2D Jupyter: Design and Evaluation of 2D Computational Notebooks

Elizabeth Y. Christman

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

Master of Science
in
Computer Science & Applications

Christopher North, Chair
Muhammad Ali Gulzar
Mahdi Belcaid

May 2, 2023
Blacksburg, Virginia

Keywords: computational notebooks, software, data science

2D Jupyter: Design and Evaluation of 2D Computational Notebooks

Elizabeth Y. Christman

(ABSTRACT)

Computational notebooks are a popular tool for data analysis. However, the 1D linear structure used by many computational notebooks can lead to challenges and pain points in data analysis, including messiness, tedious navigation, inefficient use of screen space, and presentation of non-linear narratives. To address these problems, we designed a prototype Jupyter Notebooks extension called 2D Jupyter that enables a 2D organization of code cells in a multi-column layout, as well as freeform cell placement. We conducted a user study using this extension to evaluate the usability of 2D computational notebooks and understand the advantages and disadvantages that it provides over a 1D layout. As a result of this study, we found evidence that the 2D layout provides enhanced usability and efficiency in computational notebooks. Additionally, we gathered feedback on the design of the prototype that can be used to inform future work. Overall, 2D Jupyter was positively received and users not only enjoyed using the extension, but also expressed a desire to use 2D notebook environments in the future.

2D Jupyter: Design and Evaluation of 2D Computational Notebooks

Elizabeth Y. Christman

(GENERAL AUDIENCE ABSTRACT)

Computational notebooks are a tool commonly used by data analysts that allows them to construct computational narratives through a combination of code, text and visualizations. Many computational notebooks use a 1D linear layout; however data analysis work is often conducted in a non-linear fashion due to the need to debug code, test new theories, and evaluate and compare results. In this work, we present a prototype extension for Jupyter Notebooks called 2D Jupyter that enables the user to arrange their notebook in a 2D multi-column layout. A user study was conducted to evaluate the usability of this extension and understand the benefits that a 2D layout may provide. Feedback on the extension's design was also collected to inform future design opportunities. The prototype received a positive reaction overall and users expressed a desire to use 2D computational notebooks in their future work.

Dedication

This thesis is dedicated to my family.

Acknowledgments

I would first like to thank my advisor, Dr. Chris North for all of his guidance throughout my degree. Without him, this thesis work would not have been possible.

I would also like to thank my committee members, Dr. Muhammad Ali Gulzar and Dr. Mahdi Belcaid for their time and effort dedicated to my defense.

Thank you to Jesse Harden for his support and advice in completing this research.

Thank you to Timothy Wu, Rituraj Sharma, and Jonathan West for all their hard work on 2D JupyterLab.

Contents

List of Figures

List of Tables

List of Abbreviations

1D 1 dimensional

2D 2 dimensional

In this work, 1D refers to the traditional computational notebook layout of a single column of code cells.

2D refers to a 2 dimensional computational notebook layout. This work primarily discusses a multi-column layout; however this can also refer to other layouts where both horizontal and vertical navigation is possible.

Chapter 1

Introduction

Computational notebooks like Jupyter [15, 21] and Google Colab [4, 11] are popular tools for data analysis used by millions across diverse fields. The ability to combine code, text, and visualizations in a single document has allowed data analysts to quickly test models, refine results, and construct computational narratives [30]. While computational notebooks provide various features supporting the specific needs of data analysts, many users still find difficulties and pain points present in their use [8]. One limitation may be the 1D linear structure of computational notebooks, which presents challenges when dealing with non-linear data analysis and creating computational narratives.

A study conducted by Rule et. al identified support for non-linear structures as a design opportunity for computational notebooks, with almost half of the notebooks they examined containing a non-linear run order [30]. Additionally, non-linear structures may be helpful in navigating longer notebooks, and may help to prevent and manage messiness in a notebook [13, 18]. Harden et. al [12] proposed a 2D layout for computational notebooks and found that users found 2D space to be helpful in conducting analyses and organizing their notebooks. The concept of space to think [3] may also be beneficial; tools such as Einblick [2] and CoCalc [14] have emerged that make use of the concept of space to think to benefit data analysis.

Prior research has shown the benefits of space to think in a variety of environments, including immersive virtual reality spaces [23] and collaborative spaces [20]. In this work, we evaluate the potential impacts of space to think in computational notebooks, specifically through

the implementation of a 2D layout. Because Jupyter is the most popualar comptutational notebook software [25], we chose to target our efforts towards enabling 2D layouts in Jupyter Notebooks. We focused on the following research questions:

- **RQ1:** How can we extend Jupyter Notebooks to make use of 2D space?
- **RQ2:** What features do users want in a 2D layout of Jupyter Notebooks?
- **RQ3:** How do people use 2D space when conducting data analyses?
- **RQ4:** What advantages and disadvantages does a 2D layout have compared to a 1D layout?

To answer these questions, we designed a developed a prototype Jupyter Notebooks extension called 2D Jupyter that enables multi-column layouts of code cells. This extension also allows for freeform cell placement through the use of a drag-and drop feature for each code cell. 2D Jupyter was used to conduct a user study where participants completed a data analysis task in the 2D computational notebook environment. The results of this study show that participants had a variety of strategies in their use of the 2D space. Participants also found several advantages of the 2D space as compared to 1D notebooks, including a better ability to reference code and making it easier to find cells. Disadvantages of the extension related to struggles in navigating in 2 dimensions, especially when working on smaller displays that required lots of scrolling in both horizontal and vertical dimensions in order to view the entire notebook. Overall, the study showed favorable opinions of a 2D computational notebook environment. The key contributions of this work are:

- A Jupyter Notebook extension that enables a multi-column layout for notebooks, as well as freeform cell placement in 2D space.

- An overview of results showing how 2D space was utilized when completing a data analysis task from scratch
- Future design and development recommendations that can be used to enhance the benefits of the 2D Jupyter extension, as well as to further the exploration of 2D layouts in computational notebooks.

In the next section, we provide a review of previous work related to our research. In the third section, we describe the design and development of the 2D Jupyter extension. The fourth section provides an overview of the study that was conducted and the results obtained from the study. Finally, we provide a discussion of the findings and potential future work.

Chapter 2

Review of Literature

This research conducted in this work spans two key areas of research: computational notebooks and Space to Think. This chapter will provide an overview of the above areas.

2.1 Computational Notebooks

The design of today's computational notebooks stems from Knuth's idea of "literate programming" [22], where Knuth proposed the combination of expository text with code to make programs more robust and understandable. Computational notebooks make use of this concept by interleaving code, text, and visualizations, providing support for the data analysis process and construction of computational narratives. Data analysts often engage in "exploratory programming" [13] and will write, rewrite and refine code in order to understand their data, test models, and generate desired results. As a result, computational notebooks have seen wide adoption in recent years due to their flexible programming environments as well as the inclusion of features tailored to the needs of analysts.

However, computational notebooks still present with struggles and pain points for users [8], including in exploration and analysis. The incremental and iterative process of data analysis can often result in messy notebooks, as analysts tend to prioritize finding results over clean, readable code [17]. Many analysts consider their notebooks to be "experimental" or "throwaway" [16] and in need of cleaning before presenting or sharing [30]. The process of

cleaning a notebook can also be laborious and tedious in current notebook environments [10]. Head et al. establishes that messiness in notebooks is a result of disorder, where code is run in a different order than presented; deletion, where the user has deleted cells but the effects of the code are retained in the interpreter; or dispersal, where the code producing a single output is spread across several cells [13]. These issues can eventually get in the way of data analysis, as results accidentally get overwritten [30] or analyses are split across many cells, making them difficult to understand [13]. For these reasons, computational notebooks often struggle to bridge the gap between data exploration and constructing coherent computational narratives for sharing or presenting, as analysts are reluctant to share messy notebooks with others [13]. Various tools have been developed to help manage messiness through features such as cell dependency graph visualization [35] or version control systems tailored to data scientists [19].

These issues may be exacerbated by the 1D linear structure of many computational notebooks, especially given the looping nature of the sensemaking process [26, 27]. Scrolling through a long notebook can also be tedious and negatively impact tasks such as debugging or code cleaning. One possible solution may be the use of 2D space in computational notebooks. Participants in the study conducted by Harden et al. [12] found that the ability to arrange a notebook in 2D space was beneficial in tasks such as comparative analysis and organization of the notebook. Stickyland [33] also explores the potential for non-linear layouts of notebooks through various features such as floating cells that can be freely resized and repositioned, as well as the ability to "stick" cells to a dock that remains persistently at the top of the screen. Other tools such as Fork-It [34] enable the use of 2D space by allowing for side-by-side views of alternative code paths, which helped users in data exploration and debugging code.

2.2 Space to Think

The concept of "space to think" was identified in work done by Andrews, Endert and North [3], in which they showed that large, high resolution displays are beneficial to the sensemaking process. They classified analysts' use of space into two categories: memory and semantics. First, they found that the large displays enabled analysts to externalize memory, since the space allowed documents and files to be spatially persistent. Analysts were able to remain aware of the current document space through visual cues, and did not have to search for information or switch away from the current task. Second, space to think provided analysts with the ability to encode meaning within the space through the organization of their files. Actions such as clustering similar documents together, or ordering documents in a certain way, reduced the need for complex internal models to keep track of information. Space to think has also shown benefits in collaborative environments [20], and recent studies have expanded the concept of space to think into virtual reality and immersive spaces [9, 23]. Other research papers have used space to think as an influence on their designs [24, 28, 29].

Computational notebooks in their current state do not sufficiently support space to think, and users will often create workarounds such as opening up multiple versions of the notebook side-by-side, or even exporting their code into another tool to facilitate better data exploration [8]. The use of 2D space may help enable space to think in computational notebooks [12], and to this end, several tools have been created to explore the potential of this idea. Einblick [2] and CoCalc [14] are interactive whiteboard interfaces where users can create code cells and move them freely around the canvas. CodeBubbles [5] uses code "bubbles", or editable fragments of code, that can be clustered together to form "concurrently visible working sets". VisSnippets [7] allows users to connect blocks of code via arrows in order to form a display of the 2D visual dataflow.

Chapter 3

Design and Development

In this chapter, we provide an overview of the design and development of the 2D Jupyter extension.

3.1 2D Jupyter Design

Based on the results found in the study done by Harden, et. al [12], we identified two primary goals in the design of 2D Jupyter.

Enable a multi-column organization of cells: Harden's study found that all participants used columns as either a primary organizational strategy or as a low level strategy in their organization of cells in the 2D space. As such, we chose to focus on enabling multi-column layouts in the 2D Jupyter prototype. Features were implemented to support this layout type, such as buttons to add and delete columns, and a button to add cells to a specific column.

Provide flexibility in placement of cells: Participants in Harden's study also liked the flexibility in code cell placement in 2D space. Notably, almost all participants thought that the 2D environment could be useful for comparing code and visualizations by placing cells side-by-side. We wanted to support this flexibility in the 2D Jupyter extension, and so we implemented different methods of moving cells and columns around the 2D space. Buttons at the top of the columns allow the user to move the entire column to the left or right. Each

cell also contains a drag and drop ability that allows freeform placement of the cell. Users are able to move cells higher or lower in the column, move cells to a different column, or even place them outside of a column entirely.

Figure 3.1 shows a new notebook that has been created with the extension enabled. Like the original Jupyter Notebook, a single cell is generated for users to begin coding. However, the interface has been modified so that the initial cell is placed within a column object, denoted by the toolbar above the cell.

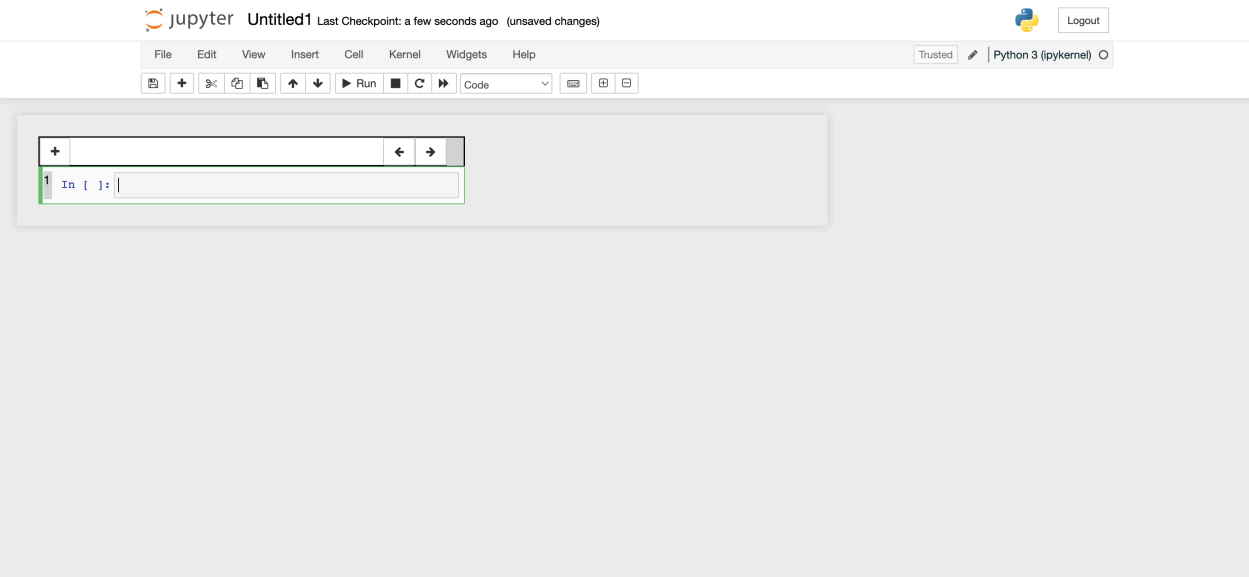


Figure 3.1: A new Jupyter Notebook with the 2D Jupyter extension enabled

3.1.1 Adding, Deleting, and Selecting Columns

User may add and delete columns using the buttons shown in Figure 3.2. By default, new columns will appear at the rightmost end of the notebook, and deleting a column will remove the rightmost existing column. Users may also select a column by clicking on the whitespace at the top of the column, which highlights that column in blue, as seen in Figure 3.3. When a column is selected, clicking the "Add Column" button will place the new column immediately

to the right of the selected column. Clicking the "Delete Column" button when a column is selected will delete the selected column.

