OutlineSpark: Igniting AI-powered Presentation Slides Creation from Computational Notebooks through Outlines

Fengjie Wang
Sichuan University
Chengdu, China
The Hong Kong University of Science and Technology
Hong Kong SAR, China
wangfengjie@stu.scu.edu.cn

Yanna Lin
The Hong Kong University of Science and Technology
Hong Kong SAR, China
ylindg@connect.ust.hk

Leni Yang*
The Hong Kong University of Science and Technology
Hong Kong SAR, China
lyangbb@connect.ust.hk

Haotian Li
The Hong Kong University of Science and Technology
Hong Kong SAR, China
haotian.li@connect.ust.hk

Mingyang Gu
Sichuan University
Chengdu, China
gumingyang@stu.scu.edu.cn

Min Zhu*
Sichuan University
Chengdu, China
zhumin@scu.edu.cn

Huamin Qu
The Hong Kong University of Science and Technology
Hong Kong SAR, China
huamin@cse.ust.hk

ABSTRACT

Computational notebooks are widely utilized for exploration and analysis. However, creating slides to communicate analysis results from these notebooks is quite tedious and time-consuming. Researchers have proposed automatic systems for generating slides from notebooks, which, however, often do not consider the process of users conceiving and organizing their messages from massive code cells. Those systems ask users to go directly into the slide creation process, which causes potentially ill-structured slides and burdens in further refinement. Inspired by the common and widely recommended slide creation practice: drafting outlines first and then adding concrete content, we introduce OutlineSpark, an AI-powered slide creation tool that generates slides from a slide outline written by the user. The tool automatically retrieves relevant notebook cells based on the outlines and converts them into slide content. We evaluated OutlineSpark with 12 users. Both the quantitative and qualitative feedback from the participants verify its effectiveness and usability.

CCS CONCEPTS

• **Human-centered computing** → **Interactive systems and tools**; **Natural language interfaces**; • **Applied computing** → *Document preparation.*

*Correspondence authors.

CHI '24, May 11–16, 2024, Honolulu, HI, USA

KEYWORDS

computational notebooks, slides generation, outlines, data science

ACM Reference Format:
Fengjie Wang, Yanna Lin, Leni Yang, Haotian Li, Mingyang Gu, Min Zhu, and Huamin Qu. 2024. OutlineSpark: Igniting AI-powered Presentation Slides Creation from Computational Notebooks through Outlines. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24), May 11–16, 2024, Honolulu, HI, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3613904.3642865

1 INTRODUCTION

Computational notebooks like JupyterLab [18] and Jupyter Notebook [17] are widely popular for data exploration and analysis [26, 50]. With a block-based interface and dynamic code execution, computational notebooks enable an iterative analysis process in which users flexibly create cells to explore new analysis approaches and directions, inspect intermediate results, and make documentation [23, 28, 45]. However, such an analysis process often results in lengthy and poorly formatted notebooks [15, 50]. This brings challenges when users need to present critical analysis details and results to teammates in collaboration or report to stakeholders such as clients and decision-makers. Users have to utilize external presentation tools (e.g., Google Slides or Microsoft PowerPoint) to create slides for effective communication [8, 46]. They have to put lots of effort into 1) conceiving the structure and content of their presentations from messy notebooks [46, 66], 2) locating and retrieving relevant cells from notebooks to extract details [27, 50, 57], and 3) making