, does not follow a single sequential progression of steps; it can include steps done in parallel, as well as jumping back and forth between steps. Comparative analyses, or analyses involving comparison of multiple visuals and/or printed (numeric or text) results, are one common nonlinear analysis in data science. Analysts often jump to and from different results and back to earlier results to compare, contrast, and make decisions. Indeed, data science as a process has a great deal of nonlinearity in its progression; analysts explore their data and different options or algorithms

to perform tasks, and often jump back and forth between cleaning, exploring, and analyzing. In current computational notebooks, their linear structure for organizing cells clashes with the nonlinearity of much data science work, often resulting in exploration of data and explanation of results being at odds [62]. This tension can cause and/or exacerbate issues such as messiness [25] and an inability for others to reproduce the results of a completed computational notebook [56].

Another disadvantage of current linear computational notebooks is the fact that they do not make effective use of available screen screen, especially for larger display setups such as widescreen monitors and multiple-monitor setups. Larger screen setups can provide more external memory and the ability to encode meaning spatially, the core benefits of Space to Think [2]. Just as writing notes on a large sheet of paper (e.g. for a test) enables users to put more information down and organize the information better when compared to a small notecard, so too does having more available screen real estate provide such benefits. In addition, larger screen setups can enable physical navigation, such as turnings one's head, to replace virtual navigation, such as scrolling and switching tabs, resulting in more efficient performance on tasks [1]. By restricting cell organization to 1 dimension (1D), organized top-to-bottom, current computational notebooks can struggle to use the horizontal length of available screen space well, especially if code line are generally not as long as the screen, and thus fail to empower users to make the most of their screen space.

Given the nonlinear nature of much data science work, how such work benefits from larger displays, and how such work can produce nonlinear narratives, enabling nonlinear organization of computational notebook cells, or *spatialized* computational notebooks, could help address some of the issues affecting the usability of computational notebooks as found in prior works [11, 62].

As a simple example, consider an analyst, Christine, who wants to learn how different states

and counties in the United States were affected by COVID-19 in terms of cases, deaths, business costs, and more. In her Jupyter Notebook, Christine gathers data from various sources, joins tables, and begins to analyze various different metrics through visualizations. To get a more holistic perspective of the data by comparing different visualizations, she has to constantly scroll up and down since she can't fit all the visualizations on her screen at once. With a spatialized 2D computational notebook, Christine could arrange the visualizations on her screen in a way that minimizes, if not eliminates, the need to scroll during her analysis.

Later in her analysis, Christine decides to drill down to several particular counties of a particular state; she plans to run the same analyses and visualization code on each county so that she can compare the results. Unfortunately, her Jupyter Notebook, with its linear structure, does not allow her to organize these subsections of her analysis in such a way that relates to her mental model of how her analysis is structured, not to mention the difficulty in comparing results between sections due to needing to scroll. With a spatialized 2D computational notebook, Christine could arrange the sections in a way that better represents her mental model, such as putting the subsections of her analysis side-by-side, and enables her to perform the comparisons she wishes to do with ease.

While a few recent works outside of this dissertation, such as Fork-It [75] and Stickyland [73] have begun exploring ways to improve on the current state of computational notebook design through enabling some level of spatialized nonlinear organization, a more thorough exploration of potential designs and benefits of such computational notebooks has been missing. This dissertation seeks to explore the potential of spatialized nonlinear 2D computational notebooks, such as what problems they could address, how users might prefer to organize cells in such notebooks, and what benefits they could bring to data science work.

1.2 Research Questions

RQ 1: What problems with current computational notebooks are due to their linear structure, which could feasibly be addressed with spatialized nonlinear computational notebook designs?

Prior work on problems with computational notebooks have focused on a variety of pain points [11] for users and how exploration and explanation are at odds [62], but none have focused on how the nonlinearity inherent in data science work can cause or exacerbate issues with computational notebooks. To address this research question, we conducted a literature review to identify patterns of nonlinearity and a survey study to evaluate and validate the patterns.

RQ 2: How would users organize computational notebook cells in spatialized 2D notebooks?

Given that nonlinear computational notebooks could address certain patterns of nonlinearity found in computational notebook work, one must then question how users would prefer to organize cells in 2D and, indeed, if they even would utilize 2D space to organize cells. To understand how future nonlinear computational notebooks might be used and designed, we conducted a user study where participants arranged a set of images of cells, both input and output together, on a 2D canvas and explained their designs in a post-task survey. Using the findings from this study, we propose 3 overarching design patterns, 2 of which have sub-patterns, for organizing computational notebook cells in 2D.

RQ 3: What potential benefits could spatialized nonlinear computational notebooks which implememnt a multi-column structure provide for data science over current linear notebooks? Given that the multi-column pattern was commonly used by participants in the exploratory organizational study, we sought to examine whether

a well-structured multi-column notebook could improve usability and efficiency, especially as it relates to the Comparing Visualizations & Results pattern of nonlinearity. Thus, we designed and implemented an extension for Jupyter Notebooks, 2D Jupyter, which we then used to statistically test our hypotheses. We found that usability and efficiency are improved in a well-structured multi-column notebook.

RQ 4: What potential benefits could spatialized nonlinear computational notebooks which are freeform, or have no set structure such that the user must decide on, design, and implement a structure, provide for data science over current linear notebooks? Given the prior work showing usability and efficiency gains for a spatialized 2D notebook with the multi-column style, we sought to generalize this work to freeform, or user-imposed structured, spatialized nonlinear computational notebooks in 2D. In particular, we sought to test whether the benefits to efficiency and usability held when using such a freeform 2D notebook for a task in which users had to make their own arrangement of the cells mostly from scratch. We utilized SAGECells in SAGE3 as the freeform 2D computational notebook, which we compared with regular Jupyter Notebooks to test our hypotheses. We found that efficiency was similar to Jupyter Notebooks and subjective perceptions of usability increased for the freeform 2D notebook.

1.3 Structure of Dissertation

Chapter 1 introduces the background and motivations for this work's goals and presents the overarching research questions of this work.

Chapter 2 presents a literature review of foundational works in computational notebooks, Space to Think, and computer science education.

Chapter 3 addresses Research Question 1, as well as partially addressing Research Question 3, by presenting a literature review and survey study to understand and evaluate patterns of nonlinearity in computational notebook work. (Submitted)

Chapter 4 addresses Research Question 2 in presenting a user study exploring how users would organize computational notebook cells in 2D. (Completed)

Chapter 5 addresses Research Question 3 in presenting a quantitative user study testing whether a well-structured nonlinear computational notebook could provide efficiency and usability benefits over a similar, well-structured linear computational notebook. (Completed)

Chapter 6 addresses Research Question 4 in presenting a quantitative and qualitative user study testing whether a freeform (or user-imposed structure) 2D nonlinear computational notebook could provide efficiency and usability benefits over a linear computational notebook. (Completed)

Chapter 7 summarizes the contributions of this dissertation, goes over lessons learned, and elucidates potential future work.

Chapter 2

Review of Literature

The concept of spatializing computational notebooks and utilizing a whiteboard approach for higher education draws on two key sources of inspiration: the benefits provided by larger display setups, such as Space to Think [2] and physical navigation [1], and the issues present with current computational notebooks that affect their potential as data science and analysis tools.

2.1 Benefits of Larger Displays

Larger display setups, such as multi-monitor setups, widescreen displays, and tiled wall displays, enable certain benefits beyond that of which smaller screens, such as laptop screens, can supply. These benefits include Space to Think [2], physical navigation [1], along with benefits to collaborative work [7, 37].

2.1.1 Space to Think

Andrews, Endert, and North [2] focused on large, high-resolution displays for sensemaking, and found additional space aided users in two ways: it enabled externalized memory, letting users focus on the task at hand rather than on recalling important info, and it enabled encoded meaning into space, such as by clustering similar items together. That work has

been expanded through study of additional space provided by virtual reality [43], as well as collaborative uses of large spaces through increased and varied content contribution [37]. Similar works demonstrating benefits of space to programming tasks include Code Bubbles [8] and VisSnippets [9]. All of these works demonstrate how effective use of space can open up new possibilities for analytical work.

2.1.2 Physical Navigation

As Andrews and North found in their 2013 work [1], enabling physical navigation, such as turning one's head, through larger display spaces can help users utilize spatial thinking in addition to the other benefits provided by a large display. Their work is situated in earlier work by Kirsh [36] on utilizing external representations, which is related to the idea of embodied cognition. Kirsh claims that utilizing external representations, as opposed to internal representations, "may increase the efficiency, precision, complexity, and depth of cognition" for various tasks. In a similar vein, Andrews and North [1] found that larger display spaces which enable physical navigation methods may provide much the same sort of benefits over spaces which require virtual navigation methods, such as scrolling and changing tabs.

2.2 Computational Notebooks

Computational notebooks, influenced by Knuth's *literate programming* [39] concept where authors weave "human language with live code and the results of the code" to produce a computational narrative [54], support incremental and iterative analysis, explanation of an analyst's thoughts and processes, and sharing of code, text, and visuals in one document

[62]. These affordances can contribute to scientific endeavors through making it possible to reproduce the analyses done if one has access to the requisite data or a similarly-gathered dataset.

2.2.1 Issues with Computational Notebooks

However, Chattopadhyay et al. [11] found users struggle with computational notebooks. One struggle with the iterative process of exploration and analysis is messiness [34, 47, 62]. Analysts described notebooks and their code as "ad hoc" or "throw-away" [25, 31] and in need of cleaning [62] before presentation. Kery et al. [34] found data analysts, as part of the process of exploring alternatives, replicate code across many cells that must later be refactored, a process both tedious and error-prone [34, 75]. In short, exploration and analysis can complicate constructing clean computational narratives.

Messiness in Computational Notebooks

Head et al. [25] showed messiness can come from disorder, deletion, and dispersal, where disorder means run order and presentation order are different, deletion means overwriting or deleting necessary code, and dispersal means related cells are far apart. Many tools have been developed to help deal with messiness, from Head et al.'s work [25], to cell dependency graph visualization [76] to version control systems for computational notebooks [32, 33]. This dissertation does directly touch on messiness as a problem potentially exacerbated by the linear, 1D structure of current computational notebooks and as a concern that may influence discussion around how much structure spatial computational notebooks should have.

Computational Notebooks and Reproducible Science

Reproducible research is an important and challenging issue for any scientific endeavor, and research in HCI and computer science is no different [5, 12, 16]. At their best, computational notebooks and the computational narratives formed using them enable reproducible scientific workflows [4, 38, 63]; the ability to interleave documentation with code and results contributes to this potential. However, computational notebooks in the wild are rarely reproducible [56]; issues such as messiness [25], out-of-order execution [56], and dependency issues contribute to the pain of trying to reproduce computational notebook findings [11]. Indeed, less than 5% of notebooks studied by Pimentel et al. [56] were reproduced with the same results. Work to address issues with reproducibility, such as Osiris by Wang et al. [72], has helped, however. While this dissertation does not focus directly on reproducibility of computational notebooks, spatial notebooks may enable more thoughtful management of code cells in a way that may hopefully combat messiness.

2.3 Potential of Spatial Computational Notebooks

In addition to potentially enabling Space to Think and other benefits provided by larger display setups, spatializing computational notebooks has already found some success in the work of Weinman et al. on Fork-It [75]. While their extension was initially created as a way to test different analytical forks in a compact, comparable way within a computational notebook, the ability to create forks was used for more than just that; users used the split column structure to organize code and contain messes, among other things.

Another similar work on spatializing code is Code Bubbles [8], which uses the *bubble* metaphor in integrated development environments (IDEs); a bubble is an interactive, editable frag-

ment of items like code or documentation. Bubbles can be clustered together to represent a "concurrently visible working set." Their work found similar benefits from space in Code Bubbles as Andrews, Endert, and North [2] found for large displays. Specifically, Bragdon et al. [8] found that developers appreciated being able to externalize information from their "limited working memory" through "bubbles and annotations." They also found that the ability to position bubbles strategically made developers more efficient through the use of "spatial proximity and spatial memory."

VisSnippets [9] is designed for collaborative data exploration using blocks of reusable code ("snippets") connected by arrows to form a 2D visual dataflow display; it empowers quick exploration of "complementary and contrasting analyses" through reuse of snippets a group has made over time.

Spatialized notebooks have also been explored recently in 3D using virtual reality in recent works by Sungwon In et al. [27], with a system for special branching and merging interactions designed to make it easier to code and compare different branches of code. This work compared a 2D desktop setting with 3D virtual reality, both with and without the special branching and merging feature. In et al. [27] found 3D space enhanced navigation and the branching and merging feature enhanced comparison.

In addition to research works, the idea of performing analysis in a spatialized environment has become more popular in business recently with the acquisition of Einblick.ai, a canvas with low-code and no-code data science tools, by Databricks [19], and the upcoming Observable Canvas tool [6]. These tools both utilize an infinite canvas with tools for quickly visualizing and analyzing data, with the workflows representable in a nonlinear, spatialized manner, showcasing how spatialized nonlinear computational notebook-like environments can address real-world issues in more than just research works.

These works demonstrate the potential of spatialization to address more than just comparative analysis problems with current computational notebooks, and motivates much of the work done in this dissertation.

Chapter 3

Pinpointing Problematic Patterns of Nonlinearity in Computational Notebook Work

3.1 Introduction

Computational notebooks like Jupyter [30, 38] are popular tools for exploration and explanation in data science. Through interleaving text, code, and outputs, like visual graphs, computational notebook users can construct, refine, and present computational narratives [55, 62, 66]. However, current computational notebooks, with their linear, top-down one-dimensional (1D) structure, may make nonlinear analyses and narratives more tedious and harder to perform, and do not make efficient use of larger and widescreen displays; this may be due in part to the lack of support for nonlinear arrangements of their cells.

Earlier work found nonlinear notebooks helpful for data science. 2D Jupyter [23] increased efficiency and usability by arranging multiple columns of cells in a row, or a multi-column style per exploration of two-dimensional (2D) arrangements of cells by Harden et al. [22]. Fork-It [75] improved debugging and exploration in addition to unexpected benefits like, containing messes, using forks. Stickyland [73] introduced "sticking" cells to a persistently

visible dock at the top of a notebook, like freezing the top row in an Excel spreadsheet to keep it in view regardless of scrolling. These works shows how breaking the linearity of computational notebooks can benefit users.

More research is needed to identify which issues linear computational notebooks have that nonlinear computational notebooks should address. This work contributes to computational notebook research by identifying several patterns of nonlinearity in computational notebook work for data science and verifying which issues are most problematic from user perspectives. We focus on the following research questions:

- RQ1: What nonlinear patterns emerge in data science and analysis work with computational notebooks?

- RQ2: How problematic is each pattern, and why?

- RQ3: Do users see nonlinear computational notebooks as having the potential to address these problems?

To answer these questions, we did a literature review, used affinity diagramming, and reflected on our own usage of computational notebooks to identify patterns of nonlinearity; then we designed and conducted a survey study to determine how problematic each pattern is. Participants saw the patterns we identified as moderately problematic in linear, 1D computational notebooks and had cautious optimism about nonlinear computational notebooks' potential to address these issues.

3.2 Related Works

This work builds on computational notebook research on problems with current computational notebooks and on nonlinear approaches to organizing computational notebook cells.

3.2.1 Potential & Problems of Computational Notebooks

Computational notebooks can enable exploratory programming, good for data science and analysis [31, 62], and explanatory computational narratives showing work done and justifying conclusions drawn, good for reproducible science [4, 38, 63]. Through interleaving markdown text cells, code cells, and outputs like visualizations, computational notebook users can document progress during analysis and weave a computational narrative of their thought processes and decisions.

However, computational notebooks have pain points [11]; issues include messiness [25, 34, 47, 62], non-linear analyses and narratives [62], and navigating longer notebooks [22]. Users often find exploration and explanation at odds in computational notebooks [62], which leads to issues like the above and to notebooks that are not reproducible. In the wild, notebooks are rarely reproducible; less than 5% of notebooks studied by Pimentel et al. [56] were reproducible. In addition, users with widescreen displays, multiple monitors, or other large screen space environments that can provide more externalized memory and semantic encoding of meaning in the space, or Space to Think [2], cannot make efficient use of their display space with current computational notebooks without tedious workarounds such as opening the same notebook multiple times. One reason for these issues may be the linear, 1D arrangement of cells in current computational notebooks.

3.2.2 Works on Improving Computational Notebooks

Here we discuss two types of improvements to computational notebooks: those which stick to the 1D, linear organization of current computational notebooks, and those who showcase the potential of breaking out of the linear mold.

Improvements to Linear Computational Notebooks

Since the popularization of computational notebooks, there has been a good deal of work done to improve them, especially in areas such as version control and interactive visualizations.

Kery, Horvath, and Myers helped users deal with version control through Variolite [33], which enabled users to perform local versioning by creating "variant boxes" to contain variants of a code cell within 1D notebooks; virtual navigation between variants by switching was needed. Kery and Myers followed this up with Verdant [32], an updated version control system with interactions to explore code history; they found multiple uses for such a system, from comparing outputs to backtracking to an earlier version. Head et al. [25] also made a tool that enables users to find, gather, and recover code that is necessary to produce a given output. While these tools help address version control issues, they tend to rely on virtual navigation methods, which are less efficient than physical navigation [1].

Interactive visualizations enable quicker discovery of relationships between variables [77], yet computational notebooks and interactive visualizations do not always mix well. One issue here is taking visual exploration findings and using them in code; both B2 [78] and AnyWidget [49] address this. While they improve usability of interactive visualizations in computational notebooks, Wang et al. [74] note that many visualization tools must choose between putting the tools within the 1D, linear flow or outside of it; they note that exploring "alternative notebook layouts," such as nonlinear computational notebooks like Stickyland

[73], could help address this tradeoff.

Other works on improving computational notebooks within the 1D paradigm include using graph visualizations to help users better understand and keeping track of dependencies between cells [76], exploration of the role synchronous editing can play in enabling collaborative programming with computational notebooks [71], and a framework for moving between code operations and GUI-based operations called mage [35].

Potential of Nonlinear Notebooks

Recent work began exploring the potential of nonlinear computational notebooks, which enable cell arrangement in more dimensions than just the linear, 1D, top-down arrangement of current computational notebooks. Fork-It [75] is a Jupyter Notebook extension enabling forking, temporarily splitting into multiple columns; users used Fork-It's split columns for comparing different branches and, unexpectedly, also to contain messes. Stickyland [73] enabled users to "stick" cells to the top of the viewport, like "freezing" the top row in Excel to keep it in view even while scrolling, and their paper showed potential use cases for it. 2D Jupyter [23] is a Jupyter Notebook extension enabling the multi-column organizational style, a row of columns of cells, based on prior work [22]; that study found statistical evidence that such a nonlinear arrangement can improve efficiency and usability for comparative analysis tasks. These works explored nonlinear notebooks in two dimensions. In et al. [27] evaluated nonlinear notebooks in three dimensions (3D) with and without a special "branching and merging" feature; the physical navigation afforded by 3D enhanced navigation and their "branching and merging" feature enhanced comparative analysis. These works show nonlinear computational notebooks' potential to address issues plaguing current, linear computational notebooks, like messiness, exploration and explanation being at odds, and enabling Space to Think [2] and physical navigation [1] for large display users for

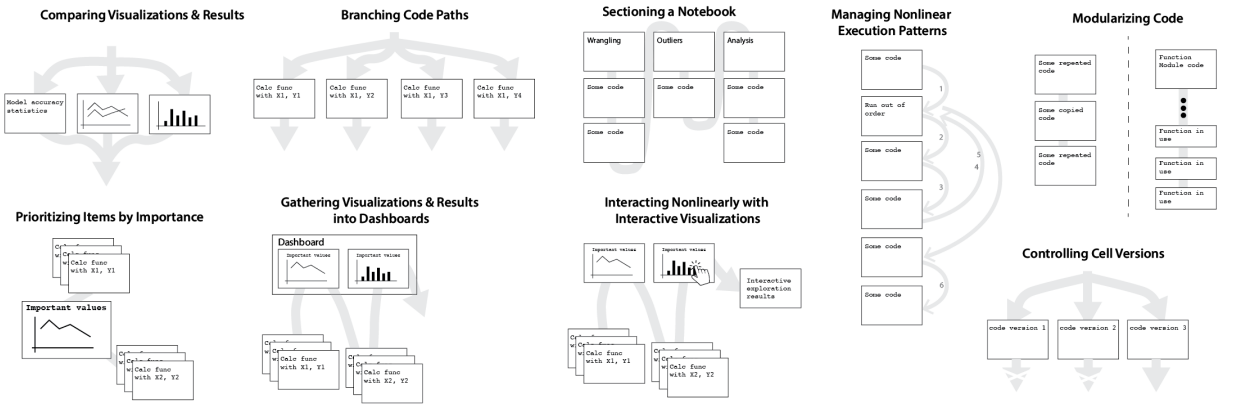

Figure 3.1: Visual Depiction of the Patterns

sensemaking with computational notebooks.

While the potential benefits of nonlinear computational notebooks have been established, the question of what problems such notebooks should be designed to address remains open; this work helps answer that question.

3.3 Pinpointed Patterns of Nonlinearity

To discern focus areas for future nonlinear computational notebook designs and answer **RQ1**, we identified patterns of nonlinearity in computational notebook work. This process started with a literature review on problems with computational notebooks and the potential benefits found by earlier nonlinear notebook works, as seen in Related Works. Our literature review used papers on computational notebooks found through Google Scholar that were generally published in an ACM or IEEE venue to ensure quality. We then conducted two guided focus groups with small sets of consulted data scientists who had interests in nonlinear computational notebooks to brainstorm more nonlinear patterns based on their data science experience. We then had 2 of the authors continue with affinity diagramming using open coding to organize the patterns from the review and brainstorming into groupings based on

similarity. Finally, the 2 authors conducted multiple rounds of refinement on the final set of patterns based on asynchronous feedback from a broader set of data scientists. We detail each pattern below with its definition, explanation of its nonlinearity, challenges it raises in current, linear notebooks, and our main methods of deriving it.

3.3.1 Comparing Visualizations & Results

This pattern is common in data science. Whether comparing different machine learning models, trying different parameters to optimize results, or looking to gain deeper insight into their data, comparative analyses are important. In linear computational notebooks, the spatial relationships between different visualizations and results requires jumping between different, sometimes far-apart cells, resulting in a nonlinear interaction flow. This requires virtual navigation such as scrolling, which is tedious compared to physical navigation [1] and can mean forgetting key details while navigating. Users can order cells out of run order to do comparison, but this can lead to messiness in notebooks [25]. We derived this pattern from literature [23, 75], and the data scientists.

3.3.2 Branching Code Paths

This pattern deals with semantically parallel code sections, such as when sequences of cells are copied and pasted to make similar chunks of code with at least some modifications, such as, but not limited to, different parameters, functions, inputs, or models. Linear computational notebooks don't enable parallel organization of cells, resulting generally in one branch being right after another in the 1D list. This can cause issues in comparing outputs of branches due to tedious scrolling, worsened by the fact that representing multiple branches in the same linear notebook results in a much longer notebook. Furthermore, debugging and correcting

errors in multiple branches is dependent on tedious scrolling in such linear notebooks. We derived this pattern from literature [75] and the data scientists.

3.3.3 Sectioning a Notebook

This pattern refers to the process and results of organizing cells in a notebook into different sections and subsections to aid navigation, create visual landmarks, construct or refine a computational narrative, and more. Purely linear sections, without parallelism or iterations, are created at the end of a data science workflow, but the section structure during analysis is often more nonlinear (e.g., tree-like), with many iterations and experiments. Making a computational narrative linear at the end of a workflow may require excising parts secondary to the main narrative; these parts, like failed experiments, discarded models, or older versions of cells, are lost to viewers. If not done well, this process creates a non-reproducible computational notebook. Also, navigating between sections involves tedious scrolling if jumping to different headings is not an option. Finally, current notebooks fail to enable visual representations of parallel sections and only support grouping cells insofar as the top-down order is maintainable. We derived this pattern from the data scientists and affinity diagramming.

3.3.4 Prioritizing Items by Importance

This pattern concerns emphasizing some cells over others, like important cells we wish to draw attention to and cells providing its context, or minimizing unimportant cells. Emphasizing based on relevance to a given task regardless of run order is nonlinear. This pattern includes highlighting disparate cells (e.g., finding cells that affect or are affected by changes to code in a given cell), minimizing/collapsing cells irrelevant to a task, and crafting different views of a computational narrative depending on what is relevant to each audience. Current

linear notebooks sometimes allow a level of prioritization, like collapsing individual cells and tailoring markdown cell text to indicate importance, but showing context around a given important cell is limited by the top-down structure. We derived this pattern from the data scientists and affinity diagramming.

3.3.5 Gathering Visualizations & Results into Dashboards

Creating a central hub, or dashboard, for ease of viewing, comparison, and diving into the details is valuable for deep insights. Since dashboards rely on gathering disparate outputs from various locations in one place, it is inherently nonlinear. Most current linear notebooks, with their top-down organization, do not allow users to gather visualizations in one central location as part of the analysis and computational narrative. We derived this pattern from the data scientists.

3.3.6 Interacting Nonlinearly with Interactive Visualizations

Interactive visualizations are valuable for deep analysis of data. In computational notebooks with interactive visualizations, users may interact nonlinearly, jumping between different visualizations while analyzing. Interacting within an output visual or widget may trigger callback execution located elsewhere in the notebook, possibly updating other visuals or results. Common examples are interactive filtering. Brushing is a particularly complex interaction in that it can be bi-directional between two visuals. This presents challenges for current linear notebooks, which tend to require tedious scrolling between multiple visualizations, or even visualizations and their code. It can be unclear which code was executed as a result of the interaction. Scrolling can make comparisons and debugging less efficient and potentially more difficult. We derived this pattern from prior literature [49, 74, 77, 78].

3.3.7 Modularizing Code

This pattern focuses on managing code modules (e.g., user-defined functions, classes) within a computational notebook. Code modules are often stored either near where they are used or in a specific location for ease of reference, if they are stored in the computational notebook instead of an additional file. Given that oft-used code modules can be far from at least some code that references them, jumping back and forth is required for usage and especially when code modules are updated; thus, this pattern achieves a degree of nonlinearity. This nonlinearity can be especially tedious to deal with when new parameters are added to functions, potentially requiring code updates in a number of different places throughout a notebook. We derived this pattern from literature [46, 58] and the data scientists.