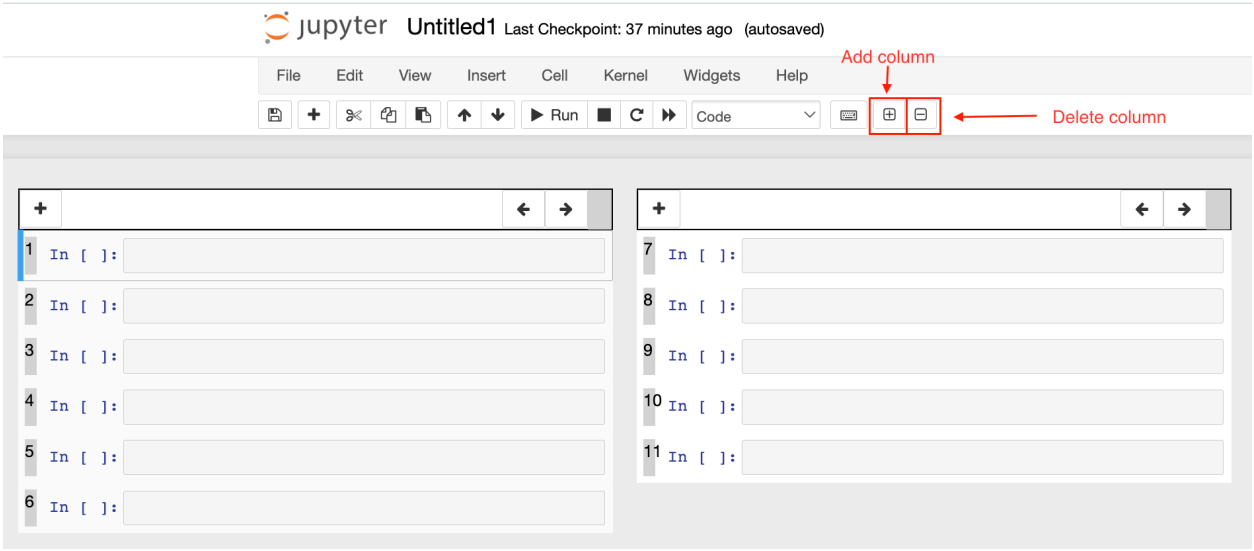


Figure 3.2: Buttons to add or delete a column

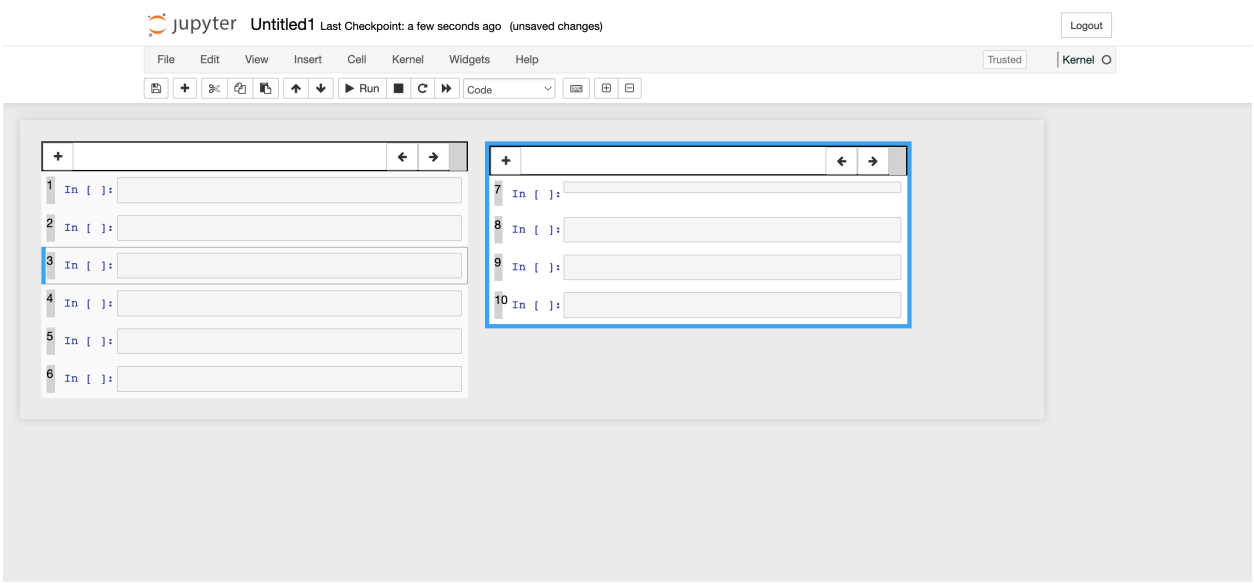


Figure 3.3: Appearance of a column when selected

3.1.2 The Column Toolbar

At the top of each column is the column toolbar (Figure 3.4), which is made up of 3 buttons that can be used to modify the columns. The first button, denoted by the plus symbol (Figure 3.5a) can be used to add a cell to the column. This new cell will be added to the bottom of the column, below all existing cells within the column. The left and right arrows, shown in Figure 3.5b can be used to move the entire column one place to the left or right respectively. The gray box at the rightmost end of the toolbar can be used to modify the width of the column by clicking and dragging this box left and right.

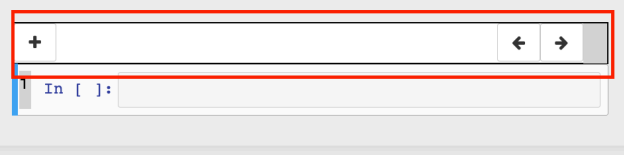
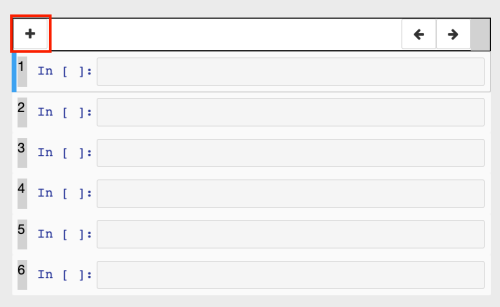
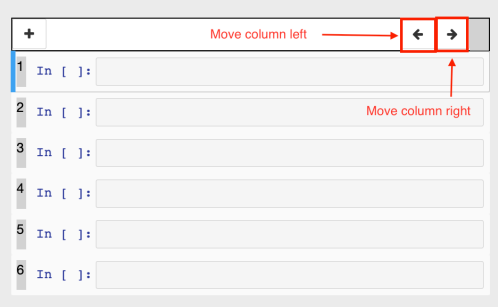


Figure 3.4: A column toolbar



(a) Button to add a cell to a column



(b) Button to move column left or right

(c) Box to resize column width

Figure 3.5: Features of the column toolbar

3.1.3 Freeform Cell Placements

The gray box to the left of each code cell, highlighted in Figure 3.6 is used to place the cell at any location in the notebook. The user can click this box and drag the cell around the notebook. If the user releases the cell over a column, the cell will be added to that column at that position. If the user releases the cell over empty space, the cell will be converted to a scratch cell, as shown in Figure 3.7. Scratch cells can be used for temporary actions such as testing code chunks. These cells are omitted from the "Run All" process. When "Run All" is selected, cells within columns will run top to bottom within the column, and columns will be run left to right.

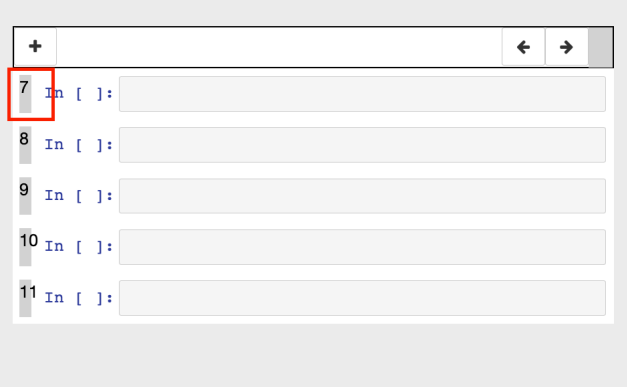


Figure 3.6: Drag and drop box on a cell

3.1.4 Cell Indices

To help keep track of the run order of cells, the run index of each cell has been appended to the cell inside the drag and drop box, as shown in Figure 3.8. As cells are moved around the notebook, these numbers will automatically update to reflect the run order of the cells. Scratch cells, which are not included in the "Run All" command, do not have a cell index number.

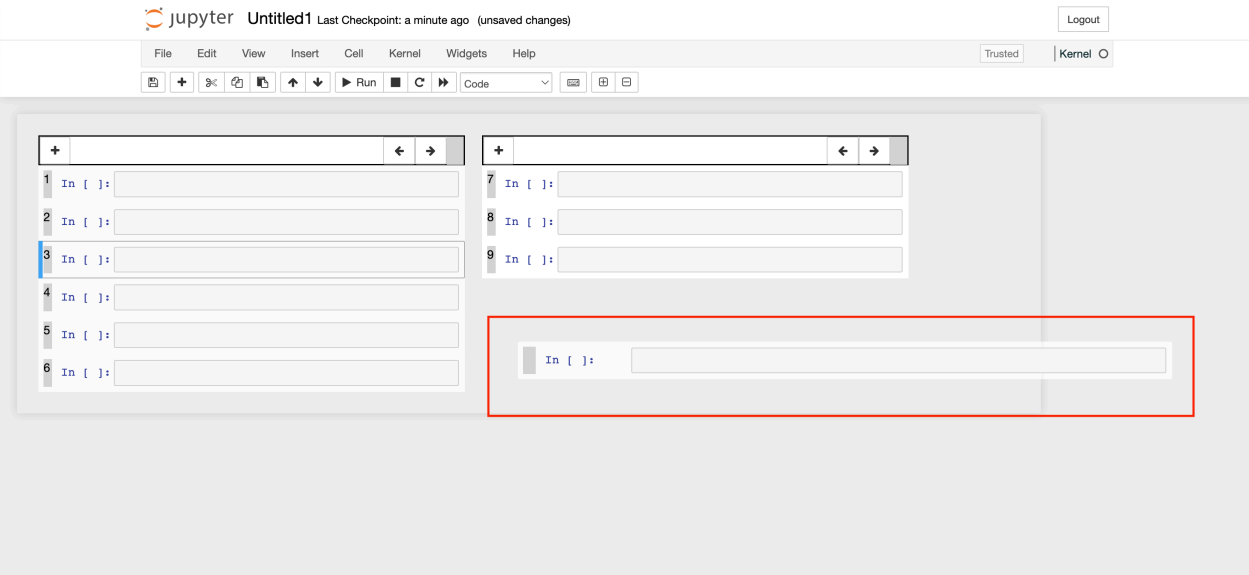


Figure 3.7: Appearance of a scratch cell

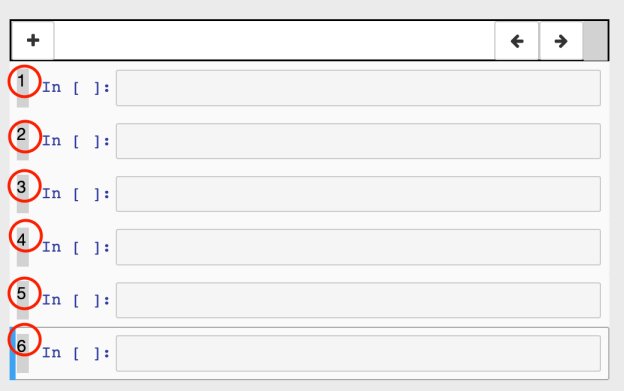


Figure 3.8: Cell index labels to indicate cell run order

3.2 2D Jupyter Development

3.2.1 Extension Architecture

Extensions are add-ons to Jupyter Notebooks that add functionality to the notebook interface [32]. Most extensions are written in JavaScript and are loaded locally in the browser, as shown in Figure 3.9. The package `jupyter_contrib_nbextensions` must be installed

to enable extensions, and this package additionally contains a set of unofficial extensions contributed by the community. During installation, the package copies the JavaScript and CSS files for the extensions into the server's search directory, which allows the server to load the extensions on startup.