3.3.8 Controlling Cell Versions

Version control for computational notebooks is an issue researchers have worked on [25, 32, 33]. Since code cells can be changed and run, knowing the execution order of cells is not enough to ensure reproducibility; cell history is also necessary. Overwriting code can cause problems ranging from messing with established cell dependencies to being unable to reproduce the current notebook state through "Restart and Run All" and even forgetting relevant past versions of a cell. Also, overwriting cell code without saving old results somewhere makes comparing the old and the new tedious even if possible. While users can use version control mechanisms for tasks like comparing outputs from different versions, backtracking to a more stable version, switching between different versions of different cells, and maintaining older versions of code, these tasks may be burdened by the top-down, 1D arrangement of cells; expressing multiple versions, much like multiple branches, in a nonlinear way could improve usability given the nonlinear nature of version control. We derived this pattern

from literature [32, 33] and the data scientists.

3.3.9 Managing Nonlinear Execution Patterns

This pattern deals with executing cells in a different order than their view. This often happens during early exploration, restructuring, testing, or cleaning of a notebook. This ability to run cells "out of order" leads to nonlinear execution patterns, as well as messiness, debugging difficulties, issues reproducing results, and more. Finally, nonlinear execution patterns impact user awareness of notebook history and how the current notebook state can be reached again. We derived this pattern from literature [25, 62] and the data scientists.

3.4 Survey Methodology

After refining the patterns, we built a survey to assess perceptions of how problematic they are to answer **RQ2** and **RQ3**. The survey began with demographic questions (e.g., years of experience with data science and computational notebooks, the monitor setup types participants use with notebooks). Next, the survey introduced each pattern of nonlinearity individually and asked users to rate how problematic each pattern is for current linear computational notebooks; an option was available for if they are unsure or have not encountered the pattern. Users could also add qualitative feedback about their answers. Finally, the survey asked users to rate their level of agreement with the idea that nonlinear computational notebooks (specifically in 2D) could help address the nonlinear patterns introduced by the survey; users were given an example image of a nonlinear notebook, as seen in Figure 3.2.

We sent the survey through various channels, including the Jupyter discourse forum, data science Discord channels, an email list from a workshop on computational notebooks where

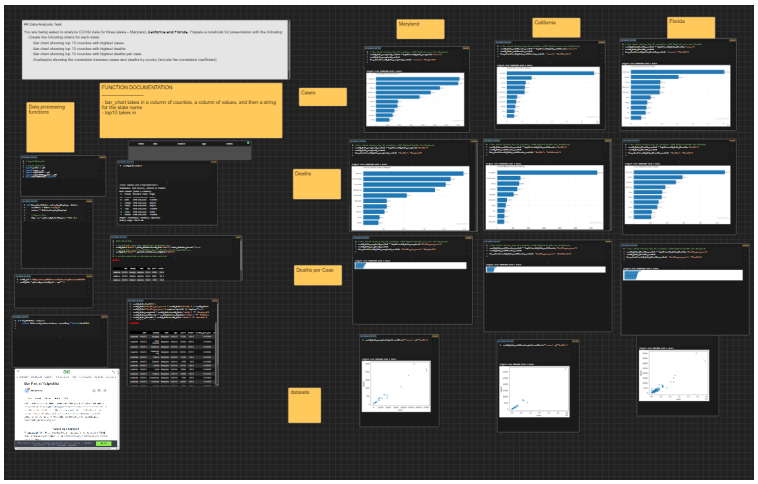

Figure 3.2: Example Nonlinear Notebook from the Survey

nonlinear notebooks were presented, and academic listservs for both a multi-university research group and a large, public research-focused university. We received 25 responses, five from academic listservs, 10 from the workshop participants and their colleagues, and 10 from a mix of the other sources.

3.4.1 Data Analysis Process

We divided the data analysis into three areas: Demographics, Likert-Scale Questions, and Qualitative Comments.

Demographics

We sought to both get a sense of the general level of experience our participants had, what computational notebooks they have used, what they use computational notebooks for, and identify individual responses with certain characteristics. For the first item, we calculated the mean values for both the number of years a participant has used computational notebooks and the number of years a participant has done data science. For the second and third items,

Patterns	(N/A) Don't Know	(1) Not a Problem	(2) Minor Problem	(3) Moderate Problem	(4) Serious Problem	(5) Critical Problem	Mean	Median
Comparing Visualizations & Results	2	0	7	11	4	1	2.96	3.00
Branching Code Paths	3	0	4	9	8	1	3.27	3.00
Sectioning a Notebook	2	3	7	10	3	0	2.57	3.00
Prioritizing Items by Importance	1	5	6	11	2	0	2.42	3.00
Gathering Visualizations & Results into Dashboards	0	3	5	8	6	2	2.96	3.00
Interacting Nonlinearly with Interactive Visualizations	0	3	5	12	4	0	2.71	3.00
Modularizing Code	0	2	9	10	4	0	2.64	3.00
Controlling Cell Versions	3	0	3	8	5	5	3.57	3.00
Managing Nonlinear Execution Patterns	1	1	6	4	7	4	3.32	3.50

Figure 3.3: Heatmap showcasing responses, including mean and median response, to how problematic each pattern is.

we calculated the proportion of participants who answered yes to each context or notebook. For the last item, we identified users who have not used a widescreen display or multiple monitor setup for their computational notebook work, which was further divided into those who have only used a laptop.

Likert-Scale Questions

We calculated mean and median responses to gauge average perceptions. Likert-Scale data is ordinal, so the median may be preferred. Finally, we counted how many respondents considered each pattern to be serious or critical with current linear computational notebooks.

Qualitative Comments

We used a grounded theory approach to identify themes present in the comments that help classify and summarize the feedback given. We went over feedback for each pattern and for nonlinear notebooks to find comments that reflect similar issues or insights to gather into themes. We also highlighted comments on important issues.

3.5 Survey Results

We divide survey results into three areas: Demographic Results, Likert-Scale Results, and Qualitative Results.

3.5.1 Demographic Results

6.92 was the mean years of experience using computational notebooks; 7.84 was the mean years of data science experience. Seven respondents reported less than four years experience with computational notebooks, nine reported between four and eight years experience, and nine reported more than eight years experience with computational notebooks. Two participants reported the minimum years of experience with computational notebooks (one year), and one participant the maximum, 20. Computational notebooks like Jupyter are fairly new, so having three answers of 12+ years begs the question of how these three define computational notebooks.

All participants had used Jupyter Notebooks, 80% used Google Colaboratory and 28% used ObservableHQ. 96% used computational notebooks for work, 76% for educational assignments, 72% for hobbies, and 16% for competitions, like on Kaggle. Two participants had used computational notebooks for prototyping, and one participant each had used computational notebooks for teaching and for small experiments respectively.

Three participants (12%) only used laptops for data science, and six (24%) other participants used a desktop monitor at most. The remaining participants (64%) had used a larger setup with a widescreen monitor (20%) or multiple monitors (52%). We also asked survey participants if they had used a tiled wall display for data science, but none had.

Table 3.1: Can 2D Notebooks Address the Patterns?

Item	Value
(1) Strongly Agree	2
(2) Agree	8
(3) Slightly Agree	10
(4) Neutral	2
(5) Slightly Disagree	1
(6) Disagree	1
(7) Strongly Disagree	1
Mean	2.96
Median	3.00

3.5.2 Likert-Scale Results

Figure 3.3 summarizes how problematic each pattern of nonlinearity was seen as by the participants. Given the mean and median responses, each pattern was deemed at least a minor problem, with most patterns closer to a moderate problem. Controlling Cell Versions and Managing Nonlinear Execution Patterns were deemed closer to serious problems than other patterns, while Prioritizing Items by Importance was deemed closer to a minor problem, receiving the most "Not a Problem" ratings by participants. Controlling Cell Versions and Managing Nonlinear Execution Patterns received the most ratings of "Serious Problem" and "Critical Problem" combined; Branching Code Paths and Gathering Visualizations & Results into Dashboards also received a high number of such responses. Finally, Comparing Visualizations & Results, while seen more as a moderate problem, scored the same mean response as Gathering Visualizations & Results into Dashboards (2.96). One participant did not rate some patterns.

Survey participants mostly agreed that 2D computational notebooks could help address the patterns, as seen in Table 3.1. This suggests cautious optimism towards 2D notebooks' potential over 1D notebooks. Of those who did not agree with the above sentiment, the "Strongly Disagree" participant focused on accessibility issues of nonlinear notebooks to

blind and low-vision users, and the "Disagree" participant dislikes infinite canvases. The "Slightly Disagree" and "Neutral" participants did not leave qualitative feedback.

3.5.3 Qualitative Results

We divide qualitative results into three sections: Feedback by Pattern of Nonlinearity, Summary Themes in Pattern Feedback, and Impressions of 2D Nonlinear Notebooks' Potential.

Feedback by Pattern of Nonlinearity

Here we note relevant feedback for each pattern of nonlinearity. Eight participants left no qualitative feedback on the patterns, and those who left feedback did not always do so for every pattern.

For **Comparing Visualizations & Results**, two participants felt visualizations being far apart is bad, with one saying this is "problematic for the comparison of results and quick interpolation from them." The other said, given difficulties seeing multiple results in a single browser tab, addressing this "requires opening multiple windows and arranging on a larger display." While users can virtually navigate through scrolling to different outputs for comparison in linear computational notebooks, four participants felt scrolling is disadvantageous; one noted how scrolling can affect a user's sense of context, stating "Minor refinements to one or both visualizations requires lot of tedious scrolling up and down; oftentimes one accidentally lands on wrong cells, losing not just time but also context." Another felt this system adds to their cognitive load especially given how often "a significant chunk of code" is needed for plots, stating "it is a mentally taxing process to (1) run the data analysis code, (2) run the plot code, (3) inspect the resulting plot, (4) scroll up to either sections of code, (5) edit the code, and (4) repeat." Furthermore, a different participant felt it was "hard to know

where to extend a given result" given the amount of scrolling needed in linear computational notebooks.

While users can code many visualizations as output of one code cell, one participant disliked this workaround, saying it is "an overhead and it is usually easier to make many plots and scroll up and down." Another agreed, noting they tend to duplicate visualization cells and change inputs even though this method is "fraught," saying "it's often better to just do side/by/side viz on the same plot", but felt "the plotting code is trickier." Some users may avoid these problems by overwriting code and re-running cell(s), but, as one participant noted, "it can be annoying to remember results from before if you re-run a cell with different parameters, settings, etc."

Overall, these usability issues around comparisons make computational notebooks less usable; in one participant's words, this pattern "is one of the reasons I've steered away from notebooks for visualization and over to scripts with multiple figure outputs." Other comments here include support for markdown cells to aid navigation, a note that VSCode stores variables for ease of comparison, and support for well-structured nonlinearity to address this pattern.

For **Branching Code Paths**, two participants preferred to use multiple notebook files to represent branches instead of working in one computational notebook. Two other participants felt branches introduce tedious scrolling, with one saying "sometimes you want to test multiple methods for analysis, and multiple cells take up too much vertical space." The other felt this pattern overloads internal memory, saying when "you are trying out different variations of the code in each branch" that "it can be hard to remember which branch is the current or stable implementation." Also, one participant noted "branching code also involves accidentally changing some shared variables, i.e., forgetting to reset them for each branch." Duplicating code, in one participant's words, can also cause notebook work to "get

inefficient and messy with lots of duplicated code." Finally, one participant noted issues with this pattern can become especially prevalent when working collaboratively.

While nonlinear notebooks could help address the Branching Code Paths pattern, better coding practices, as suggested by two participants, can also help. Per one participant, if the copied cells result in an "exact duplicate" of code or only "minor changes," then instead of using this "bad coding" practice, copied code "should be replaced with a function." For another workaround, one participant would store different branches' outputs to a CSV file and compare them in another tool. Finally, one participant didn't mind this pattern in linear computational notebooks since they view them "like scratch pads" where they make "many trials and branches;" they felt there is an art to "cleaning the notebook at the end and running it top to bottom to make sure it works."

For **Sectioning a Notebook**, some agreed that this is a problem. One participant "personally struggled with" this pattern, stating they've "tried using markdown blocks to help, but they don't work as well as needed." Another participant felt "sections and subsections help the navigation," but "the problem of scrolling back and forth" remains. A third participant felt that "you need an extra navigation tree to jump to each section, which is not very intuitive." A fourth participant expressed support for hierarchies in computational notebooks, noting that "when encountering something I want to address as a separate 'thread,' I'll simply copy the notebook into a new one and make modifications." While this is a decent workaround, this participant noted it leads to code consistency issues later on, requiring a "lengthy revision" process. Finally, a fifth participant also created multiple notebooks for sectioning.

Other responses felt Sectioning a Notebook is not a problematic pattern. One participant stated they "use markdown headers with a [Table Of Contents] to solve this just fine," while another felt their notebooks are "generally not large enough for this to be a problem where

sections need to be marked/remembered." A third participant didn't see a problem with this pattern, and a fourth participant expressed mixed feelings, noting "with some projects this is an issue, but sometimes linear sections are relatively intuitive."

Finally, two participants noted they use environments, like VSCode and JupyterLab, with tools to mitigate this issue through collapsible sections, and one participant expressed that "relegating" tools to address this issue to visual features "neglects the needs of nonvisual data scientists."

For **Prioritizing Items by Importance**, five participants said this pattern is problematic; one participant wished "there was some interaction to prioritize" such as "collapse and expand." Another stated exploratory work in computational notebooks is "messy" and that a user does not "need all the cells to make sense of the flow," which means hiding or showing cells "will be useful." Two other participants felt being able to prioritize items would be nice; one said missing such features is "not a deal breaker." Finally, one participant noted extensions for Jupyter Notebooks can address issues like "collapse/expand," and that Stickyland [73] "supports highlighting certain important cells."

Five different participants said thoughtful use of markdown cells, like using header font sizes to indicate importance, alleviates this issue. One other participant felt prioritization is a "skill of the author," not a "quality of the document," suggesting it is not a user experience issue. Another participant felt a "(book)marking feature to emphasize things/lines of importance" would be enough, instead of breaking the linearity of current computational notebooks. Finally, one participant worried prioritizing would not work well in a collaborative setting, stating it "might create an emphasis on a non-important cell."

For **Gathering Visualizations & Results into Dashboards**, eight participants said it is problematic. One participant felt "the linear structure doesn't seem to support this well."

Another said using Streamlit, a visualization package, to create dashboards is "tedious," and a different participant felt scrolling horizontally in mini-plotly dashboards "is a pain." Yet another agreed notebooks lack "good dashboarding support" but also felt notebooks will not "recreate Tableau," while another stated "you will need to use other tools" to create dashboards.

Five participants expressed skepticism on creating dashboards in notebooks. One participant uses computational notebooks "mainly to experiment and share early results" and felt dashboarding is "not really aligned with my workflow" in part because their notebooks are "typically small enough that an additional collection process is not necessary." Another doubted computational notebooks are "the ideal or perhaps intended way" to create dashboards; they recognized, however, the "potential utility" of "having that functionality as part of an end-to-end data science process in notebooks."

For **Interacting Nonlinearly with Interactive Visualizations**, four participants felt navigating between interactive visualizations necessitates tedious scrolling, which one participant noted means "you can lose context of where you are in the notebook." Another participant keeps their "long" visualization codes in a separate file to keep the code cell's length as small as possible" to deal with this issue. Other issues mentioned as related to this pattern include reproducibility, or keeping the computational narrative intact, and converting results from interactive explorations into data or variables. For the former, one participant noted if a "user interacts with a visualization" in a particular way, such as making a selection, this interaction may not be "obvious to another user" working on the notebook later. For the latter, one participant wished they could convert "exploration results into data/variables" for "reuse" in "later cells." It should be noted that AnyWidget [49] does enable such "reuse" of exploratory results.

Two participants expressed skepticism about this pattern. One participant asked "is there

a need for [interacting with multiple visualizations in one notebook]?" The other felt that education to help make "better writers in notebooks" would address this issue, stating "this is bad praxis, not theory."

For **Modularizing Code**, several participants felt it is problematic with additional caveats. One participant felt this is easier to manage "in smaller and medium size notebooks" and "very difficult as the notebook gets longer and there are too many functions to change/parameters to update." Another felt their notebooks were "typically not large enough for this to be a concern" since they "don't really have modules embedded in a single notebook." This issue related to notebook size may be due in part to what one participant noted: "find and replace tools are lacking" in some notebook environments, such as JupyterLab, leading to the need to manually change each instance of a function instead of finding all instances and replacing them with an updated function signature. Furthermore, one participant wished that find operations, when available, could be restricted to "search within a cell" instead of always the entire notebook.

Two participants felt Modularizing Code is a common problem in programming, especially since, as one put it, "updating a function means re-running most (or all) of the other code." One solution to this problem is using computational notebooks "in concert" with IDEs (integrated development environments); some IDEs, as noted by one participant, enable users to "see your hierarchy within classes, functions, etc." as well as "jump to other code sections" as noted by another participant.

Other comments here include wishing one could "split a notebook into multiple smaller notebooks as modules," a note that one can "fold those functions," and an assertion that "no one has guided folks into writing good computational literature" as the real problem here.

For **Controlling Cell Versions**, the qualitative feedback largely agreed it is problematic; one participant went so far as to say they "have never been able to do any sort of cell versioning over notebooks" while another felt it "leads to serious issues later and lost / irreproducible work." Another participant noted "it can be hard to keep track of when a code cell and intermediate results are stale" and felt annoyed at having to re-run all preceding cells, including cells "which have long run times" to ensure consistency. Even the participant most skeptical of nonlinear notebooks felt this was an issue, noting that they sometimes "share a cell that was meant to get deleted" or share a "saved version" that is not the "complete executed version," effectively sharing "too soon."

Some suggestions from participants include version control for individual cells, being able to mark specific cells "to keep history, rather than a global on or off," an "interaction to scroll horizontally to change versions," and having a "full 'commit' history" for each cell. One participant felt they tended to get by with the undo function in JupyterLab as long as they are "still actively working in that area of the notebook," while another used "git/vscode history for version control." The latter felt "it would be cool if [a] notebook can handle history itself."

For **Managing Nonlinear Execution Patterns**, responses agreed it is problematic, but also a key feature of notebooks. One participant noted this "can be a serious problem for people unaware of the code itself and may not understand various dependencies" while another felt it "can be a problem when something breaks, especially after leaving the notebook for a period of time and returning to it to make changes." Yet another participant complained that "so many things," such as "assigning multiple values to one variable, change of environment, etc.," can "cause a cell break and you have to restart kernel and run from the start." Restart and run all was suggested by several participants as a good habit for beginner coders to form, combined with "continuously moving cells up and down to provide

the correct execution order," as part of their best practices or "notebook hygiene."

Three participants felt the nonlinear cell execution ability of computational notebooks is a good feature for various reasons, like exploratory programming and audience-tailored presentations. One participant felt that without this ability, "several QUICK experiments will become impossible." Another participant noted they are often "running only specific cells when demo-ing code to collaborators, which really saved time." That being said, they also "run into issues where some error crops up because [they] forgot to execute an earlier code block."

Other comments include how third-party notebooks often have issues with out-of-order cells, making the process to "get things working" tedious, that "some linters and environments (vscode plugins) warn you when something is out-of-order," and a complaint that some computational notebook users aren't interested in learning best practices.

Summary Themes in Pattern Feedback

Through reviewing the qualitative feedback, we found the following themes, which summarized the majority of feedback received: Drawbacks of Virtual Navigation, Using Other Tools, Learning Best Practices, and Collaboration Issues.

The **Drawbacks of Virtual Navigation** include reliance on internal memory, tedious scrolling, and inefficient use of space; the first two appear more often. Linear computational notebooks require virtual navigation and do not make good use of larger, widescreen displays due to the top-down organization of cells. Participants felt these drawbacks in several of the patterns, especially Comparing Visualizations & Results, Branching Code Paths, and Interacting Nonlinearly with Interactive Visualizations. Nonlinear notebooks can address these drawbacks since they would not be constrained to a 1D, top-down organization of cells.

This would enable Space to Think [2] with more external memory use and less scrolling.

Another common theme was **Using Other Tools**, like Tableau, and IDEs that can contain notebooks. Linear computational notebooks have certain limitations which may lead users to utilize other tools for certain tasks. For Gathering Visualizations & Results into Dashboards in particular, tools other than notebooks were commonly used for dashboarding, like Tableau. Several participants also noted use of extensions to computational notebooks, such as the linearity-breaking Stickyland [73], to address issues raised by the patterns.

Learning Best Practices was a common suggestion and the regular theme of the one participant who strongly disagreed with whether 2D nonlinear notebooks had the potential to address these problems; it was also mentioned by several other participants at least once. While learning best practices, such as regularly restarting and running all cells, and utilizing them is important and can save time on cleaning, it is not the only way to address the patterns found in this paper.

The remaining theme, while less common than the above three, was still worth mentioning; **Collaboration Issues** was mentioned by two participants as a relevant consideration, once on Branching Code Paths and once on Prioritizing Items by Importance. When multiple users are coding in a computational notebook, some challenges may be exacerbated or begin to appear, such as different prioritization structures leading to disparate understandings of the importance of code cells and outputs, or more branches being created as different users work. Given the increase in collaborative coding with computational notebooks, this theme is worth consideration in designing future computational notebooks.

Impressions of 2D Nonlinear Notebooks' Potential

While the Likert-scale results for 2D nonlinear notebooks' potential were cautiously optimistic, the qualitative feedback tended to focus on the "cautious" part of said optimism. That being said, positive feedback included that 2D nonlinear notebooks "could be helpful for organizing workflows, visualizations, and branches, and might make browsing a notebook easier," with another participant stating such notebooks would "aid developers from writing and grouping and managing code better." One participant went so far as to say that "notebooks in the form of canvas seems like a great idea."

Concerns included information overload, readability for new users, maintaining "a disciplined, reliable notebook" in 2D, and accessibility for non-visual users. One participant noted the need for a large display to best use a nonlinear 2D notebook, and expressed concern about information overload, stating "seeing the whole graph (of a non-trivial workflow) seems overwhelming as well." A second participant worried about readability, saying "the final output will be less readable for someone seeing the notebook for the first time." Two other participants worried about maintaining "disciplined, reliable" computational notebooks. One of these participants felt 2D canvases, in addition to providing certain benefits, may "add another layer of complexity to navigating a notebook if the author isn't very strict with how they organize their cells"; they gave the example of messiness with scratch work in linear notebooks, and worried that "in a 2d layout, my notebook might turn out even messier." Still, they felt "some kind of guardrails or educational tooltips/predefined layout templates to help coders establish and stick to good organizing practices could be helpful" in addressing this. The other participant in this duo expressed concern for how they might use 2D canvas notebooks, stating they "could see myself spamming the charts and having them all over the place with no clue which belongs to which block of code." Finally, one participant expressed concern with how accessible nonlinear computational notebooks in 2D

canvases would be for blind and low vision users, who generally need to hear content instead of see it; this participant felt "1D is a critical predicate for the 2D and they need to work together" and expressed concern about how "ordered execution," "the hallmark of [a] trustworthy document," is something "we can't measure in 2D."

Other comments include a note that "a 2D notebook should have less confusing interactions" given how the number of interactions grows "with version control and nonlinearity," a note that layout remains challenging since "the user needs to do the organization," and a note that one participant does not "like infinite canvases all that much."

Existing Workarounds for Linear Notebooks

While nonlinear computational notebooks may well address patterns of nonlinearity, clever users have found some workarounds for linear notebooks; this includes some survey participants who detailed their workarounds in comments. Their workarounds include using multiple notebook files to represent different branches or sections, making side-by-side visualizations from one code cell, opening the same notebook multiple times to show items side-by-side, and using IDEs together with computational notebooks. Below we briefly describe each workaround and detail how a nonlinear notebook might address the issues.

Making multiple related notebook files to represent different branches/sections can work; it can also lead to consistency issues between the related notebooks when key code gets updated; this can require a tedious, lengthy revision process to correct that may include "making an external source file and importing common functions from it." Nonlinear notebooks could spatially represent different branches/sections within one notebook, avoiding consistency issues between different files.

Coding multiple visualizations side-by-side in a single notebook cell is doable, but the addi-

tional overhead to correctly code multiple visualizations into the same output is burdensome. One participant felt "it is usually easier to make many plots and scroll up and down" than to make one code cell with multiple visualizations. Nonlinear notebooks could enable users to quickly copy code cells with visualizations, make the necessary changes to the input or code, and arrange code cells in a more intuitive manner to show visualizations side-by-side.

Users can open a single notebook multiple times and arrange the multiple views of the notebook side-by-side, but this seems a hackish approach still limited by the linear format of current computational notebooks. Nonlinear notebooks could enable users to open one view of a notebook with cells arranged in nonlinear ways that make sense for what they are working on.

Finally, IDEs with computational notebook capabilities, like VSCode, are powerful tools that can address multiple issues currently plaguing linear computational notebooks, like keeping track of variable states and avoiding stale results. Nonlinear notebooks can also help with issues such as the aforementioned, as it would be possible to, for example, have a code cell or special cell dedicated just to showing the current state of variables. Overall, the authors view IDEs and nonlinear computational notebooks as likely complimentary.

The last workaround, from the authors, is using shell script windows to write python code; good code gets copy-pasted into a computational notebook. This workaround utilizes nonlinear space to arrange scripts, which suggests that a good nonlinear notebook environment could fulfill such needs without having to work with other tools.

3.6 Discussion

We divide our discussion into three sections: Evaluating the Patterns, 2D Nonlinear Notebooks & the Pinpointed Patterns, Accessibility Concerns for Nonlinear Notebooks, and Nonlinearity & Prior Literature on Computational Notebook Issues.

3.6.1 Evaluating the Patterns

Of the patterns we identified, the most critical ones according to our analysis are Managing Nonlinear Execution Patterns, Controlling Cell Versions, Branching Code Paths, and Gathering Visualizations & Results into Dashboards. The first three of these deal with issues that rise due to the iterative, exploratory nature of data science. As users explore their data, they tend to create different branches for different experiments, update code cells to newer versions, and execute cells out of order to update functions, visualizations, and more. The last critical pattern deals with an issue that affects users' ability to efficiently and effectively utilize results and visualizations to gain and communicate insights. While a single visualization may be worth a thousand words, a group of related visualizations clustered into a dashboard can be better than the sum of its parts through streamlining comparative analyses.

While not as critical, Comparing Visualizations & Results, Interacting Nonlinearly with Interactive Visualizations, and Modularizing Code still seem worth addressing. The first two of these patterns could be helped by enabling better support for Gathering Visualizations & Results into Dashboards, while the last pattern is often more of a notebook cleaning issue.

Finally, Sectioning a Notebook and Prioritizing Items by Importance seem somewhat well addressed in current linear computational notebooks. Still, there may be better ways to

address these patterns in nonlinear computational notebooks, such as the multi-column style found by Harden et al.[22] and enabled in the 2D Jupyter extension [23] for sectioning.

3.6.2 2D Nonlinear Notebooks & the Pinpointed Patterns

To answer **RQ3**, we discuss impressions of the potential of nonlinear notebooks and design opportunities and challenges.

Current Impressions of Nonlinear Notebooks' Potential

With 80% of participants at least slightly agreeing that 2D nonlinear notebooks could address the patterns of nonlinearity this paper covers, and with the qualitative feedback, it seems most participants are cautiously optimistic about nonlinear notebooks' potential. Benefits participants foresaw include better "organizing workflows, visualizations, and branches," "writing and grouping and managing code better," and easier browsing of notebooks; challenges include information overload, readability, learning new best practices, and accessibility for nonvisual users; the latter will be covered in Section 3.6.3.

Design Opportunities

Nonlinear computational notebooks, through expanded use of space, may provide opportunities to create more effective, efficient interactions for some data science tasks and processes, like the patterns listed as key problems in Section 3.6.1. Interactions like swiping horizontally to see versions of a code cell, snapping together a grid of visualizations into a dashboard, and copy-pasting code cells to create a new branch provide ample opportunity to make good use of nonlinear, 2D or 3D space. Enabling such functionalities will likely require a concerted effort by researchers, designers, and developers to fully realize nonlinear notebooks'

potential. We recommend that such future work focus on addressing the most critical of the patterns identified in this work.

Design Challenges

While there are opportunities for innovation with nonlinear notebooks, there are challenges to address, as well. Given the potential for information overload from having many items visible on a larger display, determining how much information should be displayed at once, or at different levels of semantic zooming, to prevent analysis paralysis is a worthwhile avenue of research. Navigation may also be a challenge, especially related to zooming and how markdown text should "react" to such actions to preserve the readability of a nonlinear computational notebook. Furthermore, with the expanded space and interactions available to users, learning new best practices may be essential training to best use nonlinear computational notebooks; guardrails, tooltips, templates and more may be part of this. Best practices are important given the potential for messiness to be spatialized in nonlinear notebooks. Another challenge related to this is how much structure should be enforced or provided to users as a tool for keeping cells organized; while freeform arrangement of cells can enable novel arrangements, it may also exacerbate messiness. Balancing structure and freedom in nonlinear notebooks remains an open question. We recommend anyone working to develop nonlinear computational notebooks keep these potential problems in mind.

3.6.3 Accessibility Concerns for Nonlinear Notebooks

Work on nonlinear notebooks tends to assume users are sighted; however, capable nonvisual data scientists and analysts exist and deserve consideration when designing future computational notebooks. Given that most nonvisual users use screen readers to listen to content,

a key problem facing nonlinear notebooks is that of how to represent them to nonvisual users. Two potential avenues to address this concern include developing capabilities to turn nonlinear notebooks into linear ones and creating graph representations of notebook flows navigable with keyboard shortcuts and understandable via screen readers. As we seek to improve computational notebooks for sighted users, we must also consider how to maintain and improve usability for nonvisual users.

3.6.4 Nonlinearity & Prior Literature on Computational Notebook Issues

It should be noted that prior work has identified issues with computational notebooks without citing the linear, 1D, top-down organizational style of current computational notebooks as an issue. Chattopadhyay et al. [11] focused on a diverse set of issues, ranging from resusability to managing code to scaling to large datasets, which were gathered from observing and interviewing data scientists and then validated in a survey study. Rule, Tabard, and Hollan [62] focused on how exploration and explanation, both of which are goals of computational notebooks, are often at odds. Chattopadhyay et al. did not mention nonlinearity at all, but Rule, Tabard, and Hollan did note support for nonlinear narratives, unsupported by current linear notebooks, as a design opportunity. While the linear, 1D style may be, in the words of Rule, Tabard, and Hollan [62], "elegant," the success of prior works that break the linear mold [23, 73, 75] gives reason to consider linearity as a potentially limiting factor for computational notebooks.

3.6.5 Limitations

The validation and evaluation of the patterns and the potential of nonlinear notebooks was based on a survey study of user perceptions as opposed to a set of user studies. However, since one of the goals of this paper is to help inform future research into and design of nonlinear computational notebooks, we believe a survey study is adequate to establish actionable information. Also, some survey participants may have been biased towards the potential of nonlinear notebooks due to participation in the mentioned computational notebooks workshop. Finally, we are limited by the small number of participants, which may be due to the length of the survey.

3.7 Conclusion

Computational notebooks are useful for exploration and explanation in data science work, but are limited by the current linear, top-down organization of their cells; recent works [22, 23, 73, 75] showed nonlinear computational notebooks could be better than current computational notebooks. To better inform future research and design of computational notebooks, we evaluated literature and our own experiences to come up with a list of patterns of nonlinearity in computational notebook work: Comparing Visualizations & Results, Branching Code Paths, Sectioning a Notebook, Prioritizing Items by Importance, Gathering Visualizations & Results into Dashboards, Interacting Nonlinearly with Interactive Visualizations, Modularizing Code, Controlling Cell Versions, and Managing Nonlinear Execution Patterns. We then conducted a survey study to evaluate how problematic each pattern is and why; we found that Managing Nonlinear Execution Patterns, Controlling Cell Versions, Branching Code Paths, and Gathering Visualizations & Results into Dashboards were the top issues, with Comparing Visualizations & Results, Interacting Nonlinearly with Interac-

tive Visualizations, and Modularizing Code also being worthwhile issues to address. Overall, this work may help guide future research and design of computational notebooks.

Chapter 4

Exploring Organization of Computational Notebook Cells in 2D Space

4.1 Introduction

Since computational notebooks, a popular data science tool [66], such as Jupyter [30] emerged, millions now use them for their work [62], including research communities [55]. Computational notebooks combine code, visualizations, and text in one document, enabling analysts to construct a *computational narrative* [62]. Computational notebooks also let analysts interleave results with code and edit code in-place, helping the iterative, exploratory process of data analysis [31, 62] to quickly test and refine models on their data and see results.

Current computational notebooks do not handle non-linearity well given the 1-dimensional (1D) layout of cells. Rule, Tabard, and Hollan [62] view support for non-linear narratives as a key design opportunity. We identify two common types of non-linearity: different operations on same data, and same operations on different data. The former may occur when an analyst tries different models, tweaks parameters, and compares results to iteratively refine their work on a single dataset. This is valuable for presentation to a technical audience

interested in choosing an analysis method. The latter may occur when an analyst performs the same analytic steps on two different sets of data, such as separate subsets of a single, larger dataset; this is valuable for comparative analysis.

Another limitation of current computational notebooks is navigating long notebooks with many cells, as scrolling up and down becomes tedious. To address this, some of the authors open the same computational notebook in multiple windows, with each one showing a different part of the notebook.

To better represent non-linearity and improve navigation in longer notebooks, the authors propose organizing computational notebook cells in 2 dimensions (2D); instead of ordering cells from top to bottom as an ordered list, a 2D environment may empower users to address non-linear narratives with non-linear configurations and enable additional methods of navigation in addition to scrolling vertically. 2D space may also empower users to encode meaning into space, like how analysts used extra space in Space to Think studies [2, 43].

This paper contributes to research on computational notebooks through exploring 2D organization of computational notebook cells and requirements gathering for designing 2D notebooks. We focused on the following research questions:

1. Given a notebook with non-linear features, would users utilize 2D space?

2. How would users organize notebook cells in 2D space?

3. How would users encode run order in a 2D layout?

4. What strengths and weaknesses might 2D notebooks have compared to 1D notebooks?

5. Would users want to use 2D computational notebooks?

4.2 Background and Related Works

4.2.1 Computational Notebooks

Computational notebooks, influenced by Knuth's *literate programming* [39] concept where authors weave "human language with live code and the results of the code" to produce a computational narrative [54], support incremental and iterative analysis, explanation of an analyst's thoughts and processes, and sharing of code, text, and visuals in one document [62].

However, Chattopadhyay et al. [11] found users struggle with computational notebooks. One struggle with the iterative process of exploration and analysis is messiness [34, 47, 62]. Analysts described notebooks and their code as "ad hoc" or "throw-away" [25, 31] and in need of cleaning [62] before presentation. Kery et al. [34] found data analysts, as part of the process of exploring alternatives, replicate code across many cells that must later be refactored, a process both tedious and error-prone [34, 75]. In short, exploration and analysis can complicate constructing clean computational narratives.

Some proposed solutions include forking and backtracking of stateful alternatives [75] and version control systems [32, 33]. Weinman's work on forking [75] introduces forking in 2D space, which supports using 2D to address computational notebook issues. Following best practices, such Rule et al.'s [62, 63], can also resolve many issues. We support such work; such guidelines are complementary to our work, and appropriate use of 2D space could be a future best practice.

4.2.2 Computational Narratives

Perez & Granger [54] note computational narratives are developed for a set of audiences and contexts; given different audiences and contexts, different storytelling strategies are necessary [41]. An audience only interested in results may find a linear narrative with only relevant cells helpful. However, some audiences may benefit from a non-linear narrative structure. Weinman et al. [75] note that "the ability to proximally compare" different visual representations of data is critical to analysis processes. In addition, an audience that wants to understand and evaluate the entire analytic process may find a non-linear narrative structure better, as it can expose alternatives tried and compared to the final results used, enhancing reproducibility. Weinman et al. [75] found analysts used forking paths to compare results of machine learning models as part of their exploratory analysis; while they focused on analysis, seeing different options tried and their results in a proximally comparable way could aid understanding.

4.2.3 Space to Think

Andrews, Endert, and North [2] focused on large, high-resolution displays for sensemaking, and found additional space aided users in two ways: it enabled externalized memory, letting users focus on the task at hand rather than on recalling important info, and it enabled encoded meaning into space, such as by clustering similar items together. That work has been expanded through study of additional space provided by virtual reality [43], as well as collaborative uses of large spaces through increased and varied content contribution [37]. Similar works demonstrating benefits of space to programming tasks include Code Bubbles [8] and VisSnippets [9].

4.3 Methodology

This study consisted of a screening questionnaire, user study task, post-task survey, and optional interview. The user study task focused on research questions 1-3. The post-task survey and optional interview focused on research questions 4-5. 50 participants from two universities' academic listservs of students and faculty applied. 43 passed screening to ensure Python and computational notebook experience. 25 completed the user study and post-task survey; we interviewed 5 of them.

4.3.1 User Study Task

We created a computational notebook with analysis of publicly available COVID-19 data focused on Virginia and two of its counties: Fairfax and Henrico. Knowledge of Virginia was not expected from participants. The analysis contained non-linear features; it included three different charts for Virginia overall, showing the same data with different analyses, while the two counties had the same analyses done with different data; each county had 3 graphs. We divided the analysis into sections with markdown cells at the beginning of each section. We converted each notebook cell into an image and randomly placed it in a jumbled pile on a Miro Board [51], infinite 2D canvases that allow users to move images around at will, connect them with arrows, create labelled frames that images can be put into, and more. We used Miro Boards because of these additional features to explore what kinds of additional visual features might help in 2D computational notebooks.

Each participant got a personal copy of the Miro Board and were instructed to take 40 minutes to complete the task of organizing the cells. We designed the task to set presentation and development as key considerations. Due to COVID-19, we were unable to watch participants complete the task.

To analyze the visual layouts created, we took all the participants' layouts and their mini-maps and put them into a single Miro board for study. Comparison and pattern finding was done with an open coding approach with multiple coders.

4.3.2 Post-Task Survey and Optional Interview

Participants completed the post-task survey after they finished the user study task, which consisted of 19 seven-point Likert-Scale (strongly disagree to strongly agree) questions investigating participants' attitudes towards 2D Computational Notebooks' potential, as well as qualitative questions about the visual layout they created, their reasoning for it, and initial thoughts on 2D Computational Notebooks' potential. We analyzed qualitative data using open coding to identify themes, and Likert-Scale data using frequencies for each choice, condensed to agree, neutral, and disagree.

In the interviews, which lasted 45 minutes to 1 hour, we delved deeper into participants' qualitative survey responses. We transcribed them and used open coding to identify themes.

4.4 Results

4.4.1 User Study Task Results

Most participants utilized 2D space, as seen in Fig 4.1 below, with the layouts grouped into 3 distinct approaches. Furthermore, the authors easily interpreted the run order of all participants' layouts except for the layout by participant P09.

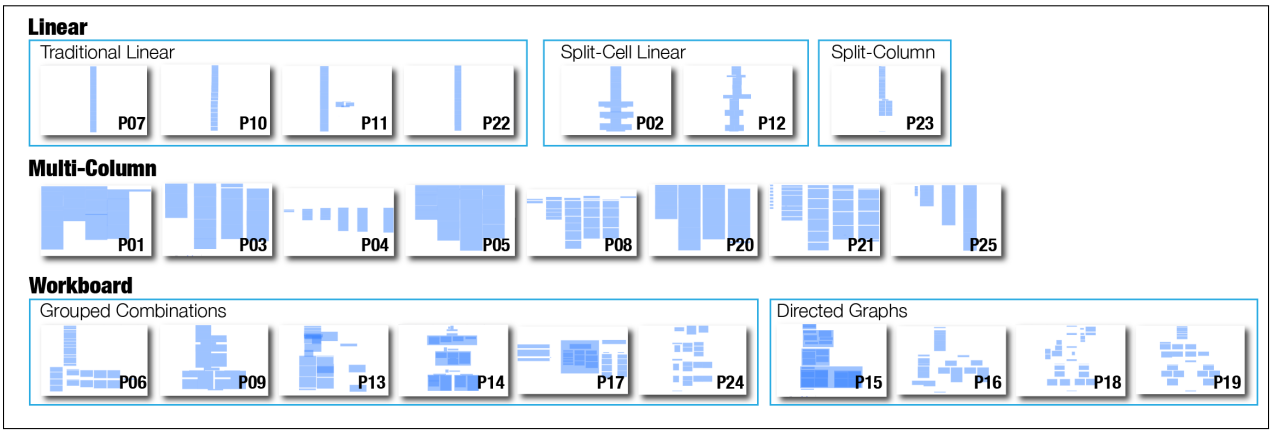

Figure 4.1: Minimaps of User Study Task Results

High-Level Design Patterns

We identified 3 high-level organizational design patterns in 2D: Linear (7 instances), Multi-Column (8 instances), and Workboard (10 instances).

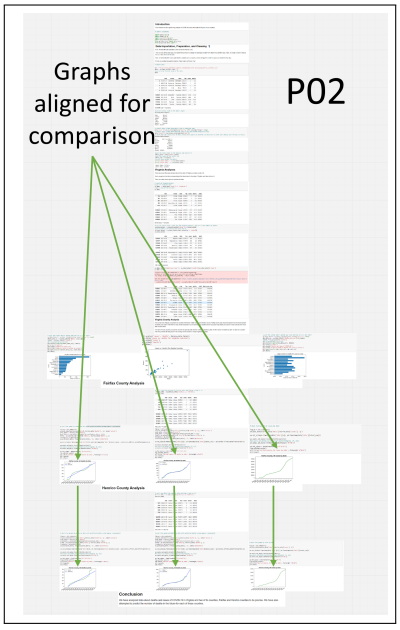

Figure 4.2: Example Linear design pattern with 3 split cells.

The **Linear** pattern uses one column as the layout's backbone and has three subgroups: Traditional, Split-Cell, and Split-Column. Traditional Linear (4 instances) is equivalent to

a 1D Computational Notebook layout. Split-Cell Linear (2 instances), like Fig 4.2 has at least 1 cell in the column is "split" into a row of cells, like the Jupyter Notebooks Split Cells extension [40]. Split-Column Linear (1 instance) eventually splits the main column into two or more columns, like Fork-It [75]. Linear layout run orders were always top-to-bottom, with Splits being left-to-right. Running splits in parallel or any order could be possible and may represent cognitive branching.

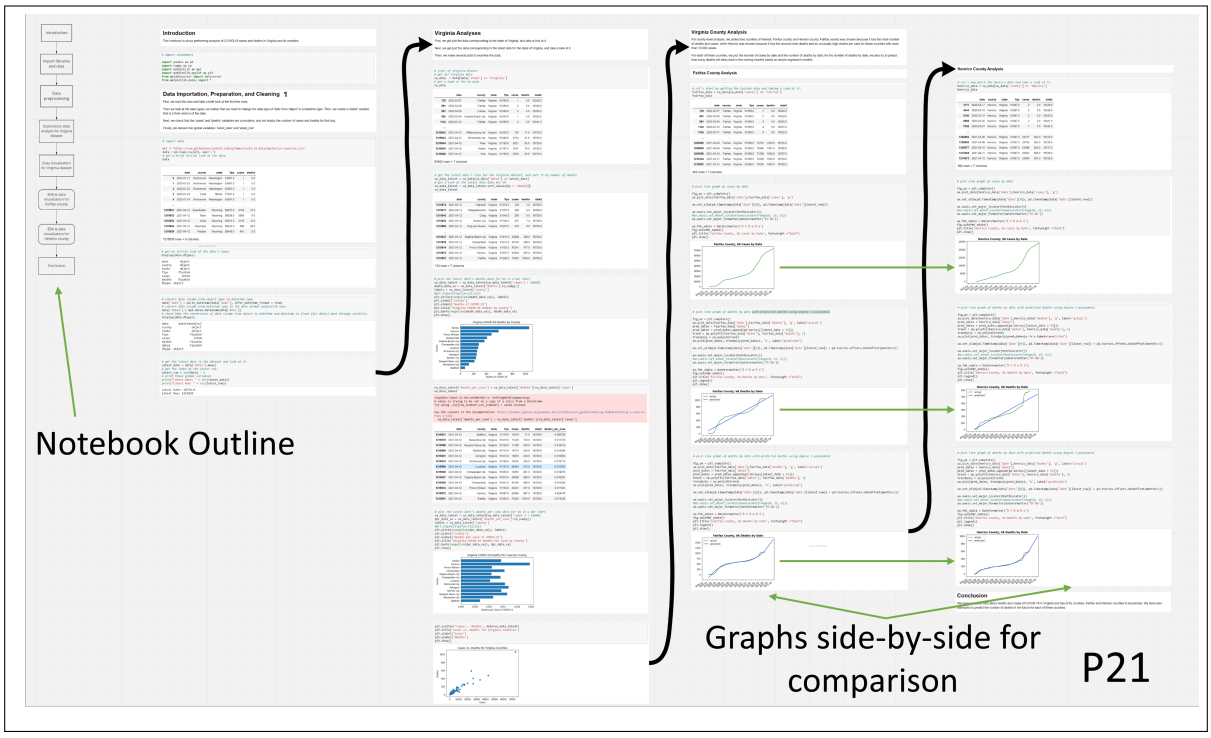

Figure 4.3: Example Multi-Column design pattern with parallel counties.

The **Multi-Column** pattern arranges cells in columns run left-to-right and top-to-bottom in columns. The columns represent "chunks" or sections from the notebook, each chunk with its own semantic meaning. Unlike Split-Column Linear, all columns start at about the same vertical position instead of splitting from a main column. 5 users aligned the 2 county columns, pushing the second column lower, as in Fig 4.3.

The **Workboard** pattern had complex 2D layouts in two sub-groups: Grouped Combina-

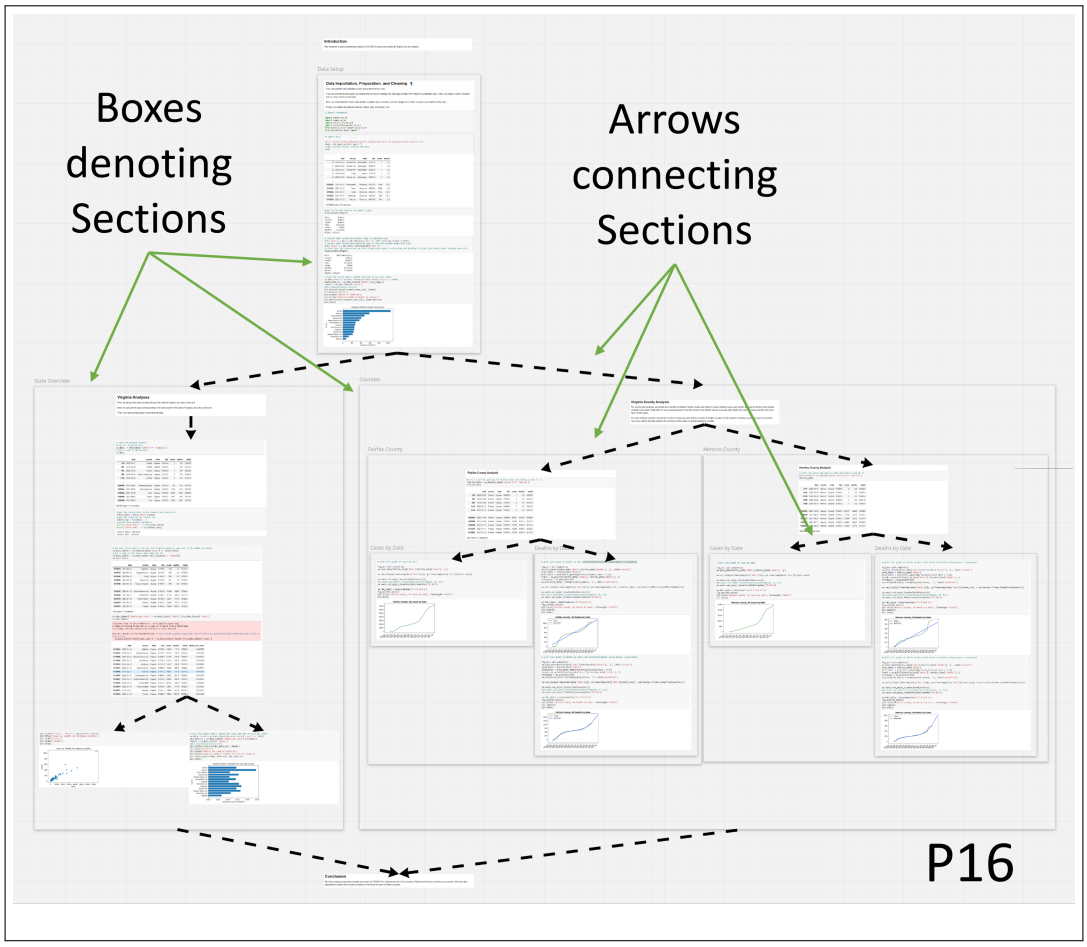

Figure 4.4: Example Workboard approach using a Directed Acyclic Graph.

tions and Directed Graphs. Grouped Combinations (6 instances) organize notebook cells into sections which may differ in strategy (e.g. Linear, Multi-Column). For example, sections may be arranged vertically with each section being multi-column. Layout P09, the only confusing run order, was in this subgroup; they arranged sections vertically and may have meant for sections to run in a clockwise rotational manner. Directed Graphs (4 instances) use arrows to develop flowchart-like run orders (as in Fig 4.4) similar to node-link diagrams, which are useful in visual programming endeavors like the block-based programming work of Suzuki et al. [67]. Each Directed Graph used a top-down progression, with 2 also using left-to-right progression.

Representations of Non-Linearity

We noted whether and how participants utilized 2D space for non-linear features of the computational notebook. 20 of 25 users utilized 2D space for one or more kind of non-linearity mentioned in the introduction. For different data, same analysis, 15 layouts aligned similar charts for the county data subsets horizontally, as in Fig 4.3 and 4 layouts aligned these charts vertically, as in Fig 4.2. 7 of 8 Multi-Column pattern users aligned the charts horizontally. These users may have sought to ease comparisons between the counties. For same data, different analysis, 4 layouts aligned Virginia charts horizontally, as in Fig 4.2.

Low-Level Features

We also noted use of certain low-level features, such as columns of cells (25 layouts), rows of cells (19 layouts), and arrows to denote flow (9 layouts). Several cell grouping features were seen, such as boxes (4 layouts) around similar cells or spatial clustering of similar cells in the same general area, with clusters separated by white space. 4 users used Miro's [51] sticky notes feature to label different clustered sections or to help others understand their layout. Finally, 2 users left a few cells apart from the rest; this could illustrate scratch space or a discard pile.

4.4.2 Post-Task Survey Likert-Scale Results

Users rated 2D notebooks' potential positively compared to 1D notebooks. Almost all survey questions had most votes on the Likert scale's "agree" side. According to the responses, summarized in Table 4.1, the tasks most likely to be helped by 2D computational notebooks are comparison activities, such as comparing results with different model parameters, different data, or different visualizations. However, some users were skeptical about 2D com-

Table 4.1: 2D Notebook Survey results with Likert-scale frequencies

Question	**Agree**	**Neutral**	**Disagree**
Better Info Layout than 1D	24	0	1
Easier to Navigate than 1D	18	2	5
More Beneficial than 1D	21	3	1
Made Meaningful 2D Layouts	20	5	0
Like Unrestricted Cell Placement	23	0	2
Infinite Canvas Useful	22	1	2
Better in 2D: Collaboration	19	1	5
Better in 2D: Presentation	20	1	4
Better in 2D: Exploring, Prepping Data	18	3	4
Better in 2D: Development, Analysis	15	3	7
Better in 2D: Debugging	12	2	11
Better in 2D: Branching Paths	20	3	2
Better in 2D: Sectioning Code	22	1	2
Better in 2D: Comparing Vis	25	0	0
Better in 2D: Comparing Parameters	23	2	0
Better in 2D: Comparing Data	23	2	0
Would Use 2D Notebooks	19	5	1

putational notebooks' usability for debugging, analysis and development; others doubted navigation would be easier in 2D space.

4.4.3 Qualitative Survey Questions and Interview Results

We analyzed the qualitative responses and grouped themes into 4 categories: Benefits, Challenges, Features, and Preferences. The first two focus on potential benefits and design challenges of 2D Notebooks. Features focuses on features that helped users make a good layout. Finally, Preferences focuses on expressed preferences for 2D or 1D notebooks. Table 4.2 shows themes by category and summarizes each theme with a participant quote. The table notes the number of participants (users) whose comments expressed the same sentiment, either in the survey response or interview format, as each theme.