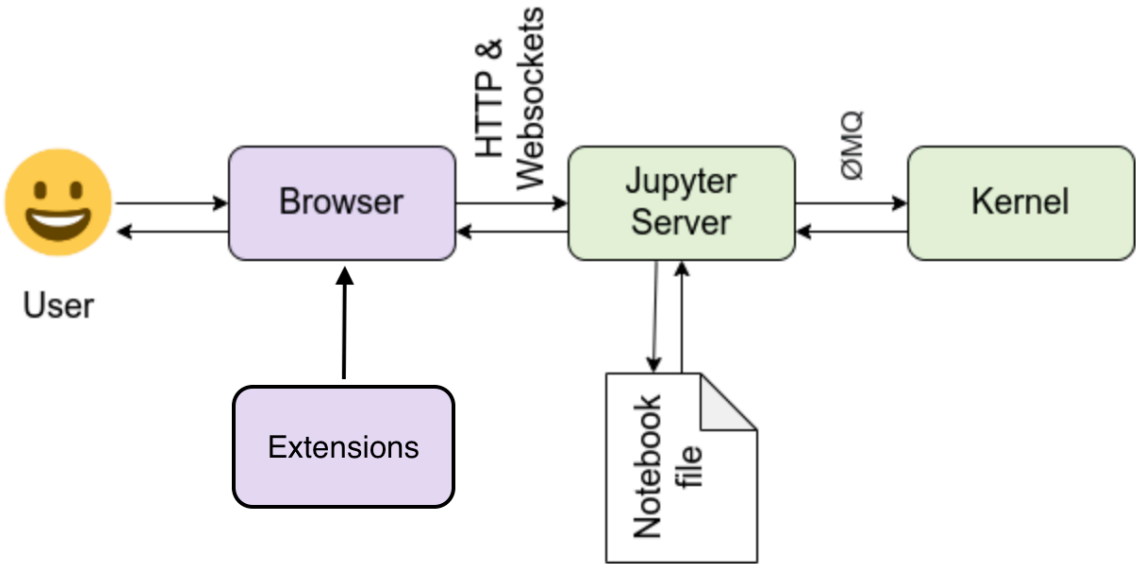


Figure 3.9: Diagram of Jupyter Notebook architecture [31]

Extensions are installed into a `src/jupyter_contrib_nbextensions/nbextensions` folder, and each extension is contained within a subdirectory in this folder. The subdirectory for each extension contains, at minimum, the **extension.js** file that implements the extension and an **extension.yml** file that describes the extension to the server configuration. Optionally, there may also be an **extension.css** file included for styling, and a **README.me** to describe the extension in a readable format. The JavaScript file for each extension must contain a function called `load_ipython_extension` that is called by the server to load the extension.

3.2.2 Extension Implementation

2D Jupyter was developed from scratch using JavaScript. It is primarily a modification of the front-end of Juypter Notebook and includes changes to pre-defined functions as well as the object definitions of CodeCell and TextCell objects. Implementation of the extension required changes to the styling of the interface, developing a method of saving the 2D layout, and maintaining the run order of cells.

Modifying the Interface

To support the 2D layout, the styling of existing HTML elements have been modified, and additional elements were added to the interface. All positioning and styling of elements is done through the use of JavaScript and jQuery within the extension file.

HTML divisions have been created for each column and cells are placed within the column division, instead of within the notebook container (Figure 3.14). Column divisions also contain an HTML element for the column toolbar, which contains the buttons for adding a cell and moving the column left or right. Additionally, an HTML element was created for the box to resize columns and added to the end of the column toolbar division. The toolbar also contains a division to denote the clickable area of the toolbar for column selection. Figure 3.10 shows the structure of a column element in the notebook HTML.

To include the drag-and-drop box and the run index in the cell elements, the object definitions of CodeCell and TextCell objects have been overridden. JavaScript and jQuery is used to create an HTML element for the drag-and-drop box and attach it to the cell element. Mouse listener functions have also been attached to the drag-and-drop box element to enable the repositioning of the cell. The run index is added to the drag-and-drop element. Figure 3.11 shows the structure of the modified cell element in the HTML DOM.

Figure 3.10: DOM structure of a column. Each column is a division in the notebook HTML. Within the column is the column toolbar division, which contains elements for each button in the toolbar. Cell elements are placed after the toolbar divison.

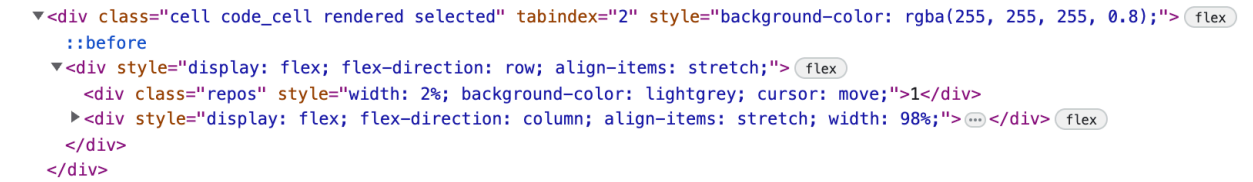


Figure 3.11: DOM structure of a cell. The drag-and-drop box is an element in the cell division with the class name "repos". The run index is text within the drag-and-drop box element.

Saving the Layout

When a notebook is saved, a JSON representation of the notebook is sent to the Jupyter server to be used as a checkpoint file. The JSON file contains metadata about the notebook, as well as metadata for each cell in the notebook, and this data is used to restore the notebook next time it is loaded. In order to save the 2D layout, 2D Jupyter adds in metadata fields to both the notebook and cell objects. At the notebook level, a new metadata field "columns" is added to store the number of columns that exist in the notebook (Figure 3.12). At the cell level, the position of the cell is saved in one of two ways. If the cell is located within a column, a "column" field is added that represents the ID of the column it was in, as well as an "index" field that represents the position of the cell in the overall run order (Figure 3.13a). Alternatively, if a cell was outside of a column, such as a scratch cell, a "spatial" field is

stored in the metadata, which contains the page coordinates of the cell through "left", "top" and "z-index" subfields.(Figure 3.13b). When a saved notebook is reopened, the notebook metadata is used to generate the correct number of columns. Cell metadata is then used to place each cell in the correct position relative to the columns.

```
{
  "columns": 3
}
```

Figure 3.12: Metadata saved for the notebook object

```
{
  "index": 2,
  "column": 1
}
```

(a) Metadata for a cell that is in a column

```
{
  "index": 16,
  "spatial": {
    "left": 591,
    "top": 611.890625,
    "zIndex": 1001
  }
}
```

(b) Metadata for a scratch cell

Figure 3.13: Cell metadata for saving layouts

Maintaining Run Order

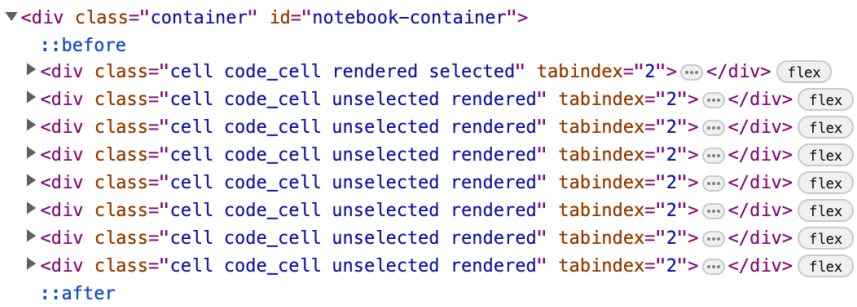
In the 1D notebook environment, the "Run All" command will run all cells in the notebook from top to bottom. However, in the 2D environment, we run cells from top to bottom in the column, and then run each column left to right. To accomplish this, we take advantage of the fact Jupyter Notebooks uses the HTML DOM to determine the run order of cells in the notebook.

In Jupyter Notebooks, each cell is a distinct element in the DOM, and when the extension is not enabled, all cells are contained within the notebook division (Figure 3.14a). As cells are moved up and down the notebook, the cell element is moved up or down in the DOM

list. However, in 2D Jupyter, cells are contained within column divisions (Figure 3.14b). When a cell is moved to a new position in the notebook, the cell element must not only be placed in the correct column div, but also at the correct position in the list of cell elements within the column. We accomplish this through implementation of mouse listeners in the drag and drop box. When the box is clicked, the coordinates of the cell are recorded as the cell is moved around the page. Nested functions within the mousedown action enable page scrolling if the cell is dragged to the edges of the page. When the cell is released, the mouseup function checks for a collision with another cell. If there is a collision, the correct column and position is calculated using the colliding cell, and the cell that is being moved is placed back into the DOM at this position. The new column number and cell index are then recorded in the cell metadata. If no collision is occurring when the cell is released, then the cell element gets placed back into the DOM outside of the notebook div, removing it from the run order entirely. The coordinates of where it was released on the page are stored in the cell metadata. Each time a cell is moved, a `reindex` function is called to update the indices of all cells.

3.2.3 Using the Extension

To begin using the extension, users must first install the `jupyter_contrib_extensions` Python package using `pip install jupyter_contrib_extensions`. Once this installation is complete, users must then install the prepackaged JavaScript and CSS files using the command `jupyter contrib nbextension install --user`. This enables the extensions package to communicate the location of the extension files to the Jupyter server. Users must then download the 2D Jupyter files from the GitHub repository and place them in a subdirectory inside `src/jupyter_contrib_nbextensions/nbextensions`. The 2D Jupyter extension can then be enabled by either going into the Jupyter extensions configuration



(a) DOM structure of a 1D notebook. Cells are placed directly in the notebook container.

(b) DOM structure of a 2D notebook. Cells are placed in a column division, which then are placed in the notebook container.

Figure 3.14: Comparison of DOM structures of 1D and 2D notebooks.

interface and selecting the checkbox for 2D Jupyter (Figure 3.15), or by running the command `jupyter nbextension enable 2D-Jupyter` in the terminal. To disable the extension, uncheck the box for 2D Jupyter and refresh any active notebooks.

Additional instructions and information about installation can be found on the 2D Jupyter Github repository (Appendix A).

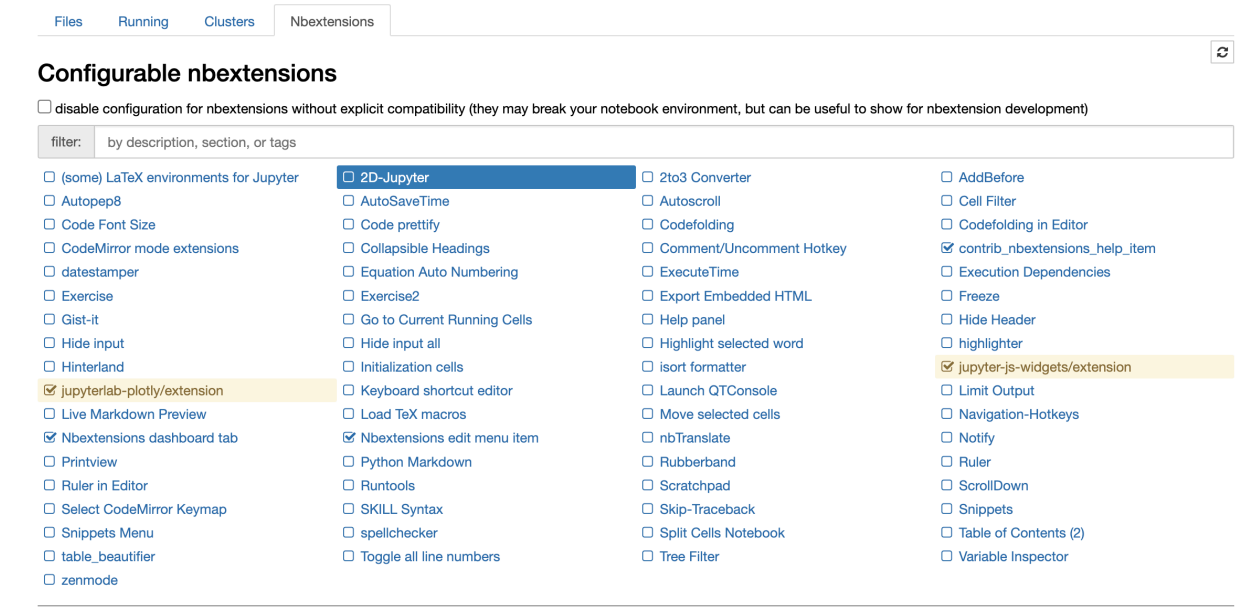


Figure 3.15: Jupyter extensions configuration interface

Chapter 4

Evaluation

To evaluate the usability of 2D Jupyter, a user study was conducted where participants given the opportunity to use the extension to create a computational notebook. Through this study, we aimed to investigate the ways that 2D space might be used in creating computational notebooks, as well as the advantages and disadvantages of the extension. We also wanted to inform design opportunties for future development in hopes of improving the 2D Jupyter extension. Throughout this study, qualitative data was collected in the form of screen recordings, recorded interview sessions, and survey results. The survey also provided quantitative data in the form of Likert-scale questions.