Table 4.2: Qualitative Themes in Survey & Interview Results regarding 2D Notebooks

Category	Theme	Sample Quote	Num. Users
Benefits	Comparison	"I think the 2D notebook will be super useful for any kind of comparison analysis."	5
	Presentation	"This is a great tool for presenting data to layman audiences."	4
	Organization	"The 2D board is definitely hugely beneficial to organize code in a meaningful manner."	2
	Externalized Memory	"If I'm able to have everything listed in one document, I wouldn't have to be switching into web browser tabs to look at documentation or things like that."	1
	Collaboration	"2D space opens up the opportunity for multiple people working multiple spaces at the same time."	1
Challenges	Navigation	"The zooming in and out and continuous scrolling to reorganize the tiles seemed tedious."	10
	Cognitive Load	"Organizing [notebook cells] might be a little tedious however, and needs to be planned in advance."	2
Features	Arrows	"The arrows were very useful, as they helped direct the flow of the narrative."	10
	Boxes or Frames	"The boxes also helped in grouping cells together."	7
Preferences	Prefer 2D	"[2D Notebooks] felt like a great and much easier way to reorganize some of my Jupyter notebooks!"	3
	Prefer 1D	"In terms of [collaboration and development], I feel that the linear layout would still work best."	2

4.5 Discussion

1D notebooks struggle with non-linearity and navigating longer notebooks, which prompted our work on 2D notebooks. Initial evidence suggests 2D space may provide the benefits found in Space to Think studies [2, 43], and future studies may validate benefits of discovered 2D notebook design patterns.

4.5.1 Organizing Notebooks in 2D

The Split-Cell extension [40], and Fork It [75], show how 1D notebooks can benefit users with limited 2D space. The prevalence of the Multi-Column approach among participants suggests that as a new notebook extension for 2D space usage.

Still, most users liked the flexibility of cell placement in the 2D computational notebook sketch using Miro [51]. Participants with Workboard layouts designed varied, creative flows to express different aspects of the notebook structure. Given this and the fact that 20 of the 21 2D layouts' run order were intuitive, a 2D Computational Notebook could be useful even given 1D notebooks with limited 2D capabilities.

4.5.2 Enabling Strengths of 2D Notebooks

Almost all participants thought the 2D Notebook environment would be useful for comparative analyses. Their layouts placed cells or sequences of cells side-by-side, such as in the two aligned columns in Fig 4.3. Thus, 2D computational notebooks should enable users to compare results from different cells or branches in flexible and easy-to-use ways; for multi-column designs, this might mean changing the order and alignment of columns easily to enable quicker comparisons.

The potential to run notebook segments in parallel is exciting. Layouts had parallel side-by-side branches in the cell layout, or with one-to-many arrows. Data analysts may try different machine learning models with a dataset for a classification task, and could run adjacent columns of code in parallel, getting results quicker to better compare model performance and choose which method to use in production.

Finally, more sophisticated controls for running cells in a 2D notebook may prove useful;a

user may want to run a group of cells, such as those in a particular column or cluster. This would require ways to designate groups of cells. Some participants used explicit grouping features, such as boxes or columns or rows, to express such semantic grouping.

4.5.3 Addressing Tradeoffs of 2D Notebooks

While the 2D Computational Notebook concept shows promise, the results indicate potential downsides. One downside some participants noted was tedious navigation. Given participants' survey and interview responses, aiding navigation in intuitive, efficient ways seems critical for 2D computational notebook design. Two ideas for dealing with this include sectioning code cells using boxes or frames and enabling quick jumps to different sections, and enabling search utilities for keywords in code and text. In addition, an interactive mini-map with features like clicking on a cell and jumping to it may help improve navigation. Finally, certain headers in markdown cells should remain readable even when users zoom out.

Another consideration is more cognitive load; a user must organize cells during analysis. However, given how notebooks become "messy" [25] in 1D, it may be that considering cell organization during analysis could benefit users. There are also ways to address cell organization. First, a templating mechanism could be designed, in which users select from a set of common templates to pre-organize cells or provide initial structure. The templates should be based on data, such as the observed high-level strategies. After selecting a template, the notebook might provide visual affordances such as pre-labeled sections in 2D space, and interactive affordances to fill in cells in the template. For example, a multi-column template could provide a set of initial empty columns to fill in with cells. AI methods could also be developed to semi-automate cell organization through actions like suggesting templates based on code structure and static analysis [76], semantic interaction [18], and active place-

ment of cells. An AI could spot non-linear branches of 'same analysis with different data' and suggest a parallel multi-column template. Wenskovitch et al.'s work [76] on visualizing dependencies and relationships between computational notebook cells uses a dynamic graph structure that might be built upon to enable such methods.

4.5.4 Assessing Interest in 2D Notebooks

The results of the user study task provide evidence that users are willing to organize computational notebooks in 2D. The fact that, out of 25 participants, only 4 opted to align the cells in 1D supports this idea. That 19 out of 25 participants agreed that they would like 2D Computational Notebooks in their analytic toolbox also supports this assertion.

4.5.5 Limitations

This study is limited by its task structure and medium. The task structure could require more panning and zooming than would be normal in a 2D notebook. This, combined with the fact that using Miro meant assuming a generic 2D canvas with standard pan/zoom 2D navigation may contribute to complaints of tedious navigation. While more structured forms of navigation may alleviate this, the fact that such tedium was noted by 10 participants (40%) suggests that navigation is a key design concern for 2D computational notebooks. Furthermore, Miro is not a 2D Computational Notebook prototype; care was taken in drawing conclusions about design requirements for 2D Computational Notebooks.

s

4.6 Conclusion

Computational notebooks enable crafting of meaningful, replicable computational narratives. However, 1D computational notebooks make presenting certain narratives and performing some analyses difficult. Thus, we investigated 2D computational notebooks' potential to address non-linear narratives and improve upon 1D computational notebook design.

Our work shows users are interested in 2D computational notebooks' potential. The ability to easily compare results, show branching analyses, and present non-linear narrative structures in a semantic way seemed valuable to our participants. Participants' layouts were about evenly split between three strategies: primarily linear, multi-column, and workboard strategies. The workboard strategy included directed graph layouts and complex nesting of other strategies. Approximately half of the participants also made use of additional annotation features, such as arrows, labels, and boxing.

However, 2D computational notebooks have design challenges. Adding another dimension may complicate navigation. If 2D computational notebooks are to succeed, effectively aiding navigation is key. In addition, the increased complexity brought by organizing cell layout in 2D means layout aids such as templates may help minimize added cognitive load during analysis. Still, 2D computational notebooks could provide a potent 'space to think' for data scientists.

Chapter 5

Design and Evaluation of 2D Jupyter Notebooks with the Multi-Column Organizational Pattern

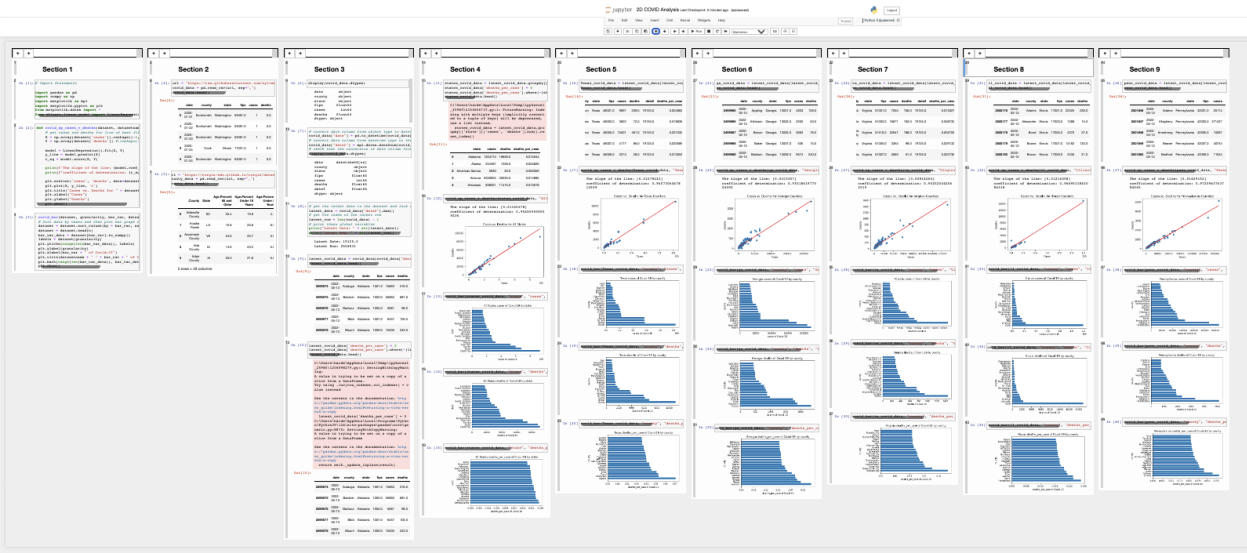

Figure 5.1: Finding & Comparing Results 2D Notebook from Study 1

5.1 Introduction

Computational notebooks like Jupyter [30, 38], used to construct and present computational narratives [55, 62, 66], struggle with non-linear analyses, such as comparative analyses, and

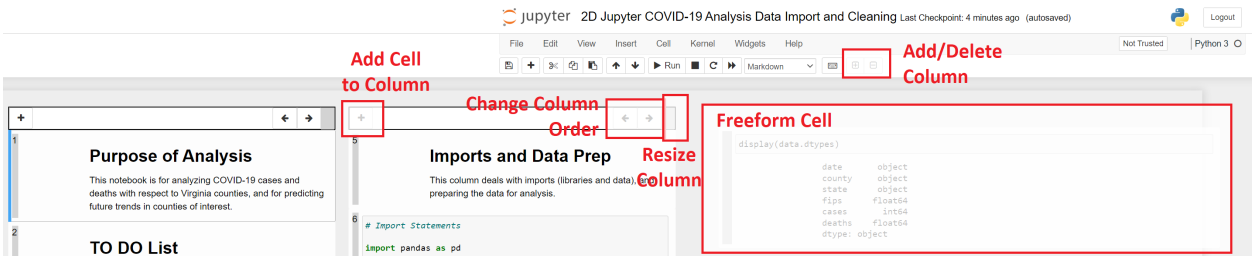

Figure 5.2: Notebook Controls for 2D Jupyter extension

non-linear narratives [22, 62], as well as navigating longer notebooks [22], preventing and managing messiness [15, 25, 32, 45, 47, 62], and efficiently using large display spaces [22]. We suggest that part of the reason for these issues is the current 1D, top-to-bottom organization of notebook cells.

Weinman et al.'s work on Fork-It [75] showed 2D space can be helpful; they introduced forking, the temporary creation of split columns in an otherwise 1D notebook. While this work helps non-linear analyses, it does not easily accommodate non-linear narratives, which may benefit from a persistent multiple column approach. Wang, Dai, and Edwards [73] also sought to shift computational notebooks from the current 1D structure with Stickyland, which allows users to "stick" cells to a dock that is persistently at the top of the computational notebook interface even when scrolling. Harden et al. [22] explored how users would arrange cells in 2D and found three different patterns: linear (with either split cells or split columns), multi-column, and workboard. This work demonstrated alternative organizations of cells, some of which would not be possible in the prior works mentioned; it also suggests that computational notebook users could benefit from 2D space usage for organizing notebook cells in a more flexible yet persistent manner.

This paper contributes to computational notebook research through evaluations of a 2D layout extension for computational notebooks. We focused on the following research questions:

1. When comparing 1D and 2D layouts, which mode supports more efficient user comple-

tion of data science tasks, such as information retrieval, results comparison, parameter tuning, and code comparison?

2. What strengths & weaknesses might 2D have compared to 1D?

3. Would users find 2D layouts more usable than 1D layouts?

4. Would users prefer to use 2D for computational notebook cells?

To answer these questions, we designed a Jupyter Notebook extension that enables a 2D multi-column cell layout. We then conducted two user studies using this extension where users performed a series of tasks in both 1D and 2D layouts, followed by qualitative data gathering through surveys and, in the second study, interviews. The first study used pre-made notebooks to evaluate whether the extension enhances performance and usability, while the second study focused on creation of a 2D notebook from scratch for a data science task. We found 2D layouts provided more efficient user task performance and enhanced usability over 1D layouts. Users overwhelmingly preferred the 2D notebooks, and made use of available display space to organize notebooks such that more cells are simultaneously visible. We also noted some design challenges for 2D layouts, including managing column width in a multi-column layout.

5.2 Background and Related Works

This work builds on two key areas of research: computational notebooks and Space to Think.

5.2.1 Computational Notebooks

Computational notebooks support incremental and iterative analysis [31, 62] and computational narrative formation through interleaving code, visualizations, and text[54, 62]. However, computational notebook users face various issues and pain points [11], such as messiness [25, 34, 47, 62], dealing with non-linear analyses and narratives [62], and navigating longer notebooks [22]. These issues may be exacerbated by the current 1D structure of computational notebooks.

Head et al. [25] showed messiness can come from disorder, deletion, and dispersal, where disorder means run order and presentation order are different, deletion means overwriting or deleting necessary code, and dispersal means related cells are far apart. Many tools have been developed to help deal with messiness, from Head et al.'s work [25], to cell dependency graph visualization [76] to version control systems for computational notebooks [32, 33]. The 1D structure may exacerbate messiness given the looping nature of sensemaking in computational notebooks [57, 59], so 2D space usage may help minimize it.

Scrolling through a long notebook can be tedious and negatively affect various tasks like debugging and cleaning. While Google Colaboratory [20] enables jumping to different sections through a table of contents, the 1D structure can still result in tedious scrolling.

Exploration of 2D space usage by Weinman et al. [75] and Harden et al. [22] produced positive responses. Within the bounded 2D of Fork-It [75], users did more than just comparative analyses; they used the split column structure to organize code and contain messes. Harden et al.'s [22] findings corroborate these potential use cases.

Computational Notebooks and Reproducible Science

Reproducible research is an important and challenging issue for any scientific endeavor, and research in HCI and computer science is no different [5, 12, 16]. At their best, computational notebooks and the computational narratives formed using them enable reproducible scientific workflows [4, 38, 63]; the ability to interleave documentation with code and results contributes to this potential. However, computational notebooks in the wild are rarely reproducible [56]; issues such as messiness [25], out-of-order execution [56], and dependency issues contribute to the pain of trying to reproduce computational notebook findings [11]. Indeed, less than 5% of notebooks studied by Pimentel et al. [56] were reproduced with the same results. Work to address issues with reproducibility, such as Osiris by Wang et al. [72], has helped; while our work does not directly focus on reproducibility, its ability to enable expanded use of space may indirectly help alleviate issues affecting reproducibility.

5.2.2 Space to Think

Andrews et al. [2] found large, high-resolution displays benefit sensemaking in 2 key ways through what they called "Space to Think": external memory and semantic encoding. External memory means more information can be stored on screen space instead of in one's mind, which allows physical navigation, like moving one's head, to replace virtual navigation, like scrolling or changing tabs. Semantic encoding means users can group related items spatially based on their mental model of the connection between items; in short, users can externalize their understanding onto the screen. Recent studies [13, 43, 44] have expanded this concept to the space provided by virtual and augmented reality or cited Space to Think as an influence on their design [53, 60, 61]. Kirshenbaum et al. [37] found Space to Think can also benefit collaborative meetings.

Current computational notebook systems with their 1D structures do not adequately use Space to Think without clumsy workarounds like opening the same notebook multiple times and arranging side-by-side. 2D space usage may enable Space to Think in data science tasks[22]. To this end, some recent tools, such as VisSnippets [9], Einblick [28, 65], CoCalc [29, 50, 70], and Code Bubbles [8], have begun to explore 2D layouts of cells using a whiteboard metaphor.

5.3 Design of 2D Jupyter Notebook Extension

Harden et al. [22] found two main categories of 2D layouts for computational notebooks based on user-generated layouts: multi-column and workboard, both of which are supported by the **2D Jupyter** extension we developed and evaluated; the extension and supplemental materials for this paper can be found at https://github.com/infovis-vt/2D-Jupyter on GitHub. Multi-column is fully supported. Workboard, or more complex structures such as directed graphs and nested columns and rows, is enabled by freeform dragging of cells. Given that the multi-column pattern in Harden et al. [22] was consistent across all its constructions and frequently constructed (32% of all participants), in addition to being part of several of the grouped combinations workboard sub-pattern, it makes sense to focus on enabling and evaluating the multi-column pattern first.

To support multi-column layouts, 2D Jupyter enables creation and deletion of columns, resizing and re-ordering of columns, adding cells to a column, and moving cells from one column to another; This is done through user interface (UI) additions, as seen in Figure 5.2. The Plus and Minus buttons on the main toolbar create and delete individual columns respectively. Also, each column now has a toolbar at its top; The bold Plus button here adds a cell to the column, the left and right arrows reorders the column in the arrow direction,

and the gray box can be clicked and dragged to resize a column. In addition, cells can be dragged to another column by clicking and holding the new gray box on each cell's left side. Finally, the Run All functionality is preserved in a top-down, left-to-right format; in other words, the leftmost column's cells are run in top-down order, followed by the cells in the column immediately to the right, repeating until all cells are run.

To enable workboard layouts, each cell can be dragged and placed outside of the columns, as seen in the freeform cell in Figure 5.2. More advanced workboard features, such as arrows to connect cells or other whiteboard annotations are not yet implemented. In addition, cells outside of columns are not run as part of the Run All functionality. For now, we suggest using workboard freeform cells for more ephemeral uses such as scratch space, viewing data, and other tasks not relevant to the final computational narrative.

5.4 Study 1 Methodology

The goal of our first study is to measure and compare user task performance in 1D and 2D notebooks. We therefore conducted a controlled study consisting of a pre-screening questionnaire, a set of user performance tasks, and survey questions. The study design had one within-subjects variable, ***layout*** with two treatments, ***1D*** and ***2D***; and one between-subjects variable, ***order*** with two treatments, ***1D-first*** and ***2D-first***. The user tasks focused on research question 1; participants completed three task sections in both 1D and 2D. For the surveys, we focused on research questions 2-4.

5.4.1 Recruitment and Screening

89 potential participants, recruited via academic listservs of students and faculty from a large state university, responded to an online screening questionnaire asking whether they had experience with both Python and computational notebooks such as Jupyter. 62 potential participants passed the screening due to having experience with both Python and computational notebooks and were notified that they could take part in the study. Of these 62, 31 chose to take part in the study. We discarded 1 of these 31 participants' data due to technical issues that arose during the study, leaving 30 participants. 15 of these participants were assigned to 1D-First, while the other 15 were assigned to 2D-First. Participants were randomly assigned to a group, with the only restriction being balancing the group numbers so that they were as equal as possible.

5.4.2 Hardware for User Study

For the user study tasks, participants used an iMac computer with a 24-inch monitor and either an iMac mouse with a built-in trackpad for horizontal and vertical scrolling, or an external trackpad with horizontal and vertical scrolling that also had buttons for clicking. The monitor was wide enough to display 4 to 5 columns of the notebook at a time.

5.4.3 Task Designs and Rationales

The tasks were designed to mimic common data science scenarios performed in computational notebooks. We created 6 computational notebooks (3 1D, 3 2D) for this study; the 2D notebooks used the multi-column pattern due to it being the most common, consistent pattern seen in Harden et al. [22] as mentioned in Section 3 and being fully supported by

our extension. Each notebook was designed for one of three task sets: Finding & Comparing Results, Parameter Tuning, and Code Comparison. Each layout (1D, 2D) and task set combo had one notebook, and each task set's notebooks were slightly different so participants could not memorize answers between layouts. However, the differences were designed to not impact difficulty between the tasks in 1D and 2D. Users had the notebooks open, one at a time, on the iMac, while the user study survey, with questions and instructions, was open on a separate laptop. Users were allowed to use any existing functionalities, such as searching for information using Control + F; we did not suggest such methods unless a participant asked.

To compare 1D vs. 2D, we measured the time it took a participant to answer the survey question and press the "Next" button on the survey as time to completion, and accuracy for all tasks was a count of correct answers; we also measured the number of times and amount of time spent scrolling for the code comparison task. 16 participants started with the 1D notebooks first, and 15 participants started with 2D first; This design, along with training in the first notebook layout type for each person, helped counterbalance the study to minimize bias from repeated tasks. One 1D First participant's data was discarded due to technical issues. Each participant took at most 1 hour to complete the study.

The 6 notebooks and a copy of the study session survey can be found at https://github.com/infovis-vt/2D-Jupyter on GitHub.

Finding and Comparing Results Task

Harden et al. [22] found that users expected finding and comparing tasks to be better in 2D layouts than in 1D layouts. Thus, this task set sought to measure statistically whether such a benefit exists.

The notebooks for this task set contained COVID-19 data analysis for the USA by state

and then for 5 individual states by county, as seen in Figure 5.1. Sections 1-3 of these notebooks had cells for imports, function definitions, and data preparation, while Sections 4-9 had cells that analyzed and visualized results for each geographic region as a scatterplot and 3 bar charts. In data science, such notebooks often result from copying-and-pasting cells for parallel analyses of different data subsets. The 1D notebook design concatenated these sections into a single long list of cells. In the 2D notebook, each of the 9 sections was separated into its own column of cells, with columns arrange left to right. This notebook design was based on common layout strategies previously observed by Harden et al. [22], where a common strategy was to organize parallel analyses in side-by-side columns to enable easy comparison.

For this task set, we included a find task, a graph comparison task, and a numerical comparison task. We did not allow participants to look over the notebook before beginning the task set.

In the find task, participants had to locate info in the notebook based on the notebook structure. The question was of the form *"Which state's analysis is found between the analysis of STATE1 data and the analysis of STATE2 data?"* We measured the time it took each participant to retrieve the info in 1D vs. 2D notebook layouts. The hypothesis was that spatial 2D columns would enable more rapid recognition and access to relevant notebook sections.

In the graph comparison task, participants had to compare results in several different charts throughout the notebook. The question was of the form *"Out of those shown in the relevant bar charts, which county in which state, EXCLUDING the ALL STATES section, had the highest number for ATTRIBUTE of COVID-19?"* We measured the time it took each participant to compare charts in 1D vs. 2D notebook layouts. The hypotheses was that 2D column structure that aligned parallel analyses would enable faster comparison by hori-

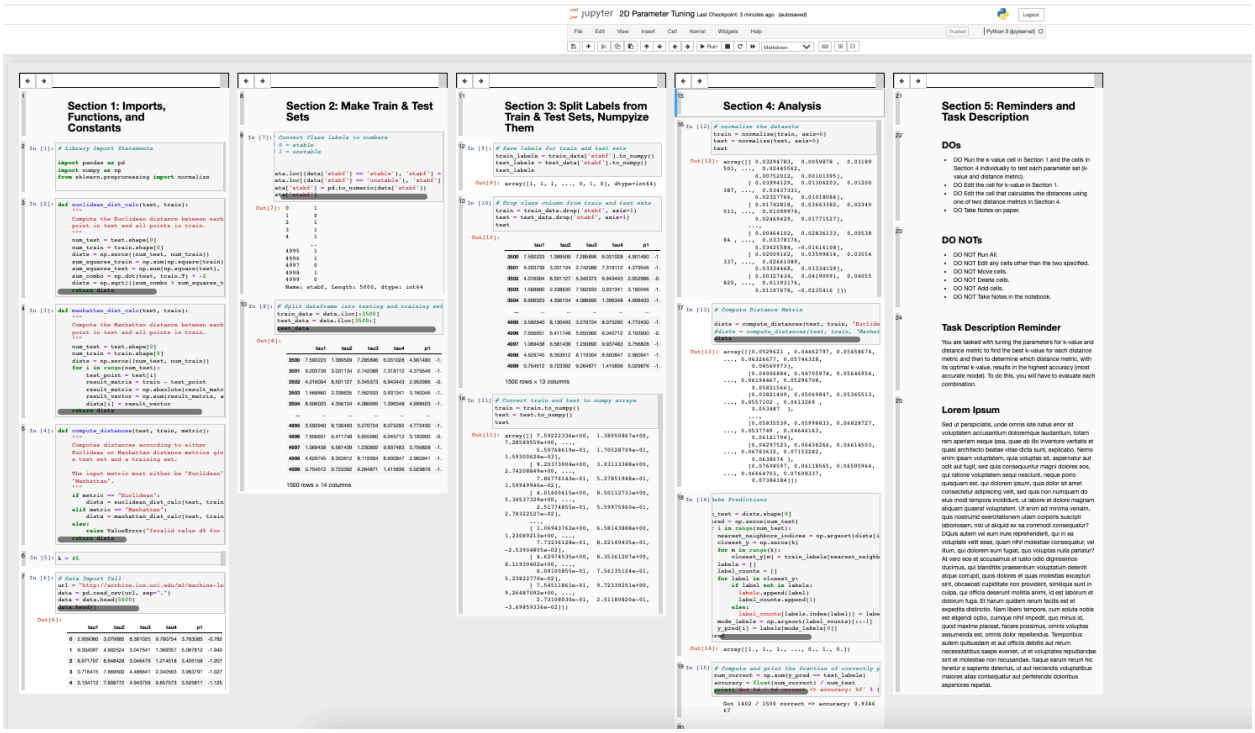

Figure 5.3: Parameter Tuning 2D Notebook

zontally scrolling through the corresponding charts, whereas the 1D notebook would require significant vertical scrolling and searching for each chart to compare.

Similarly, in the numerical comparison task, participants were asked a question of the form *"Which section's scatterplot graph's line of best fit least/best fits the data (coefficient of determination closest to 0/1)?"* The coefficient of determination was displayed above each scatterplot. We measured the time it took each participant to compare numerical results in 1D vs. 2D notebook layouts.

Parameter Tuning Task

A common problem in data science involves testing various parameter values for an ML model. The notebooks for this task, as seen in Figure 5.3, contained K-Nearest Neighbors

(KNN) algorithm used to analyze network stability data. Participants were instructed the following: "You will be asked questions that require tuning the parameter 'k' in Section 1 and choosing the distance metric in Section 4. Only run the necessary cells (the "k-value" cell in Section 1, and the cells in Section 4) to test each possible parameter set (k-value and distance metric)." In each notebook, the cell which assigns the k-value was in the first section while the code for calculating the distances, making predictions, and determining accuracy on the test set were in the fourth section; participants were not allowed to move cells. Participants were asked three questions in the following order, with different k-value options for 1D and 2D:

1. Which of the following k-values produces the most accurate model with the given dataset for the Euclidean distance metric?

2. Which of the following k-values produces the most accurate model with the given dataset for the Manhattan distance metric?

3. Given each distance metric with its optimal k-value, which distance metric produces the most accurate model on the given dataset?

In data science endeavors, code near the beginning of a notebook can influence results later on in the notebook; While it is possible to move such dispersed cells closer to each other, such re-ordering is not always feasible depending on the design of the analysis. Thus, we sought to simulate a situation in which one wants to retain the given order while continuing their analysis. The goal here is to see if 2D notebooks, with a layout where each section has its own column, can minimize the effects of dispersal [25] by making cells that are far apart in a 1D layout effectively closer on the screen in a 2D layout and lead to performance improvements. Thus, we measured how long it took participants to answer all three questions together.