4.1 Study Design

4.1.1 Participants

Participants for this study were recruited from two Virginia Tech computer science classes via class announcements. All participants were computer science students, with 5 participants being graduate students and 4 participants being undergraduate students. Additionally, all participants were required to have prior experience with data analysis in Python and working in Jupyter Notebooks.

4.1.2 Procedure

A primary goal of this study was to evaluate the ways that 2D Jupyter might be used in creating computational narratives. As such, we wanted to provide participants in the study with the opportunity to freely build their notebooks with minimal restrictions or limitations. Additionally, we wanted to to create a task that would enable the use of all features of the extension. To this end, we created a data analysis task that asked participants to analyze COVID data for counties in the United States. We provided participants with a Jupyter notebook file that contained the task instructions (found in Appendix B), as well as a COVID dataset [1] and a population dataset [6]. Using these datasets, participants were asked to answer a set of questions about COVID numbers in the various states and counties.

An initial meeting was scheduled with each participant to give them an overview of 2D Jupyter as well as introduce them to the data analysis task. Participants were then given about two weeks to complete the task. We expected the task to take several hours to complete, and as such, we allowed participants to work on it on their own time so that they could fully explore the features of 2D Jupyter and conduct a thorough analysis. Once the task was completed, an interview session was scheduled with each participant. Participants were asked several questions about their experiences using 2D Jupyter, as well as their opinions and feedback for the extension. At the end of the interview, each participant was also asked to complete a survey that contained 7 Likert scale questions and 2 short answer questions. The Likert scale questions focused on the benefits of a 2D layout in various parts of a data analysis task and the usability of 2D Jupyter. The short answer questions prompted users to provide feedback and suggestions on the 2D Jupyter extension.

2 participants were given access to large 64-inch displays and were asked to complete the task using these displays. While the extension provides a virtual infinite canvas, we wanted

to know whether the physical display space available impacted the way that a 2D notebook environment is used. Many of the benefits of space to think relate to the user's ability to see all of their work at once, which may be challenging to accomplish when using smaller laptop display. We hypothesized that a larger display would enhance the benefits of a 2D environment, potentially making 2D Jupyter more useful on bigger screens.

A modified version of the data analysis task was created after evaluation of results from the first 5 participants. The original task (B.1) was a relatively simple analysis of the data provided, and as such, participants did not feel it was necessary to use all the components of 2D Jupyter. Features such as moving columns or freeform cell placement was not required for participants to be able to conduct their analysis and answer the questions about the data. As such, the modified version of the data analysis task (B.2) that introduced additional analysis requirements and required more complex tasks that encouraged participants to make use of 2D Jupyter features. In total, 4 participants completed the modified version of the task.

4.2 Results

Study participants were interviewed about various aspects of their experience using 2D Jupyter and additionally completed a survey. We focused on four main points of interests in the results: strategies in use of space, advantages of 2D over 1D, disadvantages of 2D over 1D, and participant feedback and suggestions.

4.2.1 Strategies in Use of Space

In the participants who completed the first version of data analysis task, we found two main strategies in the use of 2D space. The first strategy, which we call **"Column Per**

Question", was used by three participants (**P1**, **P2**, **P3**). With this strategy, participants divided created a separate column for each question asked of them in the task. Each column they created contained the entirety of the data analysis required to complete one of the questions, with the exception of one participant who added an additional column to the analysis for the second question in order to reduce vertical scrolling (**P3**). The notebooks created by **P1**, **P2** and **P3** can be seen in Figures 4.1, 4.2, and 4.3 respectively. The second strategy was used by one participant (**P4**) and can be categorized as **"Column Per Data Analysis Task"**. This participant used the columns to separate the steps of the data science workflow: data processing, data exploration, creating visualizations, and finally evaluating results. This participant's notebook can be seen in Figure 4.4.

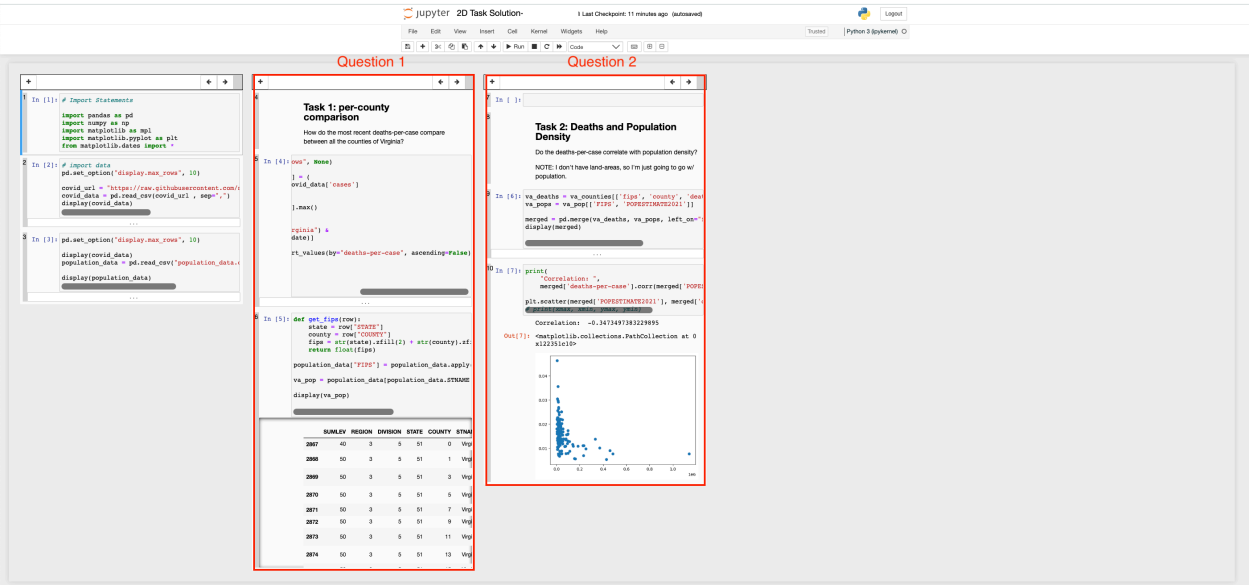


Figure 4.1: Notebook completed by **P1** using the "Column Per Question" strategy. Columns were used to separate the analysis for each question in the overall task.

In the participants who completed the second version of data analysis task, we also found 2 primary categories of use of space. 2 participants (**P7**, **P9**) used what we call the **"Linear" strategy**, and kept their notebook primarily in a 1D linear layout, using the 2D space for smaller actions such as one-to-one cell comparison or referencing. **P7** did the majority of

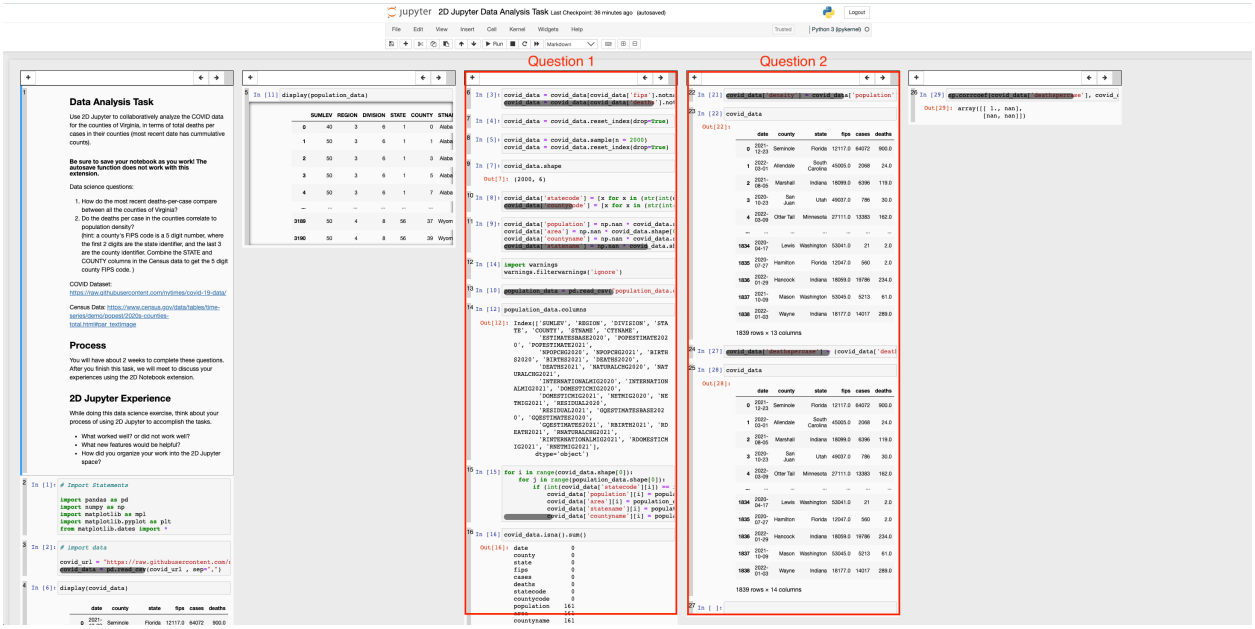


Figure 4.2: Notebook completed by **P2** using the "Column Per Question" strategy. With the exception of a couple of extraneous cells, columns were used to seperate the analysis for each question in the overall task.

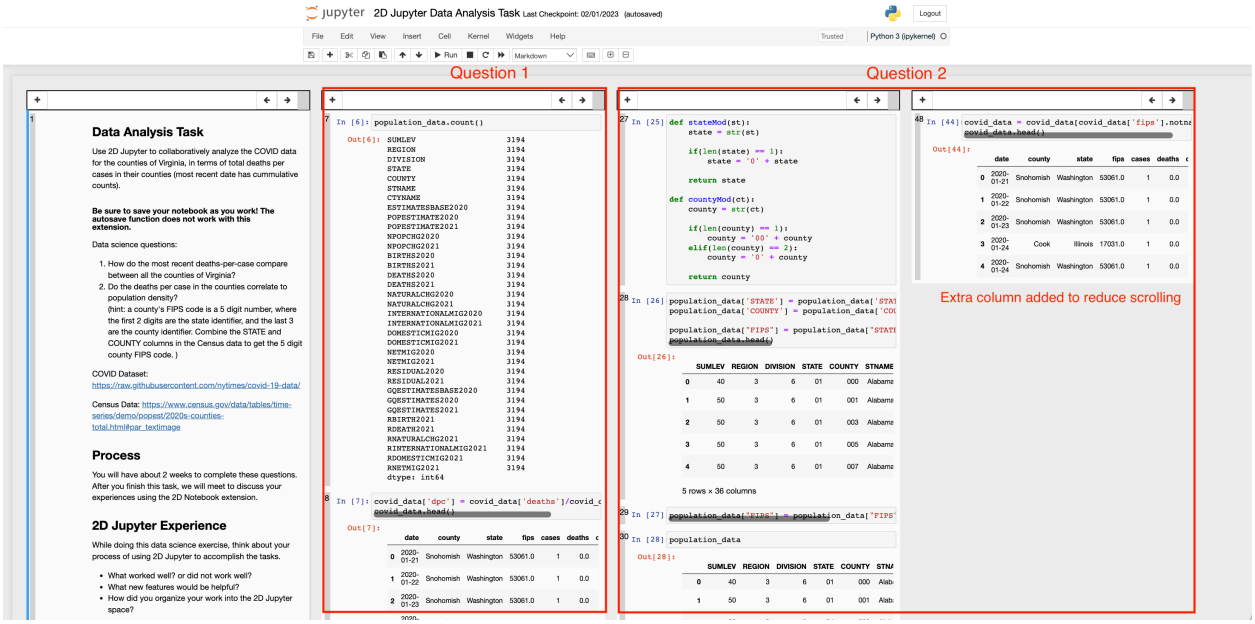


Figure 4.3: Notebook completed by **P3** using the "Column Per Question" strategy. Columns were used to separate the analysis for each question, with the addition of an extra column to reduce vertical scrolling.

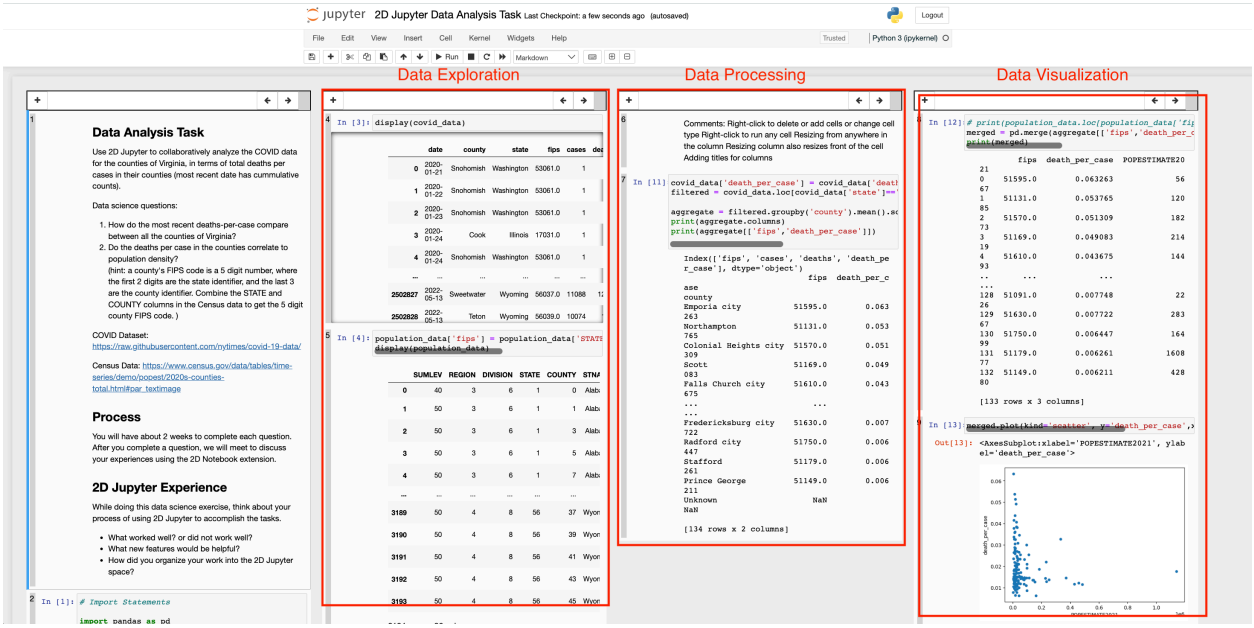


Figure 4.4: Notebook completed by **P4** using the "Column Per Data Analysis Task" strategy. This participant sectioned their notebook by the data analysis steps, creating a column for each of data exploration, data processing, and data visualization.

their analysis in a single column, and created another column only as means of referencing code or visualizations. This participant's notebook can be seen in Figure 4.5. **P9** did not create additional columns at all, and instead kept all their work in a single column. This participant used a scratch cell to keep the task instructions in view at all times. As they worked on their notebook, they used the drag and drop ability to move the cell with the instructions down on the page. Their notebook can be seen in Figure 4.6.

Two participants (**P6**, **P8**) completed the data analysis task using a large 64-inch 4K screen. With the larger screen space, these participants were able to create more columns without needing a significant amount of vertical scrolling to view the entire notebook. **P6** used a strategy we call **"Reduce Page Length"**, where they created columns as a method of reducing the amount of vertical scrolling. Instead of using columns to break up sections of the notebook, this participant created a new column when they felt the vertical length of the page was becoming too long. Their notebook can be seen in Figure 4.7. **P8** used

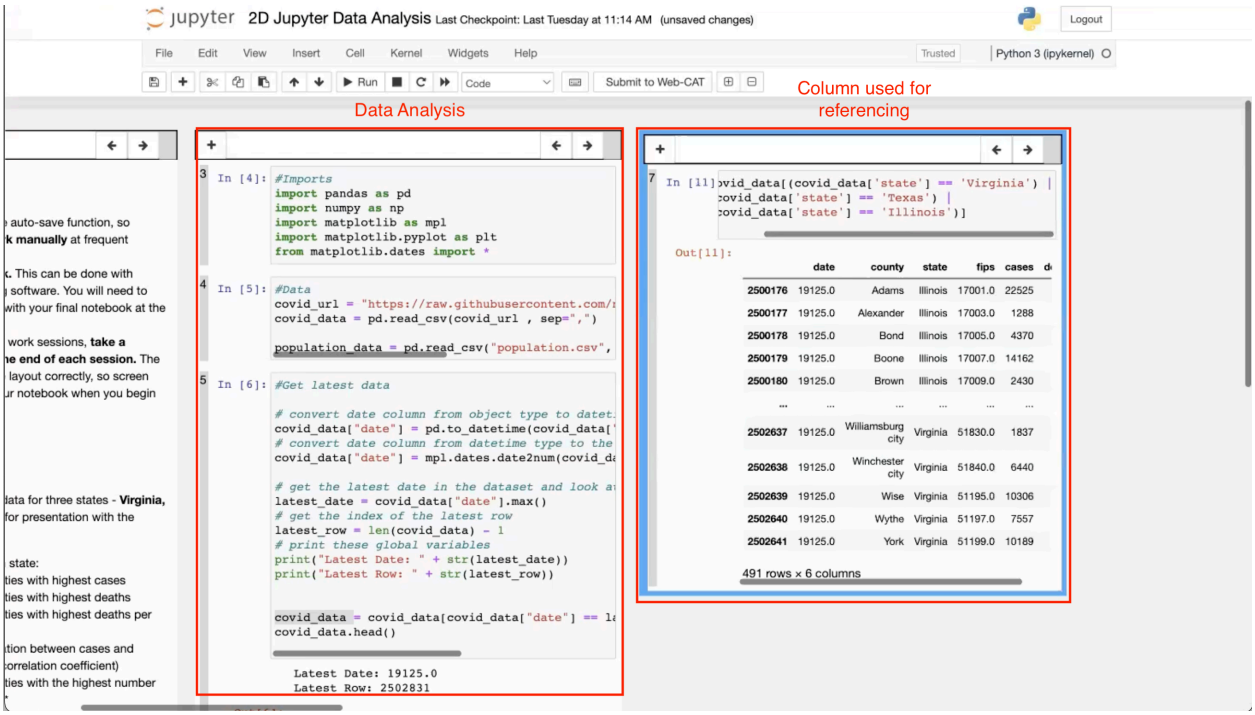


Figure 4.5: Notebook completed by **P7** using the "Linear" strategy. The majority of the data analysis was done in a single column, with an additional column only being used to reference data or compare visualizations.

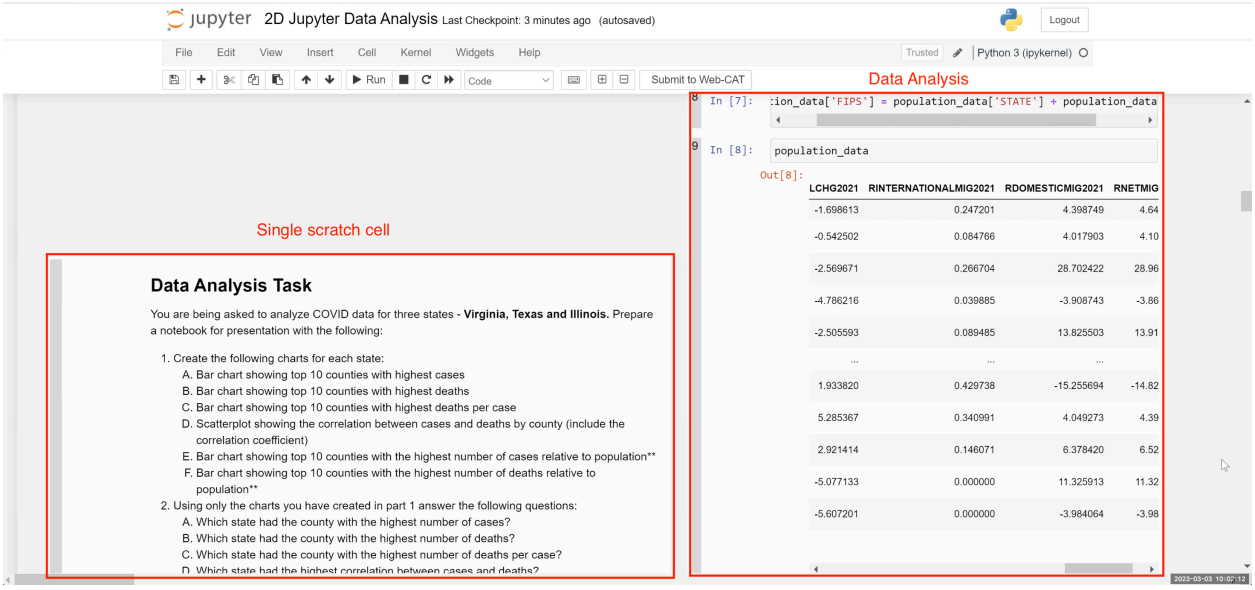


Figure 4.6: Notebook completed by **P9** using the "Linear" strategy. All of the data analysis was done in a single column. A scratch cell was used to reference the task instructions.

the **"Column Per Data Analysis Task"** strategy, primarily using columns to separate the data science subtasks. Additionally, **P8** created new columns when they wanted to align visualizations for ease of comparison. The notebook created by **P8** can be seen in Figure 4.8.

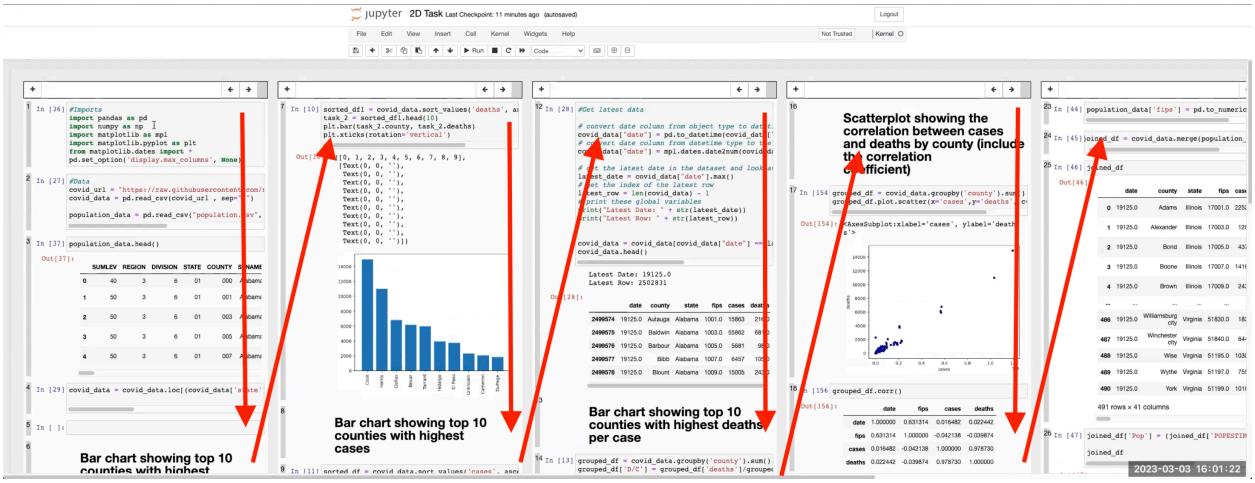


Figure 4.7: Notebook completed by **P6** using the "Reduce Page Length" strategy. Columns were created as a means of reducing the amount of vertical scrolling required to view the entire notebook.

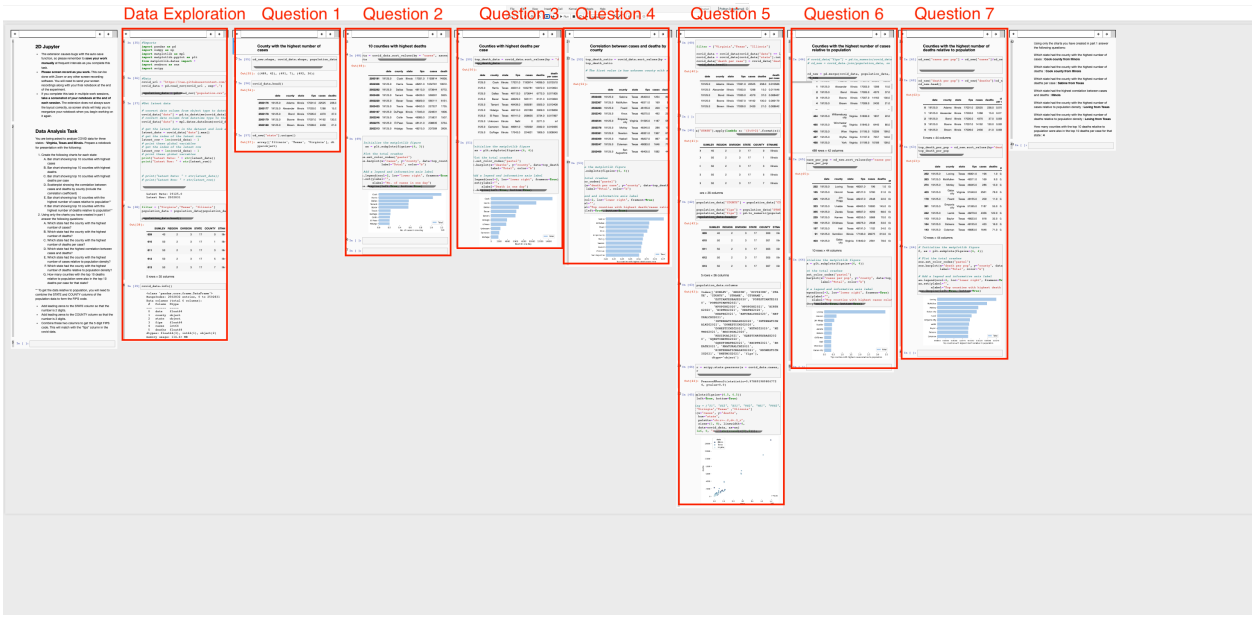


Figure 4.8: Notebook completed by **P8** using the "Column Per Question" strategy. Each column contains the data analysis needed to answer a single question in the task.