Code Comparison Task

Data scientists often need to compare the code for multiple versions of a model to understand differences. The notebooks for this task, as seen in the right image in Figure 5.4, contained two runs of a K-Nearest Neighbors ML algorithm with several code differences between them. Participants had to choose which items from the list of options, ordered in terms of appearance, differed between each run. The 2D notebook organized the two runs into adjacent columns. The list of differences included items such as the following:

1. The cutoff number for the training and testing splits
2. Different distance metrics (Manhattan, Euclidean) used
3. The variable name for the distance matrix
4. The value of k (number of nearest neighbors)

The goal of this task was to test how quickly users can find differences between two similar sets of code, which often happens when debugging model errors. Given that Harden et al. [22] found significant skepticism about the potential of 2D notebook layouts for debugging, it makes sense to test this important debugging sub-task.

5.4.4 Survey Questions Design

Likert-scale questions were used at the end of both the 1D and 2D task sections, and after both sections were completed. The 5 questions at the end of the 1D and 2D task sections focused on rating each layout individually, without comparison to the other, while the 13 questions at the end focused on comparing 1D and 2D layouts; these 13 questions were

largely taken from Harden et al.'s experiment [22]. After the 13 questions was a comment box where users could elaborate on any answers they gave.

The questions after each of the 1D and 2D task sections focused on perceptions of usability for the layout on the given tasks; We compared their answers between layouts to better understand whether users saw potential improvements in 2D layouts over 1D layouts.

5.4.5 Data Analysis Process

We divided the quantitative data analysis for Study 1 into 3 areas: Efficiency Measurements, Survey Questions, and Scrolling Time.

Efficiency Measurements

We started our analysis of efficiency using 2-Factor ANOVA. However, due to completion time being log-normally distributed and thus violating the assumption of normality for ANOVA, we also used the Scheirer-Ray-Hare Test [64], a non-parametric alternative to 2-Factor ANOVA and extension of the Kruskal-Wallis Test [42], to test if layout (1D or 2D), as well as order (1D First, 2D First) and interaction between layout and order, affected time to completion; significant results were followed up with calculations of the differences between the means and medians of the 1D and 2D layouts to determine whether the 2D layout resulted in more efficient performance and to calculate average time saved as a percentage. R [26] and the packages RCompanion [48] and FSA [52] were the main tools in this analysis.

Accuracy was measured by counting the number of questions answered correctly in each layout by all participants and then dividing by the multiplication of the number of participants and the number of questions.

Survey Questions

For the Post-1D and Post-2D questions, we created and analyzed a bar chart of average rating by order and layout, and a heatmap of ratings. We also tested the statistical significance of the differences in ratings using a paired t-test and the Wilcoxon test, inspired by work by De Winter and Doduo on analyzing Likert-Scale questions [14]. For the Post-Experiment questions, we made and analyzed a heatmap of ratings. The qualitative comments were analyzed for themes using open coding by the first author. After the initial pass, feedback from the other authors was sought on the themes and used to refine them. Then, a second pass was made with all quotes grouped into themes.

Scrolling Time

To determine the amount of scrolling done in 1D vs 2D, we recorded scrolling events, including the time taken to scroll, while watching the footage for each participant's Code Comparison task work in 1D and 2D. We limited events to scrolls for navigation as opposed to micro-scrolling events that do not bring new cells into view; we did this by only considering those scrolling events that lasted for at least 2 seconds. To determine scrolling endpoints, we looked for breaks between scrolls lasting at least 2 seconds; scrolling events with smaller breaks than 2 seconds were considered as 1 event for the purpose of this analysis.

5.5 Study 1 Results

We divide our results into 4 areas: User Interaction Strategies, Efficiency Measurements, Survey Questions, and Scrolling Time.

5.5.1 User Interaction Strategies

Our observations of user behaviors with the 1D and 2D layouts, divided by task notebook, are summarized here.

Finding and Comparing Results Task

In 1D, all users started by scrolling down through the notebook to answer the Find question (which state's section was between two other states' sections) until they found the answer. Then, they scrolled through Sections 5 through 9 to answer the graphical Comparison question (which county in which state had the highest value for a particular variable) and compared the bar chart results and axes, which was sufficient to find the highest value. Some users, because they forgot a previous value or wanted to verify their memory, would scroll back to earlier results, sometimes multiple times, before submitting an answer. A couple users took notes on paper to avoid this issue. For the numerical comparison question, users repeated the process for first Comparison question with Sections 4 through 9.

In 2D, all users started by scrolling to the right to answer the Find question. Since the columns for the relevant sections (4-9) were fairly well aligned, as seen in Figure 5.1, this mitigated the need to perform vertical scrolling except for between questions. Users scrolled less distance in 2D due to more efficient use of space with 1 column representing 1 section. Then, to answer the 2 Comparison questions, all users used physical navigation (e.g. head movement) with less scrolling needed, since the screen could show 4 columns. The efficient, well-organized use of 2D also led users to perform less backtracking, if any, and eliminated the need to take notes on paper.

Parameter Tuning Task

In 1D, all users repeatedly scrolled up and down to get results for different parameter combinations (k-value and distance metric). Sometimes users scrolled past the cells they were looking for and thus did additional scrolling to correct their focus. All users took notes on paper so they could remember and compare results.

In 2D, much smaller scrolls were needed to get from the first column, where the main parameter was, and the fourth column, where results were calculated. Given the much smaller scrolling distance, scrolls were quicker and did not result in scrolling too far nearly as often. All users also took notes on paper with 2D, as well.

Code Comparison Task

In 1D, all users scrolled up and down to find code differences in the two different analyses; users examined the code in a cell in the first analysis, then scrolled down to examine the code in the corresponding cell in the second analysis before scrolling back up again to look at the next cell. This process was repeated until all potential differences were checked for. Since users were given a list of potential differences in order of appearance, they knew what to look for; this could have resulted in less forgetting (and thus less re-scrolling) than might otherwise happen.

In 2D, the two analyses were nearly horizontally aligned, so all users used physical navigation to find differences instead of virtual navigation; scrolling was used to go further into the notebook rather than to spot differences. As expected, in 2D users scrolled much less than they did in 1D due to the use of physical navigation and externalized memory on the screen.

5.5.2 Efficiency Measurements

Table 5.1: P-values for Scheirer-Ray-Hare by Task and Effect

Task	*Order*	*Layout*	*Interaction*	*Mean*	*Median*
Find	0.137	0.054	0.594	N/A	N/A
Graph Comparison	0.255	**4.2e-5**	0.131	**32%**	**45%**
Number Comparison	0.559	**5.8e-6**	0.156	**46%**	**34%**
Parameter Tuning	0.779	**0.003**	0.487	**19%**	**23%**
Code Comparison	0.882	**3.5e-4**	**4.6e-4**	**34%**	**33%**

Bolded values are statistically significant with a 0.05 threshold. All other values are not statistically significant.

As seen in Table 5.1 and summarized in Figure 5.6, we found the layout (1D or 2D) was statistically significant for all tasks except the find task in both 2-Factor ANOVA and in the Scheirer-Ray-Hare. The lack of significance for the find task may be due to it being a "cold find", one without prior knowledge of the notebook, which fails to make use of the benefits of Space to Think. The interaction between layout and order (1D First or 2D First) was significant for the code comparison task.

Analysis of mean and median differences showed the 2D layout resulted in statistically significant improvements to efficiency, summarized in Table 5.1; these improvements ranged from about 20-50% time reduction. These results likely reflect faster navigation of numerous code cells during the data science tasks when the cells are organized into columns.

Accuracy Measurements

The accuracy for 2D and 1D, measured in the number of correct answers given across all participants, was similar for 1D and 2D. 96% of questions in 1D were answered correctly, compared to 98% of questions in 2D.

5.5.3 Survey Questions

We divide the survey question results into 3 areas: Post-1D & Post-2D Questions, Post-Tasks Questions, and Qualitative Comments.

Table 5.2: Post-2D minus Post-1D Average Differences in Rating

Question	*Mean*	*Median*
Easy to Navigate	**1.87**	**2.00**
Quickly Find Info	**1.80**	**2.00**
Easy to Compare Graphs	**2.87**	**3.00**
Easy to Compare Numbers	**2.83**	**3.00**
Easy to Compare Code	**3.57**	**4.00**

Bolded values are statistically significant with a 0.05 threshold for both paired t-test and Wilcoxon. Positive values indicate 2D is considered better.

Post-1D and Post-2D Questions

As seen in the bar chart in Figure 5.7, the heatmap in Figure 5.5 and the results of Table 5.2, the user impressions of the usability of 2D layouts were significantly more positive than the 1D layouts on all metrics. Users rated 2D approximately 2-4 points higher (on a 7-point likert scale) than 1D on each metric. Users were nearly unanimously positive in rating 2D, and more evenly divided between positive and negative for 1D. Two participants gave the three negative ratings for 2D in Figure 5.5; one saw clutter in 2D notebooks as a potential issue, and the other felt the 2D notebooks could be improved by snapping cells next to each to ensure proper alignment of related cells.

Interestingly, as seen in Figure 5.7, participants exposed to 2D before 1D rated the usability of 1D as significantly worse for the usability questions asked. Thus, exposure to the 2D layout makes the 1D layout seem less usable.

Post-Experiment Questions

As seen in the Figure 5.8 heatmap, when explicitly asked to compare their experiences with the two layouts, participants overwhelmingly viewed 2D as more effective for common data science tasks, especially comparisons, and felt the 2D layout improved their performance. They also agreed that 2D made better use of screen space, and that this was key to their success. Furthermore, most participants seemed interested in using 2D layouts instead of 1D layouts, with only one participant expressing neutrality.

One curious result is that participants expressed skepticism about 2D layouts being better for presenting computational narratives and collaborating with others. Harden et al. [22] found the opposite; debugging, analysis and development, and navigation were seen as weaknesses of 2D layouts, while presentation and collaboration were seen as strengths. This difference may be due to the tasks that users performed in each study; presentation was key for Harden et al. [22], whereas debugging and comparison were key in this study.

Qualitative Comments

Of the 27 participants who left a qualitative comment on the survey, 20 expressed positivity about the 2D multi-column layout, while only 2 expressed that they might still prefer 1D notebooks for any task. 2 participants went so far as to express sentiments suggesting that the multi-column 2D layout makes 1D obsolete. 6 participants also left thoughtful feedback that may inform design of future 2D computational notebooks. Several comments pointed out the link between memory and navigation, that more time scrolling in 1D led to more forgetting important information for the task. The results are summarized in Table 5.3 with all of the themes found, a sample quote for each sub-theme, and the number of comments matching the theme.

5.5.4 Scrolling Time

For the code comparison task, we found participants scrolled more times and spent more time scrolling in 1D, as seen in Table 5.4; the average scroll event in 1D tended to be longer than those in 2D, as well. Given differences in typical user interactions described earlier, specifically the elimination of the need to scroll and reduction of scrolling distances for comparison, it makes sense that the 2D layout would have much less scrolling time and events for the Code Comparison task. This confirms that reducing scroll navigation is an important factor in enabling the faster performance results of 2D. This may be due to multi-columns bringing cells nearer to each other and fitting more cells on the screen simultaneously.

5.6 Discussion

We divide our discussion into the following categories: task efficiency benefits, usability benefits, effects of hardware, design challenges and opportunities, and limitations of our work.

5.6.1 Task Efficiency Benefits

The multi-column 2D computational notebook layout provides benefits to task efficiency by reducing the amount of scrolling necessary and shortening the length of needed scrolls. As seen in Study 1, the multi-column layout provided statistically significant reductions in time to completion overall. Given how much less scrolling was done in terms of total scrolling time, number of scrolling events, and average scrolling time in the multi-column layout, per Study 1's Scrolling Time analysis, combined with the time to completion results, the

multi-column layout clearly provide benefits to efficiency.

The reduced scrolling is a result of 2D's ability to bring more cells nearer to each other. Theoretically, 2D can reduce distances by the square root of 1D distances. Practically, 2D enables non-linear code structures, such as parallel analyses, to be horizontally aligned in columns, thus supporting common data-science tasks such as comparison. 2D enabled more such relationships to be encoded into the space. In contrast, 1D encodes only a single ordering, and would require complex refactoring tools to enable various types of parallel analyses and comparisons.

5.6.2 Usability Benefits

The multi-column layout appear more usable for certain basic and more complex tasks. Based on the results from Study 1 as seen in Figures 5.5 and 5.8, navigating and finding information, comparing results, and data science tasks such as organizing and cleaning may be easier in a multi-column notebook. This may be due to more effective use of screen space to display more information at once in an organized manner, along with more efficient scrolling options.

5.6.3 Design Challenges & Opportunities

While multi-column 2D computational notebooks may provide efficiency and usability benefits, especially with the right setup, there is still room for improvement on their design, especially as it relates to managing column width and navigating the notebook.

Column width in the multi-column design pattern may impact user experience; if the columns are too wide, fewer columns will fit on the screen, but if the columns are too narrow, visuals

may become too small to easily read and the screen may feel cluttered, potentially leading to confusions that affect performance. Thus, managing column width becomes an important factor; this is currently doable in 2D Jupyter through manual resizing of columns. Still, it may be beneficial to provide functionality that resizes columns to an ideal width through a quick interaction, such as is done in spreadsheets.

Additional navigation options tailored to different 2D layouts may also benefit users. Navigating 1D computational notebooks with arrow keys can be quicker than navigating with manual scrolls, and the same may apply to 2D computational notebooks; the challenge is whether and how to incorporate the left and right arrow keys (or even diagonals) to quickly navigate. One option is to borrow the grid metaphor and have each arrow key move to the adjacent cell in the direction of the key. Making individual columns independently scrollable may also benefit navigation, especially when working on smaller screens. This would allow longer columns to be scrolled without impacting the view of shorter columns.

5.6.4 Limitations

Our work has some limitations, especially as it concerns bugs in our extension and lack of sufficient support for 2D organizational patterns other than multi-column at the time of the studies.

Bugs in Extension

At the time of conducting both studies, the 2D Jupyter extension contained some bugs that could affect user experience. In particular, the drag and drop feature occasionally did not allow the user to release the cell at an intended location, forcing the user to reload the page. Additionally, the layout of the 2D environment was sometimes not properly saved

between kernel sessions, requiring the user to reorganize their notebook before resuming work; the extension required users to manually save their work as the autosave feature built into Jupyter Notebooks did not work with the extension. These bugs did not affect Study 1 except for contributing to the technical issues that led to discarding one participant's data.

2D Layouts other than Multi-Column

Given that both studies used an extension which does not, at the time of this writing, fully support 2D layouts other than multi-column, care must be taken in assigning benefits to other 2D layouts. Some of the advantages of the multi-column layout may be due to how compact it is; less compact 2D layouts might not see the same level of benefits in some areas, like reduced scrolling and task efficiency, without specialized navigational tools. Evaluating other 2D layouts is a subject for future work.

5.7 Conclusion

Computational notebooks are a potent tool for creating and presenting computational narratives; the 1D layout of notebooks, while elegant in its simplicity, imposes certain limitations that make comparative analyses and navigating longer non-linear notebooks, among other tasks, more difficult. Thus, we developed an extension and evaluated the potential of 2D for computational notebooks, starting with the multi-column layout enabled in our 2D Jupyter extension.

The multi-column 2D layout provides benefits in efficiency and usability for common data science tasks such as comparative analyses by enabling greater physical navigation, thus minimizing the scope and need for virtual navigation (scrolling). In addition, the multi-column

layout provides an effective sectioning mechanism that may help combat messiness along with providing more efficient navigation. While our conclusions are limited to the multi-column layout, 2D layouts may improve upon the current state of computational notebooks and provide a novel way to enhance the creation and presentation of non-linear computational narratives through enabling Space to Think.

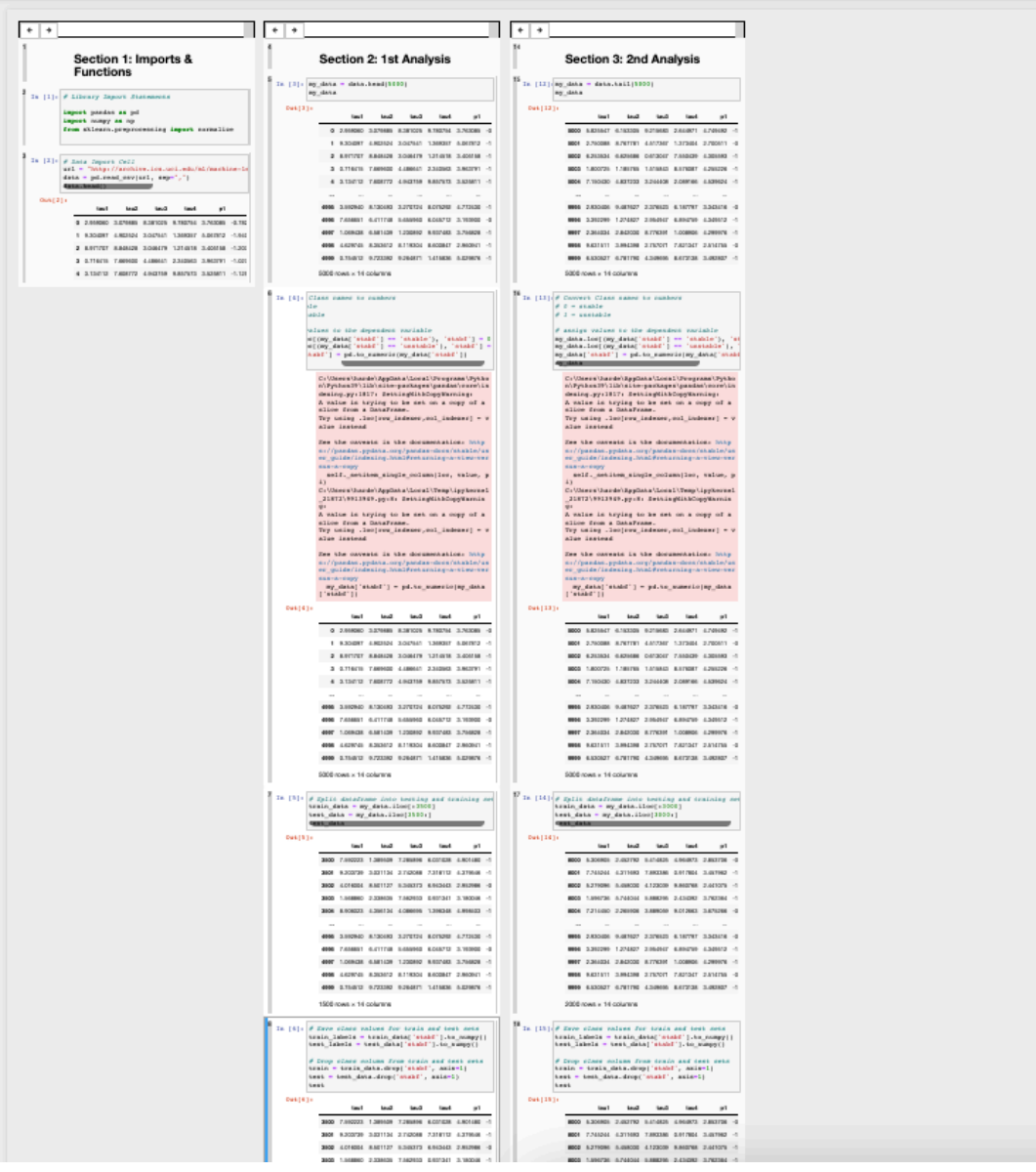

Figure 5.4: Code Comparison 2D Notebook Clip

Item		Rating						
Layout	Question	Strongly Agree	Agree	Agree a little	Neutral	Disagree a little	Disagree	Strongly Disagree
1D	Easy to Navigate	4	8	6	1	7	3	1
	Quickly Find Info	4	5	7	6	5	2	1
	Easy to Compare Graphs	3	4	5	1	5	10	2
	Easy to Compare Numbers	1	7	3	0	8	7	4
	Easy to Compare Code	0	4	5	1	5	8	7
2D	Easy to Navigate	18	8	4	0	0	0	0
	Quickly Find Info	16	13	0	0	0	0	1
	Easy to Compare Graphs	19	9	2	0	0	0	0
	Easy to Compare Numbers	19	6	4	0	0	1	0
	Easy to Compare Code	24	4	0	1	0	1	0

Figure 5.5: A heatmap comparing the ratings for the Post-1D and Post-2D questions.

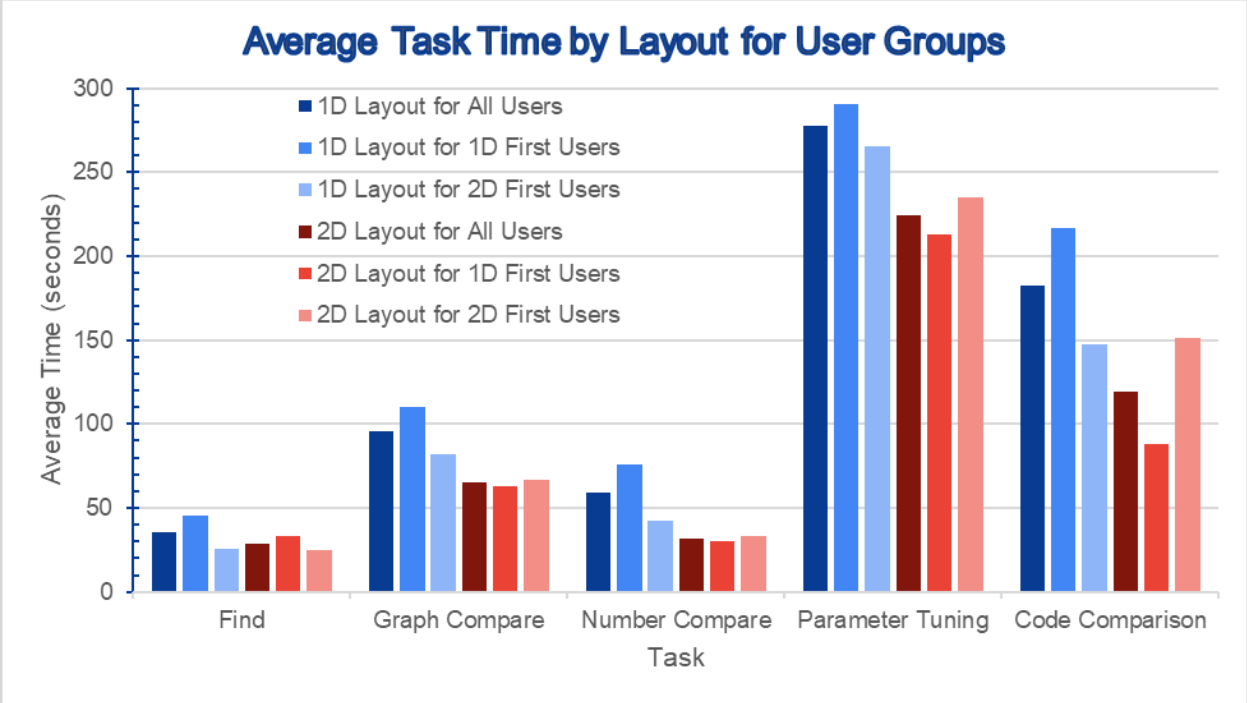

Figure 5.6: A bar chart showing average time to completion by task and layout in seconds.

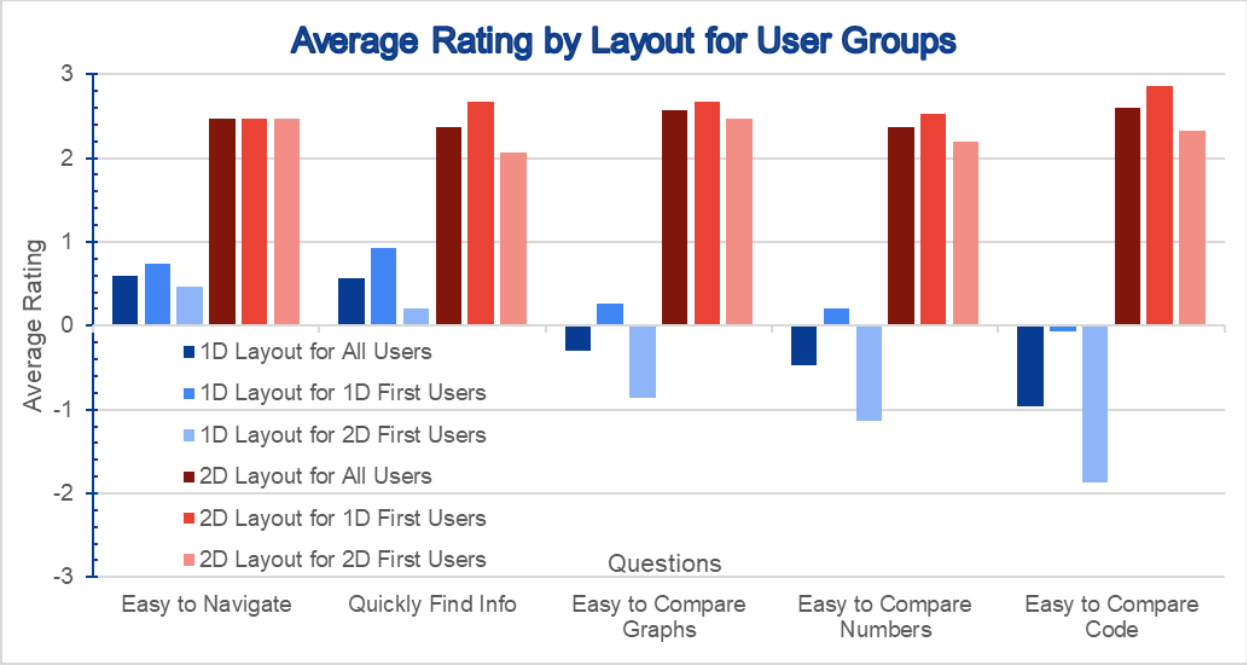

Figure 5.7: A bar chart comparing the mean ratings for the Post-1D and Post-2D questions; positive values indicate agreement with the sentiment, while negative values indicate disagreement.

Item		Rating						
Category	Question	Strongly Agree	Agree	Agree a little	Neutral	Disagree a little	Disagree	Strongly Disagree
2D Better than 1D at <task>	Navigate	15	7	2	4	2	0	0
	Locate Items	17	8	4	1	0	0	0
	Organize & Clean	10	10	5	3	2	0	0
	Present	9	7	3	7	3	1	0
	Explore & Prep Data	12	12	5	0	1	0	0
	Analyze & Develop	14	9	1	5	1	0	0
	Debug code	12	11	4	1	2	0	0
	Compare results	27	3	0	0	0	0	0
	Collaborate	8	8	3	9	2	0	0
Statements about 2D Layout	2D Spatial Layout Improved Performance	19	7	4	0	0	0	0
	More Cells on Screen in 2D Improved Performance	18	8	3	0	1	0	0
	2D Layouts Better Used Screen Space	21	6	3	0	0	0	0
	Would Use 2D instead of 1D	17	10	2	1	0	0	0

Figure 5.8: A heatmap visualizing the ratings for the Post-Tasks questions.