In one unique case, **P5** opted to not complete the given data analysis task, but instead used the 2D Jupyter extension to complete their own work for a class project. This participant was a student in an artificial intelligence class and was working on a project that involved building their own AI model that could play a game. **P5** also used the "Linear" strategy and maintained a single column of cells alongside a scratch cell in which they tested parameters for their AI model.

Summary of Strategies

Table 4.1 shows a summary of the strategies used by the participants to organize their notebooks. In total, 5 participants used columns as a means of sectioning their notebook. These participants found the 2D space helpful not only in organization of their notebook, but also in conducting their analyses. 3 participants continued to utilize a primarily 1D linear organization in their notebooks, suggesting that they used the 2D features as supplements to the computational notebook interface rather than as a means of providing new organizational opportunities.

Table 4.1: Strategies in Use of Space

Strategy	*Total Participants*	*Participants Using This Strategy*
Column Per Question	4	P1, P2, P3, P8
Column Per Data Analysis Task	1	P4
Linear	3	P5, P7, P9
Reduce Page Length	1	P6

Number of Columns and Cells Used by Participants

The number of columns used by participants can be found in Table 4.2. This count excludes the first column containing the task instructions, as this was present in the initial notebook given to participants. We found that those who used the "Linear" strategy created only 1 or 2

columns, typically relying on a scratch cell or only using the second column to reference cells one-to-one. Participants who used "Column Per Question" or "Column Per Data Analysis Task" created 3 or 4 columns, except for one participant in these categories who conducted the task on the large display. The participants who used the large display created the most columns, with 6 and 10 columns present in their notebooks, suggesting that the use of a larger display space lends itself to increased utilization of the 2D space.

In Table 4.2, the average cells per column was calculated by diving the total number of cells in the notebook by the total number of columns in the notebook. Participants varied in the number of cells they used per column, with no apparent pattern relating their overall organization strategy. It is likely that the number of cells in a column is dependent on the personal development style of the user, as cell height can vary depending on the lines of code in a cell, as well as the size of any output visualizations generated by that cell.

Table 4.2: Number of Columns and Average Cells per Column

Participant	*Number of Columns Used*	*Average Cells per Column*
P5	1	3
P9*	1	30
P7*	2	6
P1	3	3.3
P2	3	8
P3	4	11.25
P4	5	6.25
P6*	6	5.83
P8*	10	4

Starred participants completed the modified version of the task. P5 did not complete either of the provided data analysis tasks but instead used 2D Jupyter for a class assignment.

Features Used by Participants

During the interview session, participants were asked which features of 2D Jupyter they used. Table 4.3 shows the total number of participants who used each feature. Most participants used features related to organization, with 7 people using the ability to add and delete columns and 6 people using the ability to move cells around. Other features such as the ability to move columns left and right was only used by 3 participants, and only 2 participants used the ability to resize columns.

Table 4.3: Features Used by Participants

Feature	*Total Participants*
Adding/Deleting Columns	7
Moving Columns Left/Right	3
Freeform Cell Placement	6
Resizing Columns	2

4.2.2 Advantages of 2D over 1D

During the study, participants were asked to identify areas where they felt 2D notebooks had an advantage over 1D notebooks. Participant responses are summarized below.

- **Referencing other cells or comparing charts and visualizations is easier in 2D**. 5 participants (**P2**, **P3**, **P4**, **P7**, **P9**) liked that the 2D environment made it easier to refer to cells for reusing code or comparing charts and visualizations. The ability to place cells side-by-side reduced the amount of scrolling required while conducting the data analysis and did not disrupt organization of the notebook. **P2** pointed out that "...when I have to compare two data frames...side-by-side that's really useful."

- **2D notebooks make it easier to locate specific code or data.** 3 participants (**P1**, **P4**, **P8**) noted that the 2D environment made it easier to keep track of cells. The

ability to organize the notebook in 2D space meant that when looking for a certain cell, users could instantly identify the section a cell was in, instead of having to scroll to find the section. **P4** noted "It was easier for me to find the exact cell I was looking for."

- **The 2D environment allows users to view more of their code at once.** 1 participant (**P6**) liked that they were able to view more of their code at once, especially when using a larger display that allows more columns to fit on the screen. "On such a big screen...I feel like so much can fit in it..." (**P6**).

4.2.3 Disadvantages of 2D over 1D

Participants also identified several areas where 2D had a disadvantage over 1D notebook, which are listed below.

- **Extra horizontal scrolling is required, especially on smaller screens. P3** noted that while vertical scrolling was reduced, the smaller screen size of typical laptops would require more horizontal scrolling in order to view the entire notebook. They felt that this was a "major disadvantage" of a 2D notebook layout.
- **It can be harder to find a "lost" cell. P4** suggested that due to the extra dimension of space, it may be harder to find a specific cell in the 2D environment if the user doesn't remember the location of the cell. Users may create scratch cells that can be difficult to locate in a larger notebook, or a lack of organization in the notebook results in the user forgetting which column a cell is in. They pointed out that in a 1D notebook, a user could continue to scroll up until they found the cell they were looking for, but "...in 2D I have to find the right column and then I have to find the right position."

- **Presenting a 2D notebook is more challenging.** Two participants (**P1**, **P5**) felt that presenting a notebook created in the 2D Jupyter environment would be harder than a 1D notebook. **P1** commented that navigation through two dimensions made it harder to read the entirety of the notebook, and it could potentially look cluttered if there are too many columns and cells. **P5** pointed out that the extension did not support exporting the notebook layout to an HTML or PDF format, making it harder to share the notebook.

Three participants (**P2**, **P6**, **P7**) did not identify any disadvantages to the 2D Jupyter environment since they felt that the extension did not take away any features from the original 1D environment. **P7** noted "...you can use this as a 1D space...the choice to be able to make it 2D and and work side by side is useful." **P6** said "I don't see there being...any sort of disadvantage or any type of limitation that 2D has compared to 1D. If anything...the opportunities are endless."

4.2.4 Feedback and Suggestions

Throughout the study participants were also given the opportunity to provide feedback or suggestions about the 2D Jupyter extension.

- **Individually scrollable columns would help with comparison tasks.** Two participants (**P3**, **P8**) expressed a desire for individually scrollable columns. Both participants pointed out that if they wanted to, for example, compare a cell at the top of one column with a cell at the bottom of another column, there wasn't a way to place them side by side while maintaining the organization of the notebook.

- **Toolbar features should be accessible from any point in the notebook.** Some

participants pointed out that users would have to scroll to the top of the page in order to access the toolbar buttons, and suggested making the toolbars accessible from any point in the notebook. Similarly, **P1** and **P4** wanted to be able to to resize the columns from any location in the notebook. **P4** said "...if it was possible to resize [the column] directly from [the middle of the column] instead of having to go up and resize, that would be good...".

- **It would be helpful to group cells together in some way. P9** suggested being able to link scratch cells to other cells within a column in a way such that the cells could be moved as a group. Often the code needed for a single result is spread across several cells, so the ability to group related cells together could potentially enhance benefits of the 2D environment.

- **A minimap would help with notebook navigation. P1** pointed out that in larger notebooks, navigation could get more difficult and suggested adding a minimap feature to help the user orient themselves within the notebook environment.

Overall, feedback on the 2D Jupyter extension was generally positive. While some participants experienced a learning curve due to being accustomed to the 1D environment, they still found the 2D Jupyter extension easy to understand. "...After that small little learning curve, I think everything else...was super straightforward." (**P6**)

4.2.5 Survey Results

Although all 9 participants were asked to complete the survey, 2 participants chose to not answer the survey questions, which resulted in receiving only 7 participant responses. The results of the Likert-scale questions can be seen in Figure 4.9.

Statement	Strongly Agree (2)	Agree (1)	Neither agree nor disagree (0)	Disagree (-1)	Strongly disagree (-2)	Average
I found the use of the 2D extension to be beneficial in organizing my notebook.	2	3	1	0	1	0.71
I found the use of the 2D extension to be beneficial in data processing.	1	2	3	0	1	0.29
I found the use of the 2D extension to be beneficial in creating visualizations.	3	2	1	0	1	0.86
I found the use of the 2D extension to be beneficial in debugging my code.	0	4	2	0	1	0.29
It was easy to understand how to use the 2D extension.	2	5	0	0	0	1.29
It was easy to navigate in the 2D notebook.	0	5	1	1	0	0.57
I prefer using the 2D notebook environment over the traditional notebook environment.	0	3	4	0	0	0.43

Figure 4.9: Results of Survey

Survey results show that participants had a generally positive opinion of 2D Jupyter. 5 participants agreed or strongly agreed that the 2D extension was beneficial in organizing the notebook. 3 participants agreed or strongly agreed with the statement that the 2D extension was beneficial in data processing, with 3 participants feeling neutral on this statement. 5 participants agreed or strongly agreed with the statement that the use of the 2D extension to be beneficial in creating visualizations. 4 participants agreed that the 2D extension was beneficial in debugging their notebooks. All 7 participants who completed the survey agreed that it was easy to understand how to use the 2D extension. Additionally, 5 participants agreed that it was easy to navigate in the 2D environment.

Notably, responses to the last question were more neutral than expected, given the positive responses to previous questions. 4 participants were neutral on their preference of 2D over 1D, with only 3 participants agreeing that they preferred the 2D environment over the 1D environments. While participants found areas where 2D Jupyter had benefits in completing data analysis, it may be the case that these benefits did not outweigh the disadvantages found. Problems such as excess scrolling or a sense of clutteredness were cited by participants, potentially dampening the advantages of 2D Jupyter. Additionally, participants may

be more used to the 1D environment and experienced a small learning curve in their use of 2D Jupyter that impacted their ability to fully utilize the 2D layouts. It is also worth noting that the negative responses to the first four questions all came from the same participant. Interestingly, this participant also had a neutral response to the last question regarding a preference for 2D over 1D, suggesting that while they did not feel that 2D was beneficial in creating their notebook, the 2D environment also did not significantly detract from the computational notebook experience, resulting in no preference for either the 1D or 2D environment.

Chapter 5

Discussion

A goal in conducting the study was to find whether a 2D environment provided additional benefits over the 1D environment, and if so, in what aspects. Participants responses during the interview session provided insights into the specific features of the interface that gave 2D notebooks an advantage or disadvantage over 1D notebooks. In this section, we provide a high-level summary of the common themes found in the study results.

5.1 Findings

5.1.1 Usability Benefits

2D notebooks appear to be more usable for certain types of data analysis tasks, especially those that require a significant amount of referencing cells one-to-one. Data analysts often want to look at two cells together to reuse code or compare visualizations. In a 1D notebook, cells can only be moved up or down one place at a time, which leads to a tedious process in rearranging the cells if they are far apart in the notebook. Additionally, this process disrupts the organization of the notebook, resulting in additional work to reorganize the notebook or leaving the notebook in a messy state. In the 2D environment, the multi-column environment enables users to place cells side-by-side while still keeping the notebook in an organized state. The ability to move columns left and right within the notebook can help maintain notebook

structure while still being able to compare cells from any point in the notebook. Additionally, the ability to drag and drop cells anywhere in the notebook provides an alternative method of placing cells close to each other. This freeform placement of cells also allows the user to easily reinsert the cell into the correct location in the notebook.

Additionally, in the 2D environment, users are able to organize their notebooks in such a way that it was easier for them to locate specific cells in their notebooks as compared to the 1D environment. 7 out of 9 participants used the multi-column layout to separate sections of their notebook, which allowed them to place each section side-by-side. These participants found that when they were seeking a specific part of their notebook, this layout allowed them to instantly identify the section they were looking for, reducing the amount of searching needed to find a specific cell. This was a much faster process as compared to the 1D environment, where users would have to scroll through a long page of cells in order to find the one they were looking for. While a 1D notebook can be sectioned using markdown cells, users would still need to scroll to look for the required section, then continue to scroll to search for the cell within that section.