Table 5.3: Qualitative Themes in Study 1 Survey

Theme	*Sample Quote*	*Count*
Positive Comments on 2D		**20**
Better Comparison in 2D	"[2D] seems like a solid choice for a lot of analysis applications where you want to do similar but slightly different processes and compare the results."	7
Better Navigation in 2D	"It was more intuitive and easier to compare side-by-side sections compared to having to scroll so much. I spent so much time scrolling in [1D] that I forgot what I had looked at previously."	6
Practice with 2D Would Help Improve Performance	"This was my first experience with 2D notebooks after extensive use of 1D notebooks, so the advantages would be compounded given more time to familiarize myself."	3
2D is Better Than 1D	"There is no reason anybody should be using 1D anymore."	2
Other	"We always have to run multiple iteration with different parameter to calculate results and so 2D makes it very easy to see our progress in the notebook and also can be easily inferred."	2
Thoughtful Feedback on 2D		**6**
Column Width & Amount	"Putting too many columns in one screen caused [confusion] and potentially [increased scrolling]."	2
Arrow Key Navigation	"I found the 2D notebooks were more quick to navigate, but it was easier to navigate the 1D notebook using keys rather than the mouse, which might have been a little bit faster."	1
Cluttering Screen Space	"I believe one of the only things I might do in a 2D notebook that wouldn't be as easy would be displaying some visuals, as the layout would make them smaller, along with the text. Also having two visuals right next to each other might be seen as cluttered."	1
Use with Lower Resolutions	"The 2D notebooks were definitely easier to use, but for some tasks/cases (such as presenting on a [low-resolution monitor], or collaborating with... a low-resolution monitor) that might change."	1
Setup Time	"The only downside I could see is it taking slightly more time to initially set up..."	1
Skepticism about 2D		**2**
Presentation Skepticism	"[1D] looks more clean if you were to present something to another person."	1
Debugging & Dev Skepticism	"For development and collaboration the linear 1d notebook would be easier to debug."	1

Table 5.4: Scroll Event Analysis Totals Across All Participants

Measure	*1D Layout*	*2D Layout*
Sum of Scroll Event Times	2071 seconds	561 seconds
Count of Scroll Events	410 events	195 events
Mean Time per Scroll Event	5.05 seconds	2.88 seconds
Median Time per Scroll Event	4 seconds	2 seconds

Chapter 6

Exploring the Potential of 2D Freeform Computational Notebooks with SAGECells

6.1 Introduction

Computational notebooks like Jupyter [30, 38] are popular tools for data science and analysis due to their ability to construct and present computational narratives [55, 62, 66]. However, their one-dimensional (1D), top-down organization of code cells impedes users' ability to effectively deal with nonlinear patterns in analysis and narrative [22, 62] and make the most of larger widescreen displays [22, 23], and may exacerbate other issues like messiness [15, 25, 32, 45, 47, 62]. One common pattern of nonlinearity in the iterative and exploratory work of data science is that which we call "Branching Code Paths", or semantically parallel sections of code; this can happen, for example, when a user copies and pastes (or overwrites) code with minor tweaks to see how performance changes, or when a user compares performance of several different machine learning (ML) algorithms on a single dataset. This pattern serves as context for our study.

Recent works have explored the design space for organizing computational notebook code

cells in two dimensions (2D) [22] and evaluated the use of simple, rigid organizational pattern for such cells [23, 73, 75]. Software like Einblick.ai [17, 19] and now Observable HQ with canvases [6] are moving away from rigid structures. However, the authors are unaware of any works evaluating a nonlinear 2D computational notebook where the user imposes structure rather than the software. Such a "freeform" nonlinear 2D notebook could potentially provide similar, if not greater, benefits to users as a rigid nonlinear 2D notebook. This work serves as a starting point for exploring the potential of such freeform nonlinear notebooks.

This paper contributes to research on the usability of computational notebooks through a brief design and thorough evaluation of a freeform 2D computational notebook interface, SAGECells, within SAGE3 [68, 69]. We focused on the following research questions:

- **RQ1:** When comparing linear and nonlinear strategies for code cells, which supports more efficient and accurate coding and comparison for Branching Code Paths?
- **RQ2:** What strategies do users utilize in 1D and 2D notebooks to deal with Branching Code Paths?
- **RQ3:** How do users organize cells in a freeform 2D notebook to deal with Branching Code Paths?
- **RQ4:** Are freeform 2D computational notebooks preferred over 1D computational notebooks, and why?

To answer these questions, we designed and implemented SAGECells, an application in SAGE3 [68] that can be freely organized. We conducted a user study using Jupyter Notebooks and SAGECells in SAGE3, followed by qualitative data gathering through surveys. We found SAGECells increase efficiency over Jupyter Notebooks when revisiting an analysis, and are similar in setting up and running an initial analysis. Users generally preferred

SAGECells over Jupyter Notebooks, and most made use of available display space with SAGECells. We also noted some design challenges for freeform 2D notebooks.

6.2 Overview of SAGECells in SAGE3

SAGECells are an application developed for SAGE3 [68, 69], an online canvas (or whiteboard) software for collaborative data science, that act as computational notebook cells. SAGECells can be moved and resized on the canvas and have its font size changed. SAGECells rely on language kernels (e.g. Python), managed by the SAGE3 Kernel Dashboard, with a JupyterLab backend; users can create several kernels and choose which kernel to run a SAGECell on at anytime using an application menu. Furthermore, multiple SAGECells, in the form of a single row or column, can be run in order by selecting them via a click-and-drag maneuver. This is done via a simple algorithm that detects whether the overall shape of the SAGECell selection is wide or tall; if it is wide, then it runs the SAGECells left-to-right, while if it is tall, it runs the SAGECells from top-to-bottom. In addition, a single SAGECell or a selection of SAGECells can be duplicated (code and output) into new SAGECells to the right of the SAGECell or a selection of them.

Some additional features of SAGECells that we did not focus on in this study include the following:

- Users can drag image output from SAGECells onto the board for storage.
- Users can download the SAGECell code as a Python script file.
- Users can upload Python script files to SAGE3 and drag them onto the board as SAGECells.

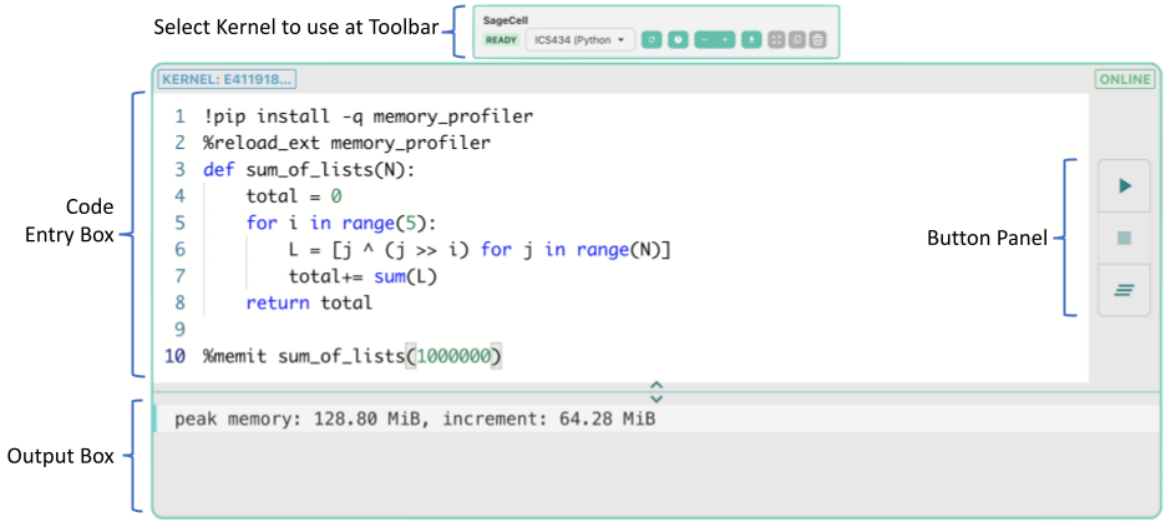

Figure 6.1: SAGECell Application containing the code section, the output section, the kernel and font size selection, and the execution control on the side.

6.3 Study Methodology

We ran a controlled study with a pre-screening questionnaire, two user study sessions in 1D (Jupyter) and freeform 2D (SAGECells) respectively, and a post-study survey. We compared performance and usability of the two different environments (1D, 2D) while also considering user strategies, and we included counterbalancing; about half the participants did 1D First and the others did 2D First.

6.3.1 Recruitment and Screening

We recruited 65 potential participants via academic listservs of students and faculty from a large state university, who completed an online screening questionnaire which asked how much experience they had with computational notebooks and Python. 52 had at least some experience in both Python and computational notebooks and were thus informed they could partake in the study. Of these 52, 19 took part in the study, with nine doing 1D First and

10 doing 2D First. Participants were randomly assigned such that the groups were evenly balanced.

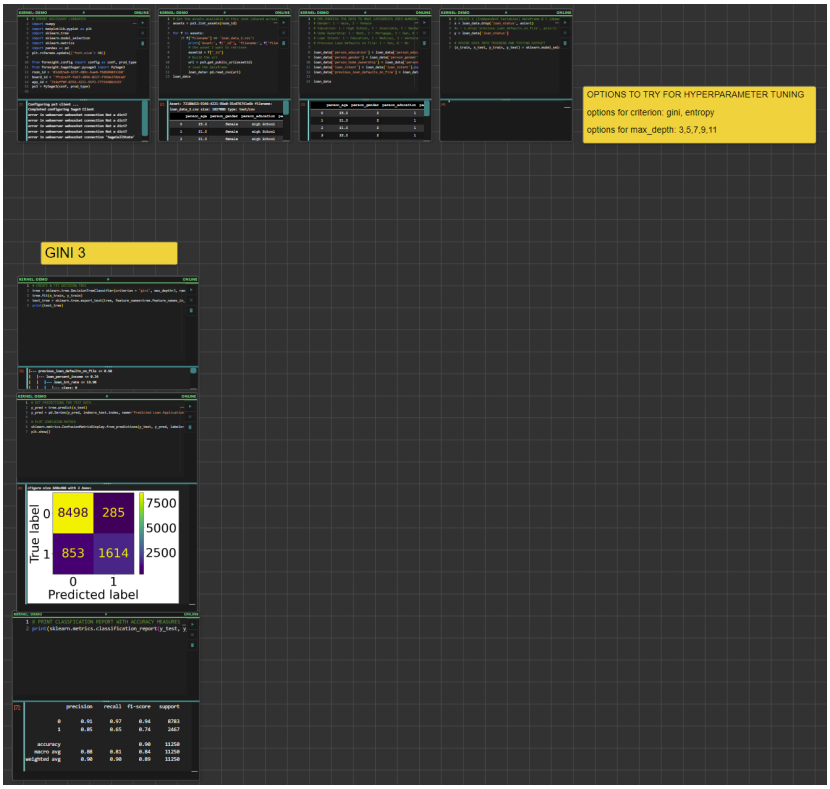

Figure 6.2: Starting Layout for the 2D SAGECells condition; the top row is data preprocessing and the column is the first branch.

6.3.2 Hardware for User Study

For the user study sessions, participants used an Alienware computer with a 28" high by 48" wide monitor with a resolution of 3840x2160 pixels. The computer had a corded mouse and keyboard. The monitor was wide enough to display six to eight columns of computational notebook cells at a time.

6.3.3 Task Design and Rationale

We designed the user study task to emulate a pattern of nonlinearity in data science, Branching Code Paths, that can cause or exacerbate user issues with standard 1D computational notebooks. Branching Code Paths means having sections of code that are semantically parallel in nature, if not in potential run order; results of such sections are often compared against each other. Two examples of this pattern are copying and pasting cells with minimal modification of code for comparing performance on a task (e.g. hyperparameter tuning) and testing various machine learning algorithms on one dataset.

We created two identical computational notebooks for this study, one in SAGE3 with SAGECells and one in a Jupyter Notebook. Each notebook had a hyperparameter tuning task with a loan classification dataset using a decision tree where users would tune a combination of criterion (Gini or Entropy) and max_depth; max_depth options were 2, 4, 6, 8, and 10 for the 1D notebook and 3, 5, 7, 9, and 11 for the 2D notebook. In addition, we split the dataset into two subsets; the 1D notebook used subset 1, and the 2D notebook used subset 2. We added these differences to prevent memorization of past results being a viable strategy while keeping the tasks practically identical. Other than that, participants could use whatever strategy they wished to complete the task. Finally, the notebooks' starting states included data pre-processing and one branch of the hyperparameter tuning task done (Gini 2, Gini 3); participants did not need to interact with data pre-processing cells for this study. The 2D notebook's starting layout, as seen in Figure 6.2, included a row and a column of cells for demonstrating SAGECell features in SAGE3 as part of a brief training on the system.

To compare performance in 1D Jupyter Notebooks vs. 2D SAGECells in SAGE3, we measured the time it took to answer the survey questions for the task and press the "Next" button on the survey as "Time to Completion." Accuracy was measured in terms of whether

the user selected all the right answers and whether the user selected any wrong answers. Each participant took at most 1 hour and 15 minutes for each study session, with one study session for the 1D Juypter Notebook and one for the 2D SAGECells in SAGE3.

Hyperparameter Tuning Task

A common task in data science is tuning the hyperparameters of a machine learning model to maximize performance. The notebooks for this task used a Decision Tree algorithm to classify loan applications, and involved two passes of this task. For the first or **Initial Pass**, we instructed participants to code and compare the 10 branches using whatever strategy they wanted, and we warned them that they would have to re-run their analysis later so as to minimize the effects of repeated tasks between sessions on the participants. We designed the first pass to see how participants would tackle Branching Code Paths and measure user performance in each layout setting. Participants could use whatever strategies they wished, such as move cells/change cell order, write new code (e.g. loops), duplicate cells, and more. For the second or **Revisit Pass**, participants could use what they had already done to re-run the analysis after the column the decision tree tended to split on first was dropped, or they could adopt a new strategy. In both tasks, participants were asked to select the best (highest) value for number of loans correctly approved, overall accuracy, precision for loans approved, and F1-score for loans denied. We designed the second pass to emulate revisiting an existing analysis with a changed dataset.

Branching Code Paths in Data Science

The task above is a simple example of Branching Code Paths; we chose it since we used a population of convenience (college students) and wanted all participants to be able to com-

plete the task with limited time. Still, this example could emulate more complex examples of Branching Code Paths in how users approach such tasks and how they might organize branches in 2D.

SAGECells with SAGE3 Training

Since participants had no experience using the 2D notebook, we provided 10 minutes of training for participants to learn to navigate the notebook, run individual cells and groups of cells, and duplicate cells. After training, users spent 5 to 10 minutes practicing with the 2D notebook and its features to become more comfortable with its interface and capabilities.

6.3.4 Survey Questions Design

We designed our surveys to evaluate user perceptions of the usability of 2D SAGECells in SAGE3 versus 1D Jupyter Notebooks to answer **RQ4**. We divide our discussion of survey questions into two areas: Session Surveys and Post-Study Survey.

Session Surveys

Both the 1D and 2D user study sessions had an accompanying survey with identical questions except for the layout referenced. After each pass of the user study task, users were asked to rate and explain their confidence level and explain the strategies they used. Finally, after completing both passes, users answered Likert-scale questions about the usability of that session's layout in relation to navigation, quickly finding information, coding and comparing Branching Code Paths, revisiting the analysis, and use of the large display.

Post-Study Survey

This survey was designed to gather user perceptions of the usability of 1D and 2D computational notebooks. It starts with a 7-point Likert-scale matrix question on which notebook type (1D or freeform 2D) participants would prefer to use for tasks such as navigation, dealing with Branching Code Paths, and data exploration and preparation, rated from Strongly Prefer 2D to Strongly Prefer 1D; following this is a 7-point Likert-scale matrix question with 6 statements participants rate from Strongly Agree to Strongly Disagree; 3 statements are pro-1D and three statements are pro-2D, with each kind being interleaved in the statement order. Participants could then elaborate on their answers to the first two Likert-scale matrix questions before proceeding. Following this on a new page was a 7-point Likert-scale matrix question on which notebook type (1D or 2D) participants would prefer to use for data science and analysis projects of varying length and type, rated from Strongly Prefer 2D to Strongly Prefer 1D. Participants could then elaborate on their answers to this matrix question. Next, on the final page were three qualitative answer questions focusing on differences in strategies for 1D and 2D notebooks, perceptions of the usefulness or uselessness of 2D notebooks, and what features users would like to see added to SAGECells and SAGE3. The final question gave participants an opportunity to share any last relevant thoughts.

6.3.5 Data Analysis Process

We divide the data analysis into three areas: Strategy Labeling, Performance and Usage Metrics, and Survey Questions.

Strategy Labeling

To help answer **RQ1** and to answer **RQ2**, we first labeled whether each user used a nonlinear or linear strategy with SAGECells in SAGE3 to complete the Initial Pass of the task; A linear strategy organizes cells in such a way that is completely replicable in a 1D, top-down notebook, whereas a nonlinear strategy makes use of more than one dimension in a way that cannot be replicated in such a notebook. "Linear strategists" were filtered out of the performance analysis after initial analysis. To determine whether a user's strategy was linear or nonlinear, we noted the following after the Initial Pass was complete:

1. We identified which cells the participant used to complete the task, which we call *relevant cells*.
2. We checked if the *relevant cells* formed a single line (row, column, diagonal) or if there was only one such cell.
    - If true, we labeled the layout strategy as linear.
    - If false, we labeled the layout strategy as nonlinear.

We focus on the Initial Pass because if a user switched to a nonlinear strategy in the Revisit Pass, their work would resemble an initial pass rather than a revisit and reuse exercise as intended in the study design. We also focus only on relevant cells to avoid classifying users who dabbled briefly with a 2D layout strategy only to retreat to a more familiar 1D strategy (e.g. loop in a single cell) as 2D. We did not require a 2D layout strategy so that we could gauge approximately what percentage of users might not make use of 2D space, and how those who made use of 2D space did so, to answer **RQ3** and to compare this to Harden et al.'s exploratory work [22]; the **RQ3** analysis considered the final layout after the second pass.

We labeled strategies used in 1D by users after their Initial Pass, to answer **RQ2**, into two mutually-exclusive groups: Overwrite, and Looping. **Overwrite** users overwrote relevant hyperparameters and re-ran code cells without adding functionally new code; **Looping** users wrote loops to show multiple branches' results in each loop code cell's output. We also noted whether users duplicated code into new cells. Finally, we tallied how many users took external notes outside the analysis screen (e.g. on paper) for both 1D and 2D.

Performance Metrics

For *Time to Completion*, given that counterbalancing was done to minimize the effects of order, and that completion time is generally not normally distributed (as noted in Harden et al. [21]), we chose to use a non-parametric version of the Paired T-test called the Wilcoxon Signed-Rank Test for Paired Samples.

For *Accuracy*, we ran a Paired T-test instead. If all the right answers were selected for a question, the participant received one point, and if none of the wrong answers were selected, the participant received one additional point, making two points possible per question for a total of eight points per pass, or 16 points total per session (1D or 2D).

Excel with the Real Statistics Resource Pack [79] was used to quickly generate analyses of the data.

Survey Questions

For the Post-1D and Post-2D questions, we created and analyzed a boxplot (see Fig. 6.8) of the ratings by layout (1D or 2D). Like Harden et al. [21], we tested the statistical significance of differences in ratings using a Paired T-test and the Wilcoxon test per De Winter's and Doduo's work on Likert-scale analysis [14]; we then calculated mean and median differences

Table 6.1: Strategies and Layouts Used in 2D Notebook

Item: Subgroup	1D First	2D First	Total
2D Initial Pass: Linear Strategy	2	4	6
2D Initial Pass: Nonlinear Strategy	7	6	13
2D Duplication Feature: Used	7	7	14
2D Duplication Feature: Unused	2	3	5
2D Layout: Linear	2	4	6
2D Layout: Multi-Column	2	4	6
2D Layout: Grouped Combinations	5	2	7

for statistically significant results. For the Post-Study questions, we created and analyzed heatmaps of ratings and analyzed qualitative answers for themes using open coding by the first author. After the initial coding, the first author got feedback from other authors on the themes to help refine them; the first author then recoded quotes and grouped all quotes into the chosen themes.

6.4 Results

We divide our results into three areas: Strategies and Layouts Used, Performance Metrics, and Survey Questions.

6.4.1 Strategies and Layouts Used

We divide results on strategies and layouts used into two sections: Strategies and Layouts in 2D, and Strategies in 1D.

Table 6.2: Strategies and Layouts Used in 1D Notebook

Item: Subgroup	1D First	2D First	Total
1D Initial Pass: Overwrite Strategy	8	6	14
1D Initial Pass: Looping Strategy	1	4	5
1D Revisit Pass: Overwrite Strategy	7	6	13
1D Revisit Pass: Looping Strategy	2	4	6

Strategies and Layouts in 2D

As seen in Table 6.1, most participants (68%) used a nonlinear strategy and layout; these participants all also used the duplication feature of SAGECells to quickly create layouts that helped them compare the results of different branches, and one of them combined code from the starting analysis cells into one cell before duplication. Review of layouts showed these participants organized code cells into two main design patterns found by Harden et al. [22]: Multi-Column, a single row of columns like in Fig. 6.3, and Grouped Combinations, a nesting of rows and columns to create more complex structures like in Fig. 6.4. The Grouped Combinations layouts avoided scrolling for the analysis task since they fit everything needed onto the screen; the Multi-Column layouts required some scrolling. No users made a Directed Graph layout, likely due to current limitations with SAGECells.

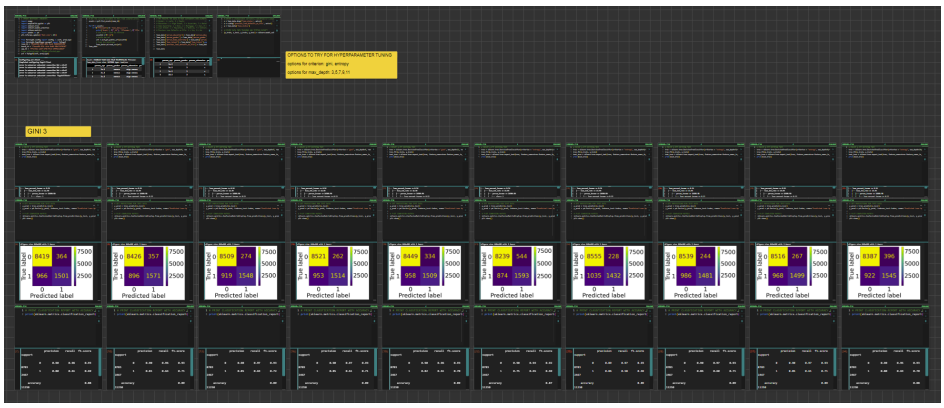

Figure 6.3: Example Multi-Column layout.

Of those who used a Linear Strategy, four used a single column, one used a single row, and

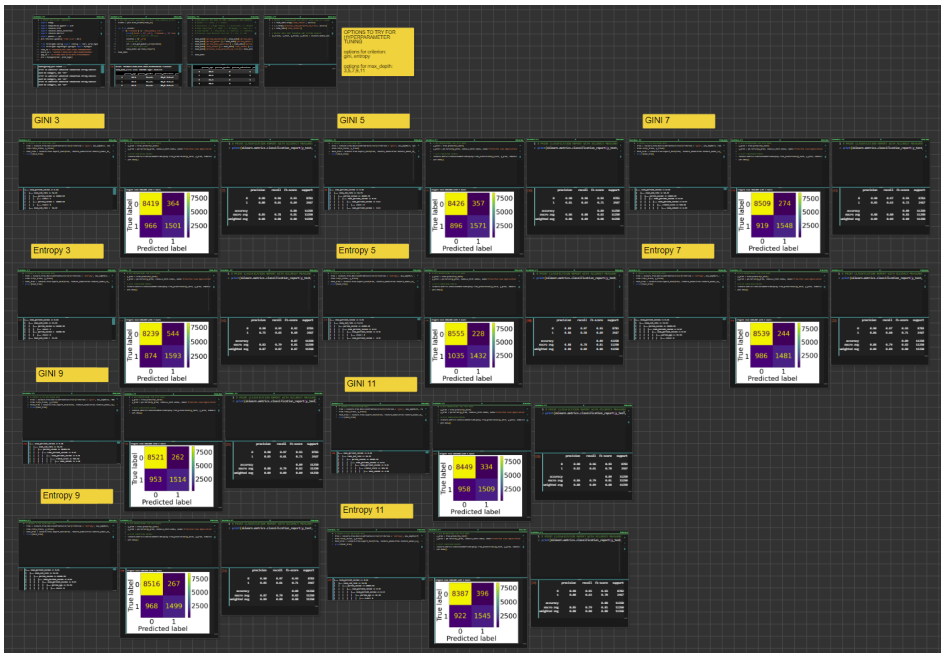

Figure 6.4: Example Grouped Combinations layout.

one wrote a loop in one cell to show all outputs. In the second pass in 2D, one linear strategist switched to using 2D space with a Grouped Combinations (nested rows and columns) layout, while the remaining five stuck with a Linear layout [22] and strategy; this change is not reflected in Table 6.1, as it focuses on the initial pass. Finally, it is worth noting that 2/3 of the Linear Strategists started with 2D, which may suggest that some users may struggle to decide how to use 2D space even after being trained.

Finally, about half of the users (9) took external notes with the 2D notebook environment.

Strategies in 1D

After the first pass in 1D using a Jupyter Notebook, we found that 14 of the 19 participants used the Overwrite and Re-Run strategy, while the remaining five used the Looping strategy. One of these participants created a custom function to generate all the outputs for each branch given the paramaters and then ran 10 cells with the function, one for each branch.

Table 6.3: Pre-Filtering Time to Completion Results

Value	1st Pass	2nd Pass	Both Passes
1D Mean	919.21	504.84	1424.05
1D STD	317.98	208.48	506.84
2D Mean	1012.95	429.68	1442.63
2D STD	383.87	190.47	527.59
1-Tailed T Test p-value	0.211	0.093	0.455
1-Tailed Wilcoxon p-norm	0.266	0.089	0.476

Table 6.4: Post-Filtering Time to Completion Results

Value	1st Pass	2nd Pass	Both Passes
1D Mean	909.08	480.00	1389.08
1D STD	221.40	157.14	349.75
2D Mean	874.15	338.31	1212.46
2D STD	279.76	125.62	376.34
1-Tailed T Test p-value	0.289	**0.014**	0.067
1-Tailed Wilcoxon p-norm	0.390	**0.015**	0.104

In the second pass in 1D, two of the Overwrite and Re-Run users switched to the Looping strategy. Finally, most users (16) took external notes with the 1D Jupyter Notebook.