5.1.2 Impacts of Screen Size

Many of the advantages of 2D Jupyter were more apparent when a larger display was used. The typical laptop screen size allows for about 3 columns to fit on the screen at a width that allows for comfortable viewing of code and visualizations. Out of 9 participants, 7 kept their notebooks limited to 4 columns or less. The 2 participants who used the larger 64-inch display to complete the data analysis task created the most columns, with 6 and 10 columns in their notebooks. This suggests that users may still feel limited by the physical space available and would prefer to limit the amount of scrolling required to navigate in the

notebook. Larger displays would allow the user to add additional columns while keeping the entire notebook in view. This would enhance many of the benefits of the 2D space, such as when making comparisons or locating certain cells.

Smaller screens can also lead to notebooks feeling cluttered. 2 participants who used a laptop display to conduct the data analysis task reported a sense of clutteredness in the 2D environment, especially when working with a larger notebook. In a 2D environment, users must scroll both horizontally and vertically to view the entire notebook. In order to minimize the amount of scrolling required, users may reduce column widths to fit more columns on the screen. However, this not only reduces readability of code and visualizations within the narrower columns, but also results in the notebook feeling cluttered, particularly when working with smaller displays. Users have a harder time reading the notebook as a single document, thus negatively impacting the ability to share or present the notebook.

5.1.3 Scrolling Tradeoffs

While the 2D environment helps to reduce the amount of *vertical* scrolling required, users now have an increased amount of *horizontal* scrolling. Users can create additional columns to reduce the vertical scrolling, but there is a tradeoff in that there will be an increase in the amount of horizontal scrolling as more columns are created. Additionally, the method of scrolling may hinder the ability to navigate through the notebook. Those who use a mouse to scroll horizontally may find it more tedious than those who are using a trackpad. Additionally, the drag-and-drop cell feature allows for the page to scroll when a cell is dragged to the edge of the screen; this method of scrolling is relatively slow and users may find it negatively impacts their experience of using 2D computational notebooks. However, it is worth noting that horizontal scrolling is not an exact equivalent to vertical scrolling and the

addition of horizontal scrolling still reduces the total amount of scrolling. Because multiple columns can fit on the screen at once, users are able to see more of their code at the same time. Additionally, the ability to use the 2D space for organization helps to condense the notebook in both the horizontal and vertical dimensions.

5.2 Limitations

At the time of conducting the study, the 2D Jupyter extension contained several bugs that may have impacted the user experience. In particular, the extension sometimes did not correctly save the layout of the notebook after each work session, requiring users to spend time reorganizing their notebook when they resumed work. Additionally, the drag and drop feature occasionally caused the cell to be stuck to the user's cursor and did not allow the user to release a cell at the intended location. This required the user to reload the notebook in order to resolve the issue. Finally, the extension interfered with the autosave function of Jupyter Notebooks, requiring the user to manually save the notebook periodically while working. This sometimes resulted in users losing work, especially if they had significant prior experience with Jupyter and were not in the habit of manually saving their notebooks.

5.3 Refinements

After conclusion of the study, refinements were made to the extension based on participant feedback and suggestions. First, we added the ability to move cells left and right. Two buttons were added to the main toolbar, as seen in Figure 5.1. In a similar manner to moving cells up and down, a user can select a cell and use the left and right buttons to move a cell left by one column or right by one column respectively. Cells are moved directly to

the left or right, or added to the end of the new column if the cell's position in the original column is lower than the last cell in the new column.

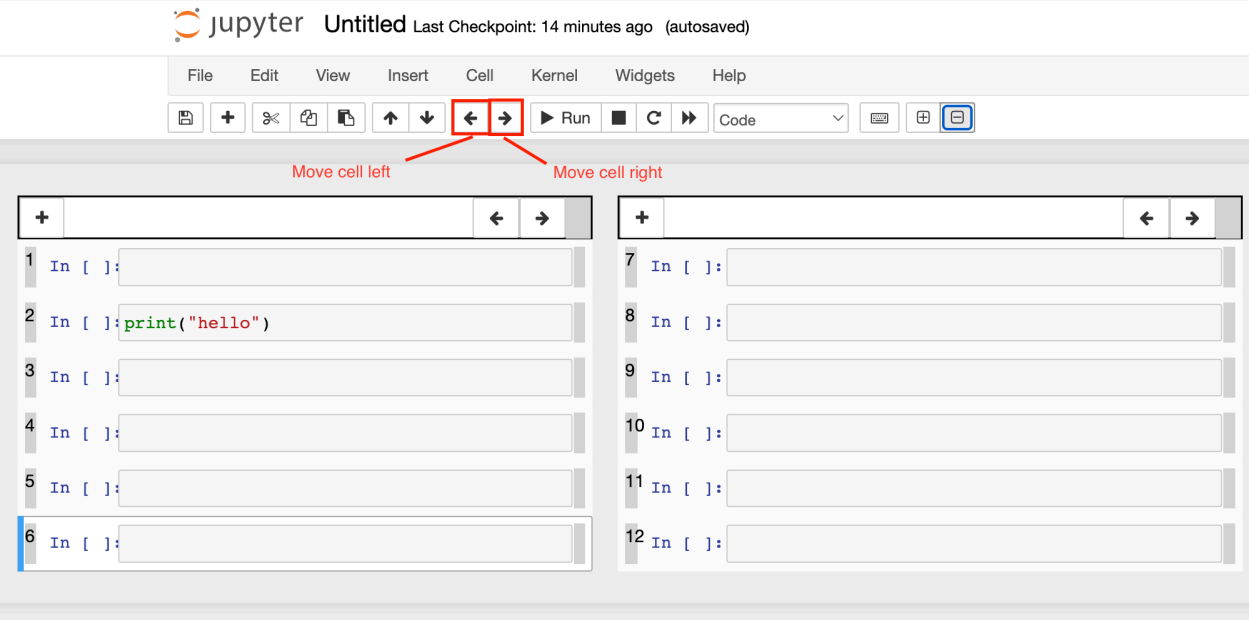


Figure 5.1: Buttons to move cells left or right

Second, the ability to resize columns was modified so that the user can resize columns from any point in the column, instead of being limited to the column toolbar. Like the original resize box in the column toolbar, users may click on the boxes at the right-end of each cell and drag their mouse left or right to resize the entire column. Figure 5.2 shows these boxes in the 2D Jupyter interface.

Additionally, the add cell button in the column toolbar was removed. During the study, users expressed a desire to be able to add cells to the columns without having to scroll up to the column toolbar. However, this ability already exists in the original add cell button in the main toolbar, as this button can add a new cell below any selected cell. Because the column toolbar was such a prominent feature in the 2D notebook, it is likely that users did not think to use the original features of Jupyter Notebooks, and so felt limited by the placement of the column toolbar. As such, the add cell button in the column toolbar has

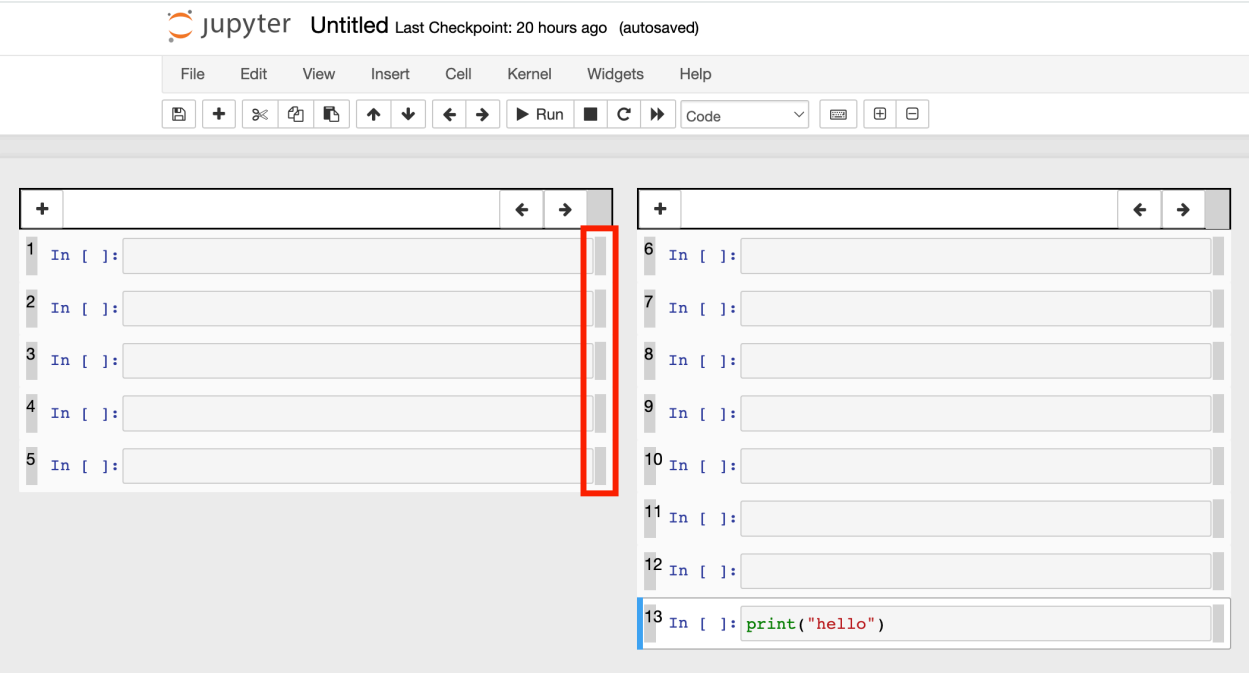


Figure 5.2: Resize boxes within cells

been removed to encourage use of the original add cell button.

Finally, some minor bug fixes were made in order to address the issue of the layout not being saved correctly. In the original implementation of the extension, certain edge cases in movement of cells and columns caused cell metadata to be incorrectly updated. These edge cases have been addressed in the updated implementation.

5.4 JupyterLab Extension

JupyterLab is a web-based development interface created by Project Jupyter that is based on the Jupyter Notebooks architecture [15]. It uses the same server and notebook format as the original Jupyter Notebooks, but also provides additional features to the development environment such as split-view windows, a visual debugger, and real-time collaboration. With these added capabilites, JupyterLab serves as a powerful and highly interactive IDE

for data analysis work, and Project Jupyter intends for JuypterLab to eventually replace the original Jupyter Notebooks. To this end, we have begun adapting 2D Jupyter to the JupyterLab environment.

Like the original extension, 2D JupyterLab enables multi-column layouts within the notebook environment. Figure 5.3 shows a screenshot of a JupyterLab notebook created with the extension enabled. Similarly to the Jupyter Notebook version, users are able to add columns, delete columns, and add cells using buttons on the main toolbar at the top of the interface. The column toolbar is also similar to the original extension, and allows users to add cells to a specific column with an "Add cell" button at the top left. Users also have the ability to move columns left and right using the arrow buttons in the column toolbar.

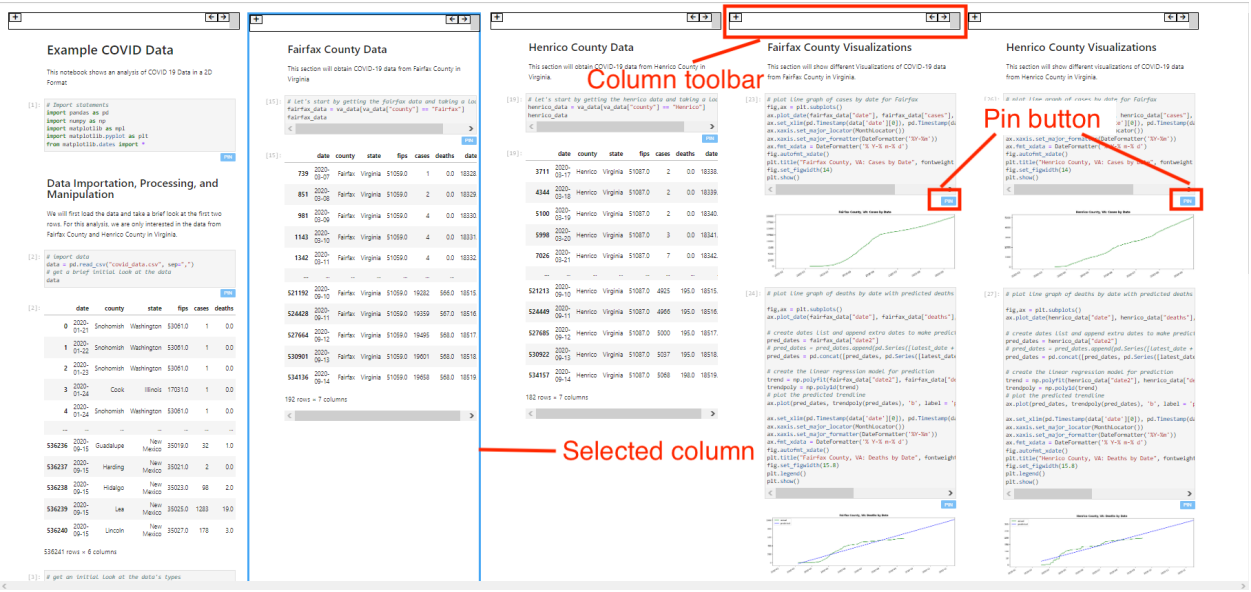


Figure 5.3: 2D JupyterLab features

The JupyterLab version of the extension is implemented using TypeScript. While many of the features have similar implementations to the original 2D Jupyter extension, several features have modified implementations due to the different interface and architecture of JupyterLab. In particular, the run order of cells in JupyterLab is contained within a backend

server instead of the HTML DOM, requiring calls to the JupyterLab API whenever the run order of cells is changed. Like the original 2D Jupyter extension, a reindex function is called each time the layout is modified in order to reset the cell indices and number of columns. However, we have additionally implemented a function named runCellsInOrder that first reorders the cells in the backend of JupyterLab, then calls the "Run All" function, in order to run the cells in order of appearance in the front end.

At the time of writing, development on 2D JupyterLab is ongoing. Future work includes debugging core functions such as runCellsInOrder, adding columns, and removing columns. We also plan on implementing many of the features present in the original extension, such as the ability to resize columns and the ability to save the 2D layout of the notebook.

5.5 Future Work

5.5.1 Further Development

The implementation of additional features in 2D Jupyter can further enhance the benefits of the 2D space. For example, the width of columns in the multi-column layout can impact user experience, especially those working on smaller displays. Making columns wider reduces the number of columns that fit on the screen, but making columns more narrow reduces readability and makes the notebook look cluttered. It may be beneficial to improve the method of resizing column width by using quick interactions such as in spreadsheets.

Additionally, there is room for improvement in navigational abilities. In 1D notebooks, some users find it quicker to navigate using the arrow keys. With the 2D Jupyter extension, users can use the arrow keys to move up and down in columns, but left and right navigation with the arrow keys has yet to be implemented. Additionally, making columns individually

scrollable can help navigation by allowing longer columns to be scrolled without impacting the viewing of shorter columns. This would be especially beneficial when working on smaller screens. Larger notebooks may also benefit from the implementation of a minimap feature or similar to aid in orienting users within the notebook.

It is also worth exploring implementation of additional ways to move cells around the notebook. The ability to move around groups of cells rather than individual cells could help enhance the benefits of the 2D environment. One possible approach is to allow columns to be dragged around the notebook in a similar manner as scratch cells. Often data analysts use multiple cells to arrive at a single result. Allowing freeform placement of all related cells at once would help in the non-linear analysis process.

Finally, the 2D layout allows for multiple potential run orders of cells. Currently, the extension runs each column from top to bottom, then runs the columns from left to right. Users may wish to run cells in an alternative order, such as in row-major order. Allowing users to select a run order through the use of a dropdown menu or other means would provide additional flexibility in the use of the 2D environment. This would require creating alternative styling layouts for each of the different run order in order to maintain the DOM structure, and thus is a subject for future work.

5.5.2 Enabling Other Layouts

The 2D Jupyter extension in its current state primarily supports multi-column layouts. However, the study done by Harden et al. [12] showed the potential for other layouts in 2D space. Many of the advantages of the extension may be a result of its compactness of the multi-column structure, such as the ability to more easily find code, and thus other 2D layouts may not see the same benefits. Additionally, the multi-column layout does not maximize the

potential benefits of having space to think, as users are still restricted to a column structure in the organization of their notebook. Other 2D layouts, like workboard style layouts, may introduce additional advantages. While the 2D Jupyter extension provides some ability to create this type of layout in the freeform cell placement feature, further development will be required to fully support workboard layouts. The implementation of alternative 2D layouts and the evaluation of their effectiveness is a subject for future work.

Chapter 6

Conclusion

In this paper, we present 2D Jupyter, a Jupyter Notebooks extension that provides the ability to use a 2D layout in computational notebooks. We first looked at the ways we might extend Jupyter Notebooks to use 2D space, as well as potential features that users might want. Through an exploration of the current pain points in computational notebooks, as well as the ideas supporting the concept of space to think, we designed 2D Jupyter with two primary goals: enabling a multi-column structure and allowing for freeform cell placement. We then conducted a user study with the extension to understand how people might use 2D space when conducting data analyses, as well as to find the advantages and disadvantages of a 2D layout compared to a 1D layout. We found that while users have different strategies in organizing their notebooks, they find the use of 2D space to be beneficial in almost all cases. Additionally, there were several advantages of 2D notebooks as compared to 1D notebooks, including the ability to more easily reference other cells and the ability to quickly locate specific code cells. The 2D environment also showed some disadvantages that were especially prevalent with smaller displays, such as the additional scrolling required with the addition of an extra dimension. Participants in the study also provided feedback and suggestions for future development of the extension. Overall, we found that the 2D layout provides benefits in usability, especially when used on larger display spaces. The multi-column structure served as a way to section notebooks in a way that reduced messiness and enabled more efficient navigation.

This work provides initial insights into the ways that 2D space can be used in the creation of computational notebooks. Further development of the extension can help enhance the benefits of 2D Jupyter. For example, additional navigational abilities can help to lessen the impacts of tedious scrolling or help orient users when working in a large notebook. Additional interaction features such as being able to drag and drop groups of cells, or quick interactions to resize columns can also make data analysis work more efficient. A possible future direction of work is to explore the impacts of other 2D layouts, such as workboard style layouts.

Bibliography

[1] CORGIS Datasets Project, 2021. URL https://corgis-edu.github.io/corgis/csv/covid/.

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

[3] Christopher Andrews, Alex Endert, and Chris North. Space to think: large high-resolution displays for sensemaking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 55–64, New York, NY, USA, April 2010. Association for Computing Machinery. ISBN 978-1-60558-929-9. doi: 10.1145/1753326.1753336. URL https://dl.acm.org/doi/10.1145/1753326.1753336.

[4] Ekaba Bisong. Google colaboratory. *Building machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners*, pages 59–64, 2019.

[5] Andrew Bragdon, Robert Zeleznik, Steven P. Reiss, Suman Karumuri, William Cheung, Joshua Kaplan, Christopher Coleman, Ferdi Adeputra, and Joseph J. LaViola. Code bubbles: a working set-based interface for code understanding and maintenance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2503–2512, Atlanta Georgia USA, April 2010. ACM. ISBN 978-1-60558-929-9. doi: 10.1145/1753326.1753706. URL https://dl.acm.org/doi/10.1145/1753326.1753706.

[6] US Census Bureau. County Population Totals and Components of Change: 2020-2022, 2022. URL https://www.census.gov/data/tables/time-series/demo/popest/2020s-counties-total.html. Section: Government.

[7] 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.

[8] 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*, CHI '20, pages 1–12, New York, NY, USA, April 2020. Association for Computing Machinery. ISBN 978-1-4503-6708-0. doi: 10.1145/3313831.3376729. URL https://dl.acm.org/doi/10.1145/3313831.3376729.

[9] 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*, PP, September 2022. ISSN 1941-0506. doi: 10.1109/TVCG.2022.3207357.

[10] Helen Dong, Shurui Zhou, Jin LC 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. IEEE, 2021.

[11] Google. Welcome to colaboratory, 2023. URL https://colab.research.google.com/.

[12] 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, September 2022. doi: 10.1109/VL/HCC53370.2022.9833128. ISSN: 1943-6106.

[13] 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, Glasgow Scotland Uk, May 2019. ACM. ISBN 978-1-4503-5970-2. doi: 10.1145/3290605.3300500. URL https://dl.acm.org/doi/10.1145/3290605.3300500.

[14] SageMath Inc. Cocalc, 2023. URL https://cocalc.com/.

[15] Project Jupyter. Project Jupyter | Home, 2021. URL https://jupyter.org/.

[16] 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. Publisher: IEEE.

[17] Mary Beth Kery and Brad A. Myers. Exploring exploratory programming. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 25–29, 2017. doi: 10.1109/VLHCC.2017.8103446.

[18] 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.

[19] Mary Beth Kery, Amber Horvath, and Brad A Myers. Variolite: Supporting Exploratory Programming by Data Scientists. In *CHI*, volume 10, pages 3025453–3025626, 2017.

[20] Nurit Kirshenbaum, Kylie Davidson, Jesse Harden, Chris North, Dylan Kobayashi, Ryan Theriot, Roderick S. Tabalba, Michael L. Rogers, Mahdi Belcaid, Andrew T. Burks, Krishna N. Bharadwaj, Luc Renambot, Andrew E. Johnson, Lance Long, and Jason Leigh. Traces of time through space: Advantages of creating complex canvases

in collaborative meetings. *Proc. ACM Hum.-Comput. Interact.*, 5(ISS), nov 2021. doi: 10.1145/3488552. URL https://doi.org/10.1145/3488552.

[21] 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.

[22] Donald Ervin Knuth. Literate programming. *The computer journal*, 27(2):97–111, 1984. Publisher: Oxford University Press.

[23] Lee Lisle, Xiaoyu Chen, J.K. 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, Atlanta, GA, USA, March 2020. IEEE. ISBN 978-1-72816-532-5. doi: 10.1109/VRW50115.2020.00073. URL https://ieeexplore.ieee.org/document/9090620/.

[24] 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, 2021. doi: 10.1109/VR50410.2021.00103.

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

[26] 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.

[27] Deepthi Raghunandan, Aayushi Roy, Shenzhi Shi, Niklas Elmqvist, and Leilani Battle.

Code code evolution: Understanding how people change data science notebooks over time. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2023.

[28] 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, February 2021. ISSN 1941-0506. doi: 10.1109/TVCG.2020.3030460. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[29] 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, Glasgow Scotland Uk, May 2019. ACM. ISBN 978-1-4503-5970-2. doi: 10.1145/3290605.3300272. URL https://dl.acm.org/doi/10.1145/3290605.3300272.

[30] 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, Montreal QC Canada, April 2018. ACM. ISBN 978-1-4503-5620-6. doi: 10.1145/3173574.3173606. URL https://dl.acm.org/doi/10.1145/3173574.3173606.

[31] Jupyter Team. Architecture, 2015. URL https://docs.jupyter.org/en/latest/projects/architecture/content-architecture.html.

[32] Jupyter Contrib Team. Unofficial jupyter notebook extensions, 2018. URL https://jupyter-contribnbextensions.readthedocs.io/en/latest/index.html#.

[33] 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.

[34] 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, Yokohama Japan, May 2021. ACM. ISBN 978-1-4503-8096-6. doi: 10.1145/3411764.3445527. URL https://dl.acm.org/doi/10.1145/3411764.3445527.

[35] 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, October 2019. doi: 10.1109/VDS48975.2019.8973385.

Appendices

Appendix A

Deliverables

2D Jupyter: https://github.com/elizabethc99/2D-Jupyter

2D JupyterLab version: https://github.com/TomWoah123/2D-JupyterLab-TS

Appendix B

User Study Materials

B.1 Original Task

You are being asked to analyze COVID data for the state of Virginia. Use the datasets provided to answer the questions below:

1. How do the most recent deaths per case compare between all the counties of Virginia?
2. Do the deaths per case in each county of Virginia correlate to the population density?

B.2 Modified Task

You are being asked to analyze COVID data for three states - Virginia, Texas and Illinois. Prepare a notebook for presentation with the following:

1. Create the following charts for each state:
    - Bar chart showing top 10 counties with highest cases
    - Bar chart showing top 10 counties with highest deaths
    - Bar chart showing top 10 counties with highest deaths per case
    - Scatterplot showing the correlation between cases and deaths by county (include the correlation coefficient)

- Bar chart showing top 10 counties with the highest number of cases relative to population
- Bar chart showing top 10 counties with the highest number of deaths relative to population.

2. Using only the charts you have created in part 1 answer the following questions:

   (a) Which state had the county with the highest number of cases?

   (b) Which state had the county with the highest number of deaths?

   (c) Which state had the county with the highest number of deaths per case?

   (d) Which state had the highest correlation between cases and deaths?

   (e) Which state had the county with the highest number of cases relative to population density?

   (f) Which state had the county with the highest number of deaths relative to population density?

   (g) How many counties with the top 10 deaths relative to population were also in the top 10 deaths per case for that state?

3. Prepare the notebook for presentation of your findings

B.3 Interview Questions

1. What was your overall strategy for using the 2D environment?

2. What features of the 2D notebook did you utilize?

3. Are there any features that you wish you had?

4. Were there any difficulties in using the 2D notebook during your data analysis?

5. Did the 2D environment provide any advantages for this task as compared to a 1D notebook?

6. Did the 2D environment provide any disadvantages for this task as compared to a 1D notebook?