6.4.2 Performance Metrics

We divide results on performance metrics into two sections: Accuracy and Efficiency.

Efficiency

As seen in Table 6.3, when considering all participants regardless of strategy, there was not a statistically significant difference in efficiency. After excluding participants who used a

Table 6.5: One-Tail Paired T-Test on Accuracy Results

Item	1D Mean	2D Mean	p-value
Pre-Filtering	14.94	14.11	0.092
Post-Filtering	15.08	14.58	0.194

TASK	Strongly Prefer 2D	Prefer 2D	Slightly Prefer 2D	Neutral	Slightly Prefer 1D	Prefer 1D	Strongly Prefer 1D	Median
Navigating through the notebook	4	3	4	2	4	2	0	Slightly Prefer 2D
Locating items (code, results) in the notebook	6	5	7	0	0	1	0	Prefer 2D
Dealing with Branching Code Paths	6	8	1	2	0	2	0	Prefer 2D
Organizing/cleaning a notebook	7	4	2	3	2	1	0	Prefer 2D
Presenting a notebook	7	5	2	3	0	1	1	Prefer 2D
Data exploration and preparation	4	5	2	5	2	1	0	Slightly Prefer 2D
Performing analysis and development	6	3	5	3	2	0	0	Slightly Prefer 2D
Comparing results	9	9	1	0	0	0	0	Prefer 2D
Collaborating on a shared notebook	5	4	2	6	0	1	1	Slightly Prefer 2D

Figure 6.5: Heatmap of Preferred Layout for Various Data Science Tasks.

CONDENSED STATEMENT	Strongly Agree	Agree	Agree a little	Neutral	Disagree a little	Disagree	Strongly Disagree	Median
1D Simplicity improved performance over 2D Complexity	1	2	4	1	4	4	3	Disagree a little
1D used available screen space better than 2D	0	2	0	3	5	5	4	Disagree a little
1D increased my confidence in my answers over 2D	1	1	1	9	2	3	2	Neutral
2D showing more code cells improved performance over 1D	8	7	3	1	0	0	0	Agree
2D setup time is worth it for benefits provided	7	7	4	1	0	0	0	Agree
2D notebooks are what would I would use instead of 1D	6	6	6	1	0	0	0	Agree

Figure 6.6: Heatmap of Agreement Level with Statements (Pro-1D statements first and then Pro-2D statements).

CONDENSED STATEMENT	Strongly Prefer 2D	Prefer 2D	Slightly Prefer 2D	Neutral	Slightly Prefer 1D	Prefer 1D	Strongly Prefer 1D	Median
Short, small data analysis project (few branches, iterations)	1	0	2	4	2	7	3	Prefer 1D
Longer-term data analysis project (many branches, iterations)	4	10	2	3	0	0	0	Prefer 2D
Project with Hyperparameter tuning	5	8	4	2	0	0	0	Prefer 2D
Project with comparing several different algorithms	7	8	2	1	1	0	0	Prefer 2D

Figure 6.7: Heatmap of Preferred Layout for Various Data Science Project Types.

linear strategy in 2D, we found that the mean time to completion for 2D was lower in the first pass, second pass, and both passes combined, as seen in Table 6.4. However, only the second pass had a statistically significant difference. The lack of significance for the first pass may be due to time taken to duplicate and arrange code cells in 2D space, rather than starting the analysis right away as in 1D.

Accuracy

As seen in Table 6.5, the accuracy was slightly lower on average in 2D in both pre- and post-filtering, contrary to Harden et al.'s results from 2D Jupyter [23]; this was not statistically significant in both cases, though. Still, this very slight drop in accuracy might suggest increased cognitive load or some other challenge in 2D.

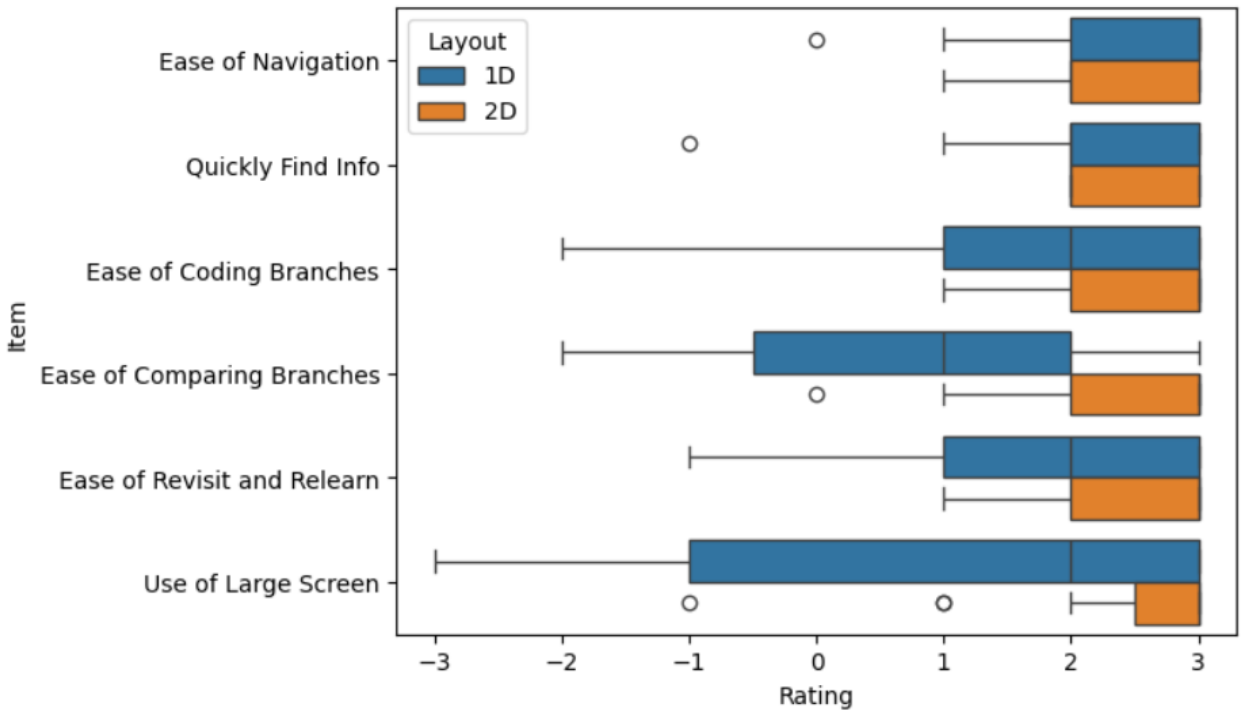

Figure 6.8: Boxplots of subjective usability measure results by layout (1D or 2D).

Table 6.6: Post-2D - Post-1D *p-values* and **Differences**

Item	*Paired T-Test*	*Wilcoxon*	**Mean**	**Median**
Navigation	0.607	0.593	-	-
Quickly Find Info	0.285	0.334	-	-
Coding Branches	0.097	0.098	-	-
Comparing Branches	0.008	0.013	1.421	1.000
Revisit and Relearn	0.039	0.045	0.789	0.000
Use of Large Screen	0.035	0.043	1.368	1.000

6.4.3 Survey Questions

We divide survey results into three areas: Post-1D and Post-2D Questions, Post-Study Questions, and Qualitative Comments.

Post-1D and Post-2D Questions

As seen in Fig. 6.8, the 2D environment (SAGECells) appears similar in ease of navigation and quickly finding info and appears better in ease of coding branches, comparing branches, revisiting and relearning, and in use of the large display. However, we found only comparing branches, revisiting and relearning, and use of the large screen were statistically significantly different in our study; in all three of these metrics, 2D was preferred, as seen by the positive values for mean for Post-2D minus Post-1D in Table 6.6. Of note is that (Ease of) Revisit and Relearn had the smallest difference, with a small mean and a median of 0.

Post-Study Questions

As seen in the heatmaps in Figs. 6.5, 6.6, and 6.7, the vast majority of participants had a positive view of the freeform 2D notebook environment. Per Fig. 6.5, the median results found 2D at least slightly preferable for each common data science task mentioned; no participants preferred 1D for comparing results, while six users preferred navigation in 1D.

Per Fig. 6.6, the median results never found 1D better than 2D, even with the pro-1D statements. Notably, participants felt they benefited from having more notebook cells on the screen, that setup time in 2D was worth it, and that they would use 2D notebooks over 1D notebooks. Still, seven participants found the simplicity of 1D notebooks beneficial. Per Fig. 6.7, participants largely preferred 2D for larger, more complex projects and 1D for short, small projects; only one participant preferred 1D over 2D for any of the three larger, more complex project types.

Qualitative Responses

Of the 19 participants, all gave some qualitative feedback; 17 participants expressed positive aspects of the 2D notebook environment, eight expressed challenges of the 2D notebook environment, 10 expressed drawbacks of the 1D notebook, and only one out of all 19 expressed a preference for 1D over 2D. Several comments point out how the 2D notebook environment with the large display effectively enables more externalized memory and spatial encoding of meaning, both key features of Space to Think [2], along with the increased efficiency from physical navigation [1] as opposed to the required virtual navigation in 1D. The results are summarized in Table 6.7 with the major themes found, a sample quote for each theme, and the number of participants who gave a comment matching the theme.

6.5 Discussion

We divide our discussion into 4 sections: How Affordances Affect Layouts, Efficiency in Linear vs. Nonlinear Notebooks, Usability Benefits of Nonlinear Notebooks, and Design Challenges and Opportunities.

6.5.1 How Affordances Affect Layouts

One interesting finding from our results is how a higher percentage of users (37%) made a Grouped Combinations layout than those in Harden et al.'s exploratory work [22]. It must be noted that the set of software affordances available with the working nonlinear notebook of SAGECells in SAGE3 is not the same as the affordances suggested by the use of Miro boards as a scratch space to explore nonlinear arrangements; for example, SAGECells, at the time of this study, do not feature directed graph tools for run order. Furthermore, the screen space provided was not able to show all 10 columns at once without zooming out so much that the results would be too small to read. Given the available features, especially the ability to run a single column or row of cells in order, and the available screen space, it makes sense that more participants opted for a Grouped Combinations layout than a Multi-Column or Linear layout, separately. Since the Grouped Combinations layouts were able to fit everything on the screen in a way that makes sense to users and makes the most of the available affordances, it is no surprise that it was the most common layout.

Fork-It [75] introduces an interesting affordance: forking, or the temporary creation of a multi-column portion, or fork, within a linear notebook. In their study, Weinman et al. found forking useful for exploration and, unexpectedly, containing messes. However, the ephemeral nature of forks in their system limits its applicability for revisiting and/or reusing explored alternatives, as well as for presentation of narratives other than the main line. Given how our participants, on average, thought they would prefer to present with 2D notebooks, and slightly prefer to explore data and perform analysis with 2D notebooks, it makes sense to preserve explored alternatives for potential later use, a capability that SAGECells enable. That being said, there may be some value in being able to permanently discard some explored alternatives, such as to avoid spatialized messiness.

6.5.2 Efficiency in Linear vs. Nonlinear Notebooks

In their work on 2D Jupyter, Harden et al. [23] found efficiency gains on common comparative analysis tasks with rigidly structured 2D notebooks. However, their analysis did not include setup time, as the notebooks were pre-made, limiting the efficiency gains to the comparative analyses alone. This work did include setup time in the Initial Pass, as the notebooks only had data preparation code and one branch of the analysis included in their starting form, as seen in Fig. 6.2. While we did not see a statistically significant difference in efficiency between the 1D and 2D notebooks on the Initial Pass and Both Passes combined for the task we laid out, we did see such a difference on the Revisit Pass. Given how comparative analyses and revisiting and reusing analyses, one of which has been demonstrated to benefit from 2D notebooks [23] and the latter which we demonstrate in this paper, are so critical in data science work, the authors contend that the nature of data science work leads to solutions that can effectively utilize 2D space. While the Initial Pass did not have a statistically significant difference, this could be due to the lack of sufficient practice with SAGECells, as well as the potential for more upfront cognitive load due to having to decide how to organize cells. The freeform nature of SAGECells, in enabling user-imposed structure as opposed to software-imposed structure, does require users to make organizational decisions; while this setup time comes at a cost, our participants agreed that the cost seemed worth it, per Fig. 6.6, for the benefits provided.

6.5.3 Usability Benefits of Nonlinear Notebooks

The freeform, nonlinear 2D computational notebook capabilities of SAGECells within SAGE3 provide benefits over rigid 1D notebooks in several ways. Based on Fig. 6.5, locating items, comparing results, presenting a notebook, and dealing with Branching Code Paths in par-

ticular are seen as better done in a nonlinear, 2D notebook. Per the qualitative results in Table 6.7 and past literature [1, 2], the aforementioned benefits may be due to the way that nonlinear notebooks, when combined with a large display setup, enable Space to Think [2] to more effectively make use of available screen space. Four participants expressed the value of externalizing memory on the screen, and three noted the ability to organize information spatially, which are the key components of Space to Think [2]. Add in the ability to use physical navigation to (mostly) replace virtual navigation, which according to the quoted participant makes navigation "much easier", and performing tasks can become easier and more efficient [1].

6.5.4 Design Challenges and Opportunities

While freeform nonlinear 2D notebooks do have benefits, they also include some challenges, especially as it relates to cognitive load and added complexity in navigation.

The nonlinear 2D notebook environment SAGECells, by its organizational nature of being freeform, does not impose a particular structure on users; instead, users can create their own structures that may match their screen space and mental models better. However, this additional task of organizing code cells appears to come at a cognitive cost. Given that approximately 25% of our participants felt more training and practice with SAGECells would improve their performance, and it becomes clear that there is a cognitive hurdle here that thoughtful solutions could help address. Some potential solutions to address this hurdle include the creation and use of organizational templates, having an AI agent suggest, upon request, organizational patterns, and giving users more tools (e.g. creating a column- or row-based cell group) with which to impose their own structure upon the 2D canvas. Whether such tools would be effective enough and how effective they would be is an open area for

future research.

On another note, while three users expressed that navigation was easier in SAGE3 with SAGECells, five users expressed sentiments suggesting that such freeform 2D notebooks require more complex navigation than 1D notebooks. Although such issues might be addressable through more practice and experience with a freeform 2D computational notebook, it is important for designers to consider how to minimize the cost of increased complexity in virtual navigation in addition to maximizing the usefulness of physical navigation. After all, using a large physical display for work is not always possible, such as when one is traveling, and so freeform 2D computational notebook designers should strive to make virtual navigation easy and intuitive while also empowering physical navigation.

6.5.5 Limitations

Our work has some limitations, notably the potential for demand bias to affect results and the imperfect emulation of more complex forms of Branching Code Paths used in this study.

Demand Bias

Demand bias occurs when participants change their views and responses because they know or think they know the research agenda. While it is possible that some of our results may be affected by demand bias, some of the qualitative feedback we received suggests that this potential effect might not be a deal-breaker, so to speak. For example, some of the expressed benefits of freeform 2D notebooks related to prior research on large displays for sensemaking (Space to Think [2] and physical navigation [1] especially) without participants likely being aware of this prior research; the authors contend that this suggests genuine benefits have been uncovered.

Imperfect Emulation of Complex Branching Code Paths

While we did create a task that fits the definition of Branching Code Paths, it is true that the task we created was a very simple example. More complex examples, such as comparing multiple different machine learning algorithms on a single dataset, may not benefit as much from certain features of SAGECells, such as the capability to duplicate a code cell with the code preserved in the new cell. Still, the authors contend that the spatial organization patterns users created could be adapted to such more complex examples of Branching Code Paths while still enabling the benefits of nonlinear 2D notebooks to make better use of screen space.

6.5.6 Future Work

This study compared rigid 1D computational notebooks with freeform 2D computational notebooks to explore how users would organize a freeform 2D computational notebook with some helpful features and evaluate how such an environment would affect performance and subjective measures of usability. While this study did help answer those questions, there remains work to be done, especially as it relates to testing nonlinear 2D notebooks "in the wild" and comparing this freeform 2D computational notebook approach to a more rigid 2D computational notebook like in 2D Jupyter [23]. "In the wild" testing could help elucidate more clearly the benefits and challenges of using nonlinear 2D notebooks and better inform future designs. Comparing a freeform 2D computational notebook to a more rigid one could help elucidate where rigid, software-imposed structure and freeform, user-imposed structure best help users. Both of these directions may help inform the next generation of nonlinear notebooks. Finally, it may also be useful to compare nonlinear 2D computational notebooks with nonlinear 3D notebooks, such as those developed by In et al. [27], to determine the

benefits and challenges of each environment.

6.6 Conclusion

Computational notebooks are a versatile, popular tool for creating and presenting computational narratives. Current notebooks' linear, 1D, top-down organization, while elegantly simple, does appear to cause and/or exacerbate certain issues users have with them, especially when it comes to making the most of larger display setups and to Branching Code Paths, a common problem in data science. Given how new the idea of nonlinear 2D computational notebooks are and how rigid earlier designs have been, we sought to explore the potential of freeform nonlinear 2D computational notebooks to address issues affecting computational notebooks.

The freeform nonlinear 2D computational notebook environment of SAGECells in SAGE3 [68], when used with a nonlinear strategy, provides some benefits in efficiency as it relates to revisiting and reusing analyses, as well as comparative analyses, by enabling physical navigation and Space to Think [2]. Most users utilized the 2D space to create nonlinear layouts with tools like duplication of code cells, especially in the Multi-Column and Grouped Combinations patterns found by Harden et al. [22]. While a freeform, user-imposed approach to generating structures, like those mentioned above, does require additional work by users, its benefits appear to be worth the investment according to our study participants. Overall, freeform 2D computational notebooks have the potential to improve upon the current, rigid state of computational notebook organization through enabling innovative spatial patterns and Space to Think.

Table 6.7: Qualitative Themes in Study Surveys

Theme	*Sample Quote*	*Number of Participants*
2D Benefits		**17**
Easier Comparison vs. 1D	"[2D] is very useful for data analysis where we need to compare between multiple things."	11
External Memory	"The ability to look at multiple data simultaneously is the best usefulness of the 2D computational notebook."	4
Spatial Encoding Meaning	"I think 2D computational notebooks are very useful for data science problems because I feel like the way you can organize information is better than the way you would with a 1D notebook."	3
Physical Navigation Good	"The rapidness of moving your head side to side to compare data or moving your eyes up and down is much easier [compared to] having to scroll down a page."	3
Easier Navigation	"A 2D Notebook was much easier to navigate and perform data analysis as compared to 1D notebook."	3
Good Use of Display	"With the 2D notebook, I could arrange the cells in different groups and fully utilize the screen space."	3
2D Challenges		**8**
Training Needed	"2D is helpful, however the time required to understand what to do could enhance the barrier to entry."	5
More Complex Navigation	"I think if I was more comfortable with the controls, I would be able to navigate around more easily."	5
Setup Time Needed	"There is more time to set up in 2D versus opening up a 1D notebook and start typing."	2
1D Challenges		**10**
Virtual Navigation Needed	"For 1D notebook, I had to repeatedly scroll up and down..."	6
External Notes Needed	"I had to note down all the values [in 1D], lot of manual effort required for this."	4
Layout Preferences		**5**
Prefer 2D	"SAGE is a really amazing platform to perform data analysis... I think this really has a good potential to improve developer productivity and CS education."	4
Prefer 1D	"I preferred 1D, I think it's cleaner and easier to navigate and I also think it has the same flexibility as 2D. I could take notes on paper, so this would make up for the shortcomings when using 1D."	1

Chapter 7

Conclusion

7.1 Conclusions

This dissertation focused on four main research questions aimed at exploring the potential of spatialized 2D computational notebooks to address issues current linear computational notebooks have and improve upon the current state of computational notebooks. This chapter examines each research question and presents the findings from the studies conducted to address these questions.

7.1.1 RQ 1: What problems with current computational notebooks are due to their linear structure, which could feasibly be addressed with spatialized nonlinear computational notebook designs?

Previous works [11, 62] on understanding user issues with computational notebooks have focused broadly on a variety of pain points, but none have investigated the tension between the linear, 1D organization of cells and the nonlinearity of much data science work. To address research question 1, we first developed a set of patterns of nonlinearity from prior literature, our experiences, affinity diagramming, and consulting with data scientists. We then devel-

oped and deployed a survey to validate and measure approximately how problematic each pattern is in the eyes of users. This survey asked computational notebook users to rate how problematic each pattern is and to give qualitative feedback on each pattern; this survey also asked users to rate how helpful spatializing computational notebooks in 2D space (with an example image) might be to help address the patterns we developed. We found that most of our patterns were considered to be moderately problematic, with **Managing Nonlinear Execution Patterns**, **Controlling Cell Versions**, **Branching Code Paths**, and **Gathering Visualizations & Results into Dashboards** being the most critical patterns to address. We also found that most survey participants appeared to be cautiously optimistic about the potential of spatialized nonlinear notebooks' potential to address such issues.

7.1.2 RQ 2: How would users organize computational notebook cells in spatialized 2D notebooks?

Given that patterns of nonlinearity exist in data science work with computational notebooks and that spatialized 2D computational notebooks may be able to address these patterns, it is important to test how users would actually prefer to organize computational notebook cells in 2D space. Specifically, we sought to understand how users might arrange a collection of cells from a completed notebook given a 2D spatial environment, and whether users would see such an environment as helpful to various tasks within data science. We performed an exploratory user study in which participants took images of code and markdown cells, placed in a pile on a Miro online whiteboard [51], and arranged the cells in the 2D space a way that made sense to their understanding of the story the notebook told; after completing the task, participants filled out a survey to help us better understand their organization of cells and gauge their thoughts on the potential of spatialized 2D notebooks. Our analysis showed that the vast majority of participants (84%) did make use of 2D space in some way; the most

consistent organizational pattern users developed was a single row of columns, or a **Multi-Column** style. Other nonlinear organizational patterns included nesting rows and columns, or **Grouped Combinations**, using arrows between cells to create a **Directed Graph**, as well as more linear arrangements such as splitting the single-column structure into multiple columns at a given point, or **Split-Column**, and splitting 1 cell in a single column into a row of cells, or **Split-Cell**. Additionally, our survey analysis showed that most participants saw a spatialized 2D notebook as having potential to improve over 1D notebooks, especially for comparison activities, although there was some skepticism regarding navigation and debugging in such an environment. Our findings from this study guided our work on RQ 3, where we tested the **Multi-Column** style in particular.

7.1.3 RQ 3: What potential benefits could spatialized nonlinear computational notebooks which implement a multi-column structure provide for data science over current linear notebooks?

To address RQ 3, we ran a user study with a prototype spatialized 2D computational notebook extension for Jupyter Notebooks that implemented the **Multi-Column** organizational pattern. This user study focused on statistically comparing efficiency and usability of this extension with linear, 1D Jupyter Notebooks. We found that the spatialized 2D Jupyter Notebooks with the multi-column style increased efficiency on various comparative analysis tasks and that users found the system more usable than 1D, linear Jupyter Notebooks. By minimizing the need to scroll or use other virtual navigation tools through enabling physical navigation, our spatialized 2D Jupyter extension helped make it easier and more efficient to perform the comparative analysis tasks. One finding of note from this study was that

those users who were exposed to the 2D condition with 2D Jupyter before the 1D condition found 1D to be less usable than those who started with 1D; this may mean that exposure to alternative, spatialized methods of organizing computational notebook cells may cause some users to see the flaws of 1D, linear computational notebooks as being due in part to their rigid, 1D organization. It should be noted that this study used already-made complete notebooks and thus did not require creation of any new code cells; this factor was explored as part of our work on RQ 4.

7.1.4 RQ 4: What potential benefits could spatialized freeform nonlinear computational notebooks provide for data science over current linear notebooks?

To address RQ 4, we ran a user study with a freeform, spatialized 2D computational notebook environment (SAGECells) within the online whiteboard/canvas software known as SAGE3. This user study focused on statistically comparing efficiency and usability of this environment with linear, 1D Jupyter Notebooks for the Branching Code Paths pattern of nonlinearity, as well as how users would organize the cells in a working freeform 2D computational notebook. It should be noted that participants started with a notebook with pre-processing and 1 branch of the analysis done, from which they had to choose how to tackle the rest of the problem. We found similar overall performance in terms of efficiency and accuracy in both the linear, 1D condition and the freeform, 2D condition; after removing those users who did not utilize the 2D environment to create nonlinear layouts from consideration, we found that efficiency was better in the 2D condition when users did not have to write any more code and just had to compare results again with the dataset slightly changed. This suggests that spatialized 2D computational notebooks, if not used in a nonlinear manner, may not

provide benefits over that of linear, 1D notebooks; if used with nonlinear layouts, however, 2D notebooks are indeed better for comparative analyses, echoing the results found for RQ 3. We also found that the layouts participants created were similar to the results found for RQ 2, except that there were no directed graph layouts due to SAGECells lacking this feature. In addition, users who opted for a Grouped Combinations layout appeared to take into account the available screen space when designing their layout, likely due to wanting to have all results on the screen at once.

7.2 Future Work Directions

7.2.1 Scalability

One unexplored area of research with respect to spatialized 2D computational notebooks is that of scalability. The user studies in this dissertation mainly focused on tasks with a medium number of cells, generally less than 40. While 1D, linear computational notebooks generally fit less than 10 cells on the screen at once and spatialized 2D computational notebooks can fit more depending on screen size, the challenge of how to deal with the situation where the screen cannot show all the cells at once without zooming far out is one worth considering and addressing. Real-world work may involve far more than 40 cells over the course of a project, with potentially 100 or more cells, and that is without mentioning different versions of a cell. How can spatialized 2D computational notebooks be used when there are more cells than can fit on even a larger screen setup? What strategies might users adopt in such an environment to deal with this issue?

One such approach may be found in semantic encoding of meaning into space. In other words, users may decide that different sections of a spatialized 2D notebook may hold cells related

to different things, such as tasks. For example, a user might designate a section for pre-processing, another for explorations in progress, and yet another for completed explorations. Another idea is to implement virtual navigation options like switching tabs between different notebooks for different parts. However, the iterative, interconnected nature of data science tasks may pose some challenges to this method, much like the drawbacks of virtual navigation found in the user studies. A third idea might be to incorporate some sort of minimization feature that enables users to minimize, or hide, certain groups of cells until they are needed again. It would be interesting to see how users deal with the challenge of organizing a notebook with many more cells than can fit on a larger display setup. Finally, scaling to 100+ cells may also require some innovation in terms of how a spatialized 2D computational notebook is coded in order to maintain fluid navigation and execution of cells.

There are numerous questions to address regarding how users deal with larger projects with a spatialized 2D notebook environment and how such an environment can be improved for such cases. Exploring these questions could help improve spatialized 2D computational notebooks and computational notebooks in general.

7.2.2 Creating and Managing Run Order

One issue that comes from breaking out of the linear, top-down, 1D mold of current computational notebooks into spatialized 2D notebooks is that of creating and managing run order. In the linear, 1D environment, it is trivial to define and implement a run order for the entire notebook, as well as to provide features like ”run all starting at this cell” or ”run all before this cell” to users. However, with the introduction of 2D space for organizing computational notebook cells, the problem of creating and managing run order is no longer trivial; while users may organize cells into (potentially nested) columns and rows, there is no longer just

one potential interpretation of run order. This fact leads to issues when trying to develop algorithms to run multiple cells in order.

One method of dealing with run order is to have algorithms make decisions based on certain characteristics of the organization of cells, such as whether the cells (either all or a selected subset) are wider than they are tall, or vice versa; this method was used in the SAGECells paper to provide users a way to run more than 1 cell in a particular order. However, such a method may have a hard time with more complex arrangements of cells than a single row or column; a grid, for example, can be organized as either a row of columns or a column of rows, leading to two possible interpretations on how it should be run. While a grid could potentially be addressed by asking a user through a pop-up to choose a run-order option, such a method may mean asking the same question of the same set of cells each time the grid is re-run. Furthermore, if a user selects the wrong option, they may have to either wait until the grid has been fully run wrongly or otherwise find a way to interrupt and stop the algorithm from continuing to run cells. Thus, there may be more salient and intuitive ways to ensure that such human-in-the-loop measures do not get tiring or unnecessarily repetitive for users.

Visualization methods could be a useful tool for helping users see and change run order without having to repeat steps unnecessarily. One potential solution for run order issues is that of drawing directed lines (with arrows) between cells to show what the system thinks the overall run order is, and allowing human users to edit those links to correct misconceptions of the algorithm. Combined with an AI backend that is perhaps trained on a dataset of organizations and run orders or otherwise trained to reason about run order based on various features of the notebook (including spatial organization), a human user could potentially quicken the process of updating the visualized run order to be more in line with the user's mental model. One potential area of investigation here would be predicting and seeing how

run order in a spatialized 2D notebook environment may evolve over the course of a data science project, and how AI assistants might be able to help quicken run order adjustments. Given how nonlinear the iterative and exploratory work of data science is, I would conjecture that run order in spatialized 2D notebooks may need adjusting fairly often.

7.2.3 User Studies with Experts in Data Science

All of the user studies in this dissertation use a population of convenience, namely college students. However, expert users in the field of data science may tackle projects with spatialized 2D notebooks differently than students who are in the process of learning data science. Would there be a difference in results on efficiency and usability with expert users compared to college students, and, if so, how would they differ? How would their strategies for tackling problems differ, and what effect might this have on how they use the space?

One approach to answering this question could be to solicit expert users through networking, academic and other listservs, Kaggle boards for competitions, and more for studying the use of spatialized 2D notebooks "in the wild" with data science experts. Such a study would likely be longitudinal in nature, with more qualitative analysis than quantitative, and may have a small sample size due to potential difficulties in getting participants. Given that such experts would be geographically distributed over a wide area, may or may not have larger display setups, may have varying years of experience, and may have varying levels of expertise in computer programming and other skills, it may be wise to use surveys and interviews to assess these various items and their impact on the use of spatialized 2D notebooks.

Overall, there are numerous interesting questions that could be answering by shifting the population of focus from one of convenience to data science experts. Exploring such questions could help significantly enhance the usability and effectiveness of spatialized 2D computa-

tional notebooks.

7.2.4 Collaborative Scenarios

The two user studies in this dissertation with working spatialized 2D computational notebooks focused on single-user scenarios; future work could address collaborative work on data science and analysis using a spatialized 2D notebook.

One area to explore would be how collaborative users organize their work, and what affects such decisions. Do collaborative users create separate areas for their own explorations and ideas, and, if so, how do they merge their work together? If collaborative users utilize a shared space, how do they negotiate aspects like who gets to run or move a code cell? How does the collaboration mode across time (synchronous, asynchronous) and space (co-located, distributed) affect such collaborations? Does the role a user has in a team (e.g. boss, junior data scientist, senior data scientist) affect organizational strategies and, if so, how? Investigating these questions could provide valuable insights into benefits, challenges, and design opportunities for spatialized 2D notebooks.

7.2.5 Utilizing More Objective Measures

The studies in this dissertation relied a lot on subjective Likert-Scale questions to assess metrics such as usability and user preferences. Including more objective or time-tested measures would not only help elucidate potential benefits and challenges facing spatialized 2D computational notebooks, but also help counter the possibility of biases such as demand bias (participants telling researchers what they think the researchers want to hear) and novelty bias (participants appraising a novel tool more highly upon first using it than they would otherwise) in research evaluating spatialized 2D computational notebooks. Although

strong qualitative feedback can help suggest that these biases may not be the main reason for effects seen, it can be argued that they should generally not be considered enough on their own.

One particular area that would be useful to evaluate with a more objective measure is that of cognitive load. While I have speculated that adding the task of organizing cells in a 2D computational notebook may add significant cognitive load up front (that may save time down the road by minimizing or even avoiding issues such as messiness), it would be better to find measurements of this that can provide more objective evidence upon which my speculations (and other similar ideas) can be judged. One way of measuring cognitive load is with questionnaires like the NASA-TLX [24]; however, such questionnaires may be more subjective than a researcher would prefer, and thus one may wish to combine it with other metrics, such as physiological measurements, post-task interviews, and perhaps even a think-aloud protocol with audio recordings to note events like where users get stuck. To this end, I would suggest that anyone looking to study such questions have a strong grasp of psychological measurements or collaborate with such an individual.

7.2.6 Delving Deeper with Qualitative Analysis

The studies in this dissertation, for the qualitative analysis, focused mainly on open coding, and did not tend to proceed to axial or selective coding; this was done mainly due to time constraints. That being said, a deeper qualitative analysis backed by rigorous methodology, such as grounded theory as described by Charmaz [10], may help elucidate further insights into the opportunities and challenges that spatialized 2D computational notebooks present. In fact, I would recommend that such methodologies be thoughtfully considered in any qualitative attempts to address questions brought up by this dissertation, including this

future work section.

7.2.7 Exploring Whether and How Users Think About Organizing Cells in 2D Notebooks

One set of questions that were not directly explored in this dissertation relates to whether and how users think about organizing cells in spatialized 2D computational notebooks. This set of questions includes whether such organizational tasks add additional cognitive load to doing data science in computational notebooks, if they simply externalize processes that users already tended to perform internally, or if something else happens to be the case; it also includes how users think about organization and what goes into their decisions and executions of organizational structures.

Some of these questions could be explored by using a Think-Aloud protocol and recording the audio during studies. For such studies, it would be best to get users who have their own datasets they want to analyze so that they are both motivated to do deep, thoughtful analyses and unconstrained by conditions usually found in a controlled study. It may be useful to emulate some of the study methods found in a work by Batch et al. [3], such as having a space users can reserve to use a spatialized 2D computaitonal notebook with a larger screen setup that is equipped to record audio (and potentially visuals as well) during analysis; this could help with recruiting enough participants to gather enough data, although enforcing a Think-Aloud protocol would be difficult with such a method.

7.2.8 Effects of Available Screen Space and its Usage

The two user studies in this dissertation with working spatialized 2D computational notebooks provided display that were of moderate and large sizes, effectively providing the potential of Space to Think, and did not delve deeply into testing different display sizes and configurations within either study; future work could address the effects of screen space and its usage to further elucidate benefits and challenges of spatialized 2D computational notebooks.

One possible research direction would be investigating the effects of screen size on efficiency, accuracy, and subjective measures of usability when using a spatialized 2D computational notebook. Such a study would benefit from having multiple different display setups, not just small laptop screen and large TV screen; at the very least, there should be a small laptop screen condition (e.g. 15”), a moderate desktop monitor condition (e.g. 24”), and a large, potentially widescreen, TV condition (e.g. the screen used in the SAGECells study). Given that there are at least three conditions for one independent variable, it may be wise to use a between-subjects design to better ensure an adequate amount of participants. While a within-subjects design could work for 3 conditions, it would require 6 subgroups for each of the potential orderings of the conditions and would likely need at least 5 participants in each subgroup (if not more); this means at least 30 participants doing all 3 conditions, which, in my experience, is impractical to achieve in a timely manner. With a between-subjects design, 30 participants means 10 to each condition, and with each participant only performing 1 condition, this should be much more easily doable. Regardless, for this kind of user study the task should be something that users have experience doing in computational notebooks without being an exact copy. Thus, recruiting students who have taken or are taking a class like CS-CMDA-STAT 3654 may be advisable. Finally, in addition to measures of efficiency, accuracy, and subjective perceptions of usability, it would be worthwhile to ask

each participant to note any design challenges or benefits that they perceived during the task.

Another potential direction is to compare a linear, 1D computational notebook with a freeform spatialized 2D computational notebook on a large, vertical screen. Given how such a screen is useful for coding tasks, especially with linear computational notebooks since it can show more cells, it would be interesting to see how a freeform spatialized 2D computational notebook performs against it; it would also be interesting to see how users organize their cells in a freeform 2D notebook given a larger amount of vertical space. Such a study could be similar in design to the SAGECells study, with a within-subjects design and counterbalancing of the order in which participants complete conditions.

Finally, it would also be useful to explore how much of a large screen is used and how apps on the screen are organized with both linear, 1D computational notebooks and spatialized, 2D computational notebooks. This could be done with screen recordings, which could be used to estimate the percentage of pixels used at different points in a study. Such explorations could also help elucidate what causes certain benefits; is it more about having more items on the screen, the way items are laid out, the size of certain items, etc.?

7.2.9 Human-Machine Teaming

Given recent advances in artificial intelligence (AI) agents and methods, such as large language models (LLMs) for coding and more, it makes sense to explore how such AI tools could be used to benefit spatialized 2D computational notebooks and their users.

Integrating AI agents could help with various problems, from suggesting statistical and machine learning methods to try with a dataset to helping users organize their notebooks according to common templates and/or tasks. Such integration also brings with it some risks,

such as the AI suggesting methods that don't fit the dataset's characteristics, suggesting organizational structures that don't fit the analysis, and more.

Some questions that could be explored include how users would want to interact with AI agents, whether and how users can correct faulty AI suggestions, what tasks users feel comfortable trusting AI with, and more. Exploring such questions could help lead to critical advances in the design and implementation of spatialized 2D computational notebooks with AI for analytics.

Bibliography

[1] Christopher Andrews and Chris North. The impact of physical navigation on spatial organization for sensemaking. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2207–2216, 2013.

[2] Christopher Andrews, Alex Endert, and Chris North. Space to think: large high-resolution displays for sensemaking. In Elizabeth D. Mynatt, Don Schoner, Geraldine Fitzpatrick, Scott E. Hudson, W. Keith Edwards, and Tom Rodden, editors, *Proceedings of the 28th International Conference on Human Factors in Computing Systems, CHI 2010, April 10-15, 2010*, pages 55–64, Atlanta, Georgia, USA, 2010. ACM. doi: 10.1145/1753326.1753336. URL https://doi.org/10.1145/1753326.1753336.

[3] Andrea Batch, Andrew Cunningham, Maxime Cordeil, Niklas Elmqvist, Tim Dwyer, Bruce H Thomas, and Kim Marriott. There is no spoon: Evaluating performance, space use, and presence with expert domain users in immersive analytics. *IEEE transactions on visualization and computer graphics*, 26(1):536–546, 2019.

[4] Marijan Beg, Juliette Taka, Thomas Kluyver, Alexander Konovalov, Min Ragan-Kelley, Nicolas M Thiéry, and Hans Fangohr. Using jupyter for reproducible scientific workflows. *Computing in Science & Engineering*, 23(2):36–46, 2021.

[5] Lonni Besançon and Pierre Dragicevic. The continued prevalence of dichotomous inferences at chi. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI EA '19, page 1–11, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359719. doi: 10.1145/3290607.3310432. URL https://doi.org/10.1145/3290607.3310432.

[6] Mike Bostock. Introducing observable canvases | observable, 2025. URL https://observablehq.com/blog/introducing-canvases-early-access.

[7] Lauren Bradel, Alex Endert, Kristen Koch, Christopher Andrews, and Chris North. Large high resolution displays for co-located collaborative sensemaking: Display usage and territoriality. *International Journal of Human-Computer Studies*, 71(11):1078–1088, 2013.

[8] Andrew Bragdon, Steven P Reiss, Robert Zeleznik, Suman Karumuri, William Cheung, Joshua Kaplan, Christopher Coleman, Ferdi Adeputra, and Joseph J LaViola Jr. Code bubbles: rethinking the user interface paradigm of integrated development environments. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pages 455–464, 2010.

[9] Andrew Burks, Luc Renambot, and Andrew Johnson. Vissnippets: A web-based system for impromptu collaborative data exploration on large displays. In *Practice and Experience in Advanced Research Computing*, pages 144–151. 2020.

[10] Kathy Charmaz. Grounded theory. *Qualitative psychology: A practical guide to research methods*, 3:53–84, 2015.

[11] Souti Chattopadhyay, Ishita Prasad, Austin Z Henley, Anita Sarma, and Titus Barik. What's wrong with computational notebooks? pain points, needs, and design opportunities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–12, Honolulu, HI, USA, 2020. ACM.

[12] Andy Cockburn, Pierre Dragicevic, Lonni Besançon, and Carl Gutwin. Threats of a replication crisis in empirical computer science. *Commun. ACM*, 63(8):70–79, jul 2020. ISSN 0001-0782. doi: 10.1145/3360311. URL https://doi.org/10.1145/3360311.

[13] Kylie Davidson, Lee Lisle, Kirsten Whitley, Doug A Bowman, and Chris North. Exploring the evolution of sensemaking strategies in immersive space to think. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[14] Joost CF De Winter and Dimitra Dodou. Five-point likert items: t test versus mann-whitney-wilcoxon. *Practical Assessment, Research & Evaluation*, 15(11):1–12, 2010.

[15] Helen Dong, Shurui Zhou, Jin L.C. Guo, and Christian Kästner. Splitting, renaming, removing: A study of common cleaning activities in jupyter notebooks. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, pages 114–119, 2021. doi: 10.1109/ASEW52652.2021.00032.

[16] Pierre Dragicevic, Yvonne Jansen, Abhraneel Sarma, Matthew Kay, and Fanny Chevalier. Increasing the transparency of research papers with explorable multiverse analyses. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–15, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359702. doi: 10.1145/3290605.3300295. URL https://doi.org/10.1145/3290605.3300295.

[17] Einblick. Einblick. *https://www.einblick.ai*, 2023. URL https://www.einblick.ai.

[18] Alex Endert, Patrick Fiaux, and Chris North. Semantic interaction for sensemaking: inferring analytical reasoning for model steering. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2879–2888, 2012.

[19] Ali Ghodsi, Ted Tomlinson, Patrick Wendell, Prem Prakash, and Cassie Miao. Databricks + einblick | databricks blog, 2025. URL https://www.databricks.com/blog/welcome-data-intelligence-platform-databricks-einblick.

[20] Google. Welcome to colaboratory - colaboratory, 2022. URL https://colab.research.google.com/.

[21] Jesse Harden. Exploring and evaluating the potential of 2d computational notebooks. In *Companion Proceedings of the 2023 Conference on Interactive Surfaces and Spaces*, pages 97–99, 2023.

[22] Jesse Harden, Elizabeth Christman, Nurit Kirshenbaum, John Wenskovitch, Jason Leigh, and Chris North. Exploring organization of computational notebook cells in 2d space. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–6, Rome, Italy, 2022. IEEE.

[23] Jesse Harden, Elizabeth Christman, Nurit Kirshenbaum, Mahdi Belcaid, Jason Leigh, and Chris North. "there is no reason anybody should be using 1d anymore": Design and evaluation of 2d jupyter notebooks. In *Graphics Interface 2023-second deadline*, 2023.

[24] Sandra G Hart. Nasa-task load index (nasa-tlx); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 50, pages 904–908. Sage publications Sage CA: Los Angeles, CA, 2006.

[25] Andrew Head, Fred Hohman, Titus Barik, Steven M Drucker, and Robert DeLine. Managing messes in computational notebooks. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.

[26] Ross Ihaka and Robert Gentleman. R: a language for data analysis and graphics. *Journal of computational and graphical statistics*, 5(3):299–314, 1996.

[27] Sungwon In, Eric Krokos, Kirsten Whitley, Chris North, and Yalong Yang. Evaluating navigation and comparison performance of computational notebooks on desktop and in

virtual reality. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2024.

[28] Einblick Analytics Inc. Einblick | multiplayer python notebooks on an interactive canvas, 2023. URL https://www.einblick.ai/.

[29] Sagemath Inc. Collaborative calculation and data science, 2023. URL https://cocalc.com/.

[30] Project Jupyter. Project jupyter | home, 2021. URL https://jupyter.org/.

[31] Sean Kandel, Andreas Paepcke, Joseph M Hellerstein, and Jeffrey Heer. Enterprise data analysis and visualization: An interview study. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2917–2926, 2012.

[32] Mary Beth Kery and Brad A Myers. Interactions for untangling messy history in a computational notebook. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 147–155. IEEE, 2018.

[33] Mary Beth Kery, Amber Horvath, and Brad A Myers. Variolite: Supporting exploratory programming by data scientists. In *CHI*, volume 10, pages 3025453–3025626, 2017.

[34] Mary Beth Kery, Marissa Radensky, Mahima Arya, Bonnie E John, and Brad A 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*, pages 1–11, 2018.

[35] Mary Beth Kery, Donghao Ren, Fred Hohman, Dominik Moritz, Kanit Wongsuphasawat, and Kayur Patel. mage: Fluid moves between code and graphical work in computational notebooks. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 140–151, 2020.

[36] David Kirsh. Thinking with external representations. *AI & society*, 25:441–454, 2010.

[37] Nurit Kirshenbaum, Kylie Davidson, Jesse Harden, Chris North, Dylan Kobayashi, Ryan Theriot, Roderick S Tabalba Jr, Michael L Rogers, Mahdi Belcaid, Andrew T Burks, et al. Traces of time through space: Advantages of creating complex canvases in collaborative meetings. *Proceedings of the ACM on Human-Computer Interaction*, 5 (ISS):1–20, 2021.

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

[39] Donald Ervin Knuth. Literate programming. *The computer journal*, 27(2):97–111, 1984.

[40] Magdalena Konkiewicz. Jupyter notebook extensions (part 2), 2020. URL https://towardsdatascience.com/jupyter-notebook-extensions-part-2-55fdb2c38348.

[41] Robert Kosara and Jock Mackinlay. Storytelling: The next step for visualization. *Computer*, 46(5):44–50, 2013.

[42] William H Kruskal and W Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621, 1952.

[43] Lee Lisle, Xiaoyu Chen, JK Edward Gitre, Chris North, and Doug A Bowman. Evaluating the benefits of the immersive space to think. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 331–337. IEEE, 2020.

[44] Lee Lisle, Kylie Davidson, Edward JK Gitre, Chris North, and Doug A Bowman. Sense-

making strategies with immersive space to think. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, pages 529–537. IEEE, 2021.

[45] Eric S Liu, Dylan A Lukes, and William G Griswold. Refactoring in computational notebooks. *ACM Transactions on Software Engineering and Methodology*, 2022.

[46] Eric S Liu, Dylan A Lukes, and William G Griswold. Refactoring in computational notebooks. *ACM Transactions on Software Engineering and Methodology*, 32(3):1–24, 2023.

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

[48] Salvatore S. Mangiafico. *rcompanion: Functions to Support Extension Education Program Evaluation*. Rutgers Cooperative Extension, New Brunswick, New Jersey, 2023. URL https://CRAN.R-project.org/package=rcompanion/. version 2.4.30.

[49] Trevor Manz, Nezar Abdennur, and Nils Gehlenborg. anywidget: reusable widgets for interactive analysis and visualization in computational notebooks. 2024.

[50] Pavlo V Merzlykin, Maiia V Marienko, and Svitlana V Shokaliuk. Cocalc tools as a means of open science and its didactic potential in the educational process. In *Proceedings of the 1st Symposium on Advances in Educational Technology*, volume 1, pages 109–118, 2022.

[51] Miro. The visual collaboration platform for every team | miro, 2022. URL https://miro.com/index/.

[52] Derek H. Ogle, Jason C. Doll, A. Powell Wheeler, and Alexis Dinno. *FSA: Simple Fish-*

eries Stock Assessment Methods, 2023. URL https://CRAN.R-project.org/package=FSA. R package version 0.9.4.

[53] Leonardo Pavanatto, Chris North, Doug A Bowman, Carmen Badea, and Richard Stoakley. Do we still need physical monitors? an evaluation of the usability of ar virtual monitors for productivity work. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, pages 759–767. IEEE, 2021.

[54] Fernando Perez and Brian E Granger. Project jupyter: Computational narratives as the engine of collaborative data science. *Retrieved September*, 11(207):108, 2015.

[55] Jeffrey M Perkel. Why jupyter is data scientists' computational notebook of choice. *Nature*, 563(7732):145–147, 2018.

[56] João Felipe Pimentel, Leonardo Murta, Vanessa Braganholo, and Juliana Freire. A large-scale study about quality and reproducibility of jupyter notebooks. In *2019 IEEE/ACM 16th international conference on mining software repositories (MSR)*, pages 507–517. IEEE, 2019.

[57] Peter Pirolli and Stuart Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of international conference on intelligence analysis*, volume 5, pages 2–4. McLean, VA, USA, 2005.

[58] Luigi Quaranta, Fabio Calefato, and Filippo Lanubile. Eliciting best practices for collaboration with computational notebooks. *Proceedings of the ACM on Human-Computer Interaction*, 6(CSCW1):1–41, 2022.

[59] Deepthi Raghunandan, Aayushi Roy, Shenzhi Shi, Niklas Elmqvist, and Leilani Battle. Code code evolution: Understanding how people change data science notebooks over time. *arXiv preprint arXiv:2209.02851*, 2022.

[60] Patrick Reipschlager, Tamara Flemisch, and Raimund Dachselt. Personal augmented reality for information visualization on large interactive displays. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1182–1192, 2020.

[61] Hugo Romat, Nathalie Henry Riche, Ken Hinckley, Bongshin Lee, Caroline Appert, Emmanuel Pietriga, and Christopher Collins. Activeink: (th) inking with data. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2019.

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

[63] Adam Rule, Amanda Birmingham, Cristal Zuniga, Ilkay Altintas, Shih-Cheng Huang, Rob Knight, Niema Moshiri, Mai H Nguyen, Sara Brin Rosenthal, Fernando Pérez, et al. Ten simple rules for writing and sharing computational analyses in jupyter notebooks, 2019.

[64] C James Scheirer, William S Ray, and Nathan Hare. The analysis of ranked data derived from completely randomized factorial designs. *Biometrics*, pages 429–434, 1976.

[65] Zeyuan Shang, Emanuel Zgraggen, Benedetto Buratti, Philipp Eichmann, Navid Karimeddiny, Charlie Meyer, Wesley Runnels, and Tim Kraska. Davos: a system for interactive data-driven decision making. *Proceedings of the VLDB Endowment*, 14(12): 2893–2905, 2021.

[66] Jeremy Singer. Notes on notebooks: Is jupyter the bringer of jollity? In *Proceedings of the 2020 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, pages 180–186, 2020.

[67] Ryo Suzuki, Takuto Takahashi, Kenta Masuda, and Ikuro Choh. Implementing node-link interface into a block-based visual programming language. In *International Conference on Human-Computer Interaction*, pages 455–465. Springer, 2018.

[68] Roderick Tabalba, Nurit Kirshenbaum, Jesse Harden, Michael Rogers, Arthur Nishimoto, Elizabeth Christman, Andy Yu, Ryan Theriot, Lance Long, Luc Renambot, Mahdi Belcaid, Chris North, Andrew Johnson, and Jason Leigh. Sage3 - the smart amplified group environment. In *Science Gateways 2023 (SG23)*. Zenodo, October 2023. doi: 10.5281/zenodo.10034793. URL https://doi.org/10.5281/zenodo.10034793.

[69] SAGE3 Team. Sage3 - collaborate smarter, 2024. URL https://sage3.sagecommons.org/.

[70] Kateryna Vlasenko, Olena Chumak, Dmytro Bobyliev, Iryna Lovianova, and Iryna Sitak. Development of an online-course syllabus" operations research oriented to cloud computing in the cocalc system". In *ICTERI*, pages 278–291, 2020.

[71] April Yi Wang, Anant Mittal, Christopher Brooks, and Steve Oney. How data scientists use computational notebooks for real-time collaboration. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–30, 2019.

[72] Jiawei Wang, Tzu-yang Kuo, Li Li, and Andreas Zeller. Assessing and restoring reproducibility of jupyter notebooks. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pages 138–149, 2020.

[73] Zijie J Wang, Katie Dai, and W Keith Edwards. Stickyland: Breaking the linear presentation of computational notebooks. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pages 1–7, 2022.

[74] Zijie J Wang, David Munechika, Seongmin Lee, and Duen Horng Chau. Supernova:

Design strategies and opportunities for interactive visualization in computational notebooks. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–17, 2024.

[75] Nathaniel Weinman, Steven M Drucker, Titus Barik, and Robert DeLine. Fork it: Supporting stateful alternatives in computational notebooks. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2021.

[76] John Wenskovitch, Jian Zhao, Scott Carter, Matthew Cooper, and Chris North. Albireo: An interactive tool for visually summarizing computational notebook structure. In *2019 IEEE Visualization in Data Science (VDS)*, pages 1–10. IEEE, 2019.

[77] Dylan Wootton, Amy Rae Fox, Evan Peck, and Arvind Satyanarayan. Charting eda: Characterizing interactive visualization use in computational notebooks with a mixed-methods formalism. *IEEE Transactions on Visualization and Computer Graphics*, 2024.

[78] Yifan Wu, Joseph M Hellerstein, and Arvind Satyanarayan. B2: Bridging code and interactive visualization in computational notebooks. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 152–165, 2020.

[79] Charles Zaiontz. Free download | real statistics using excel, 2025. URL https://real-statistics.com/free-download/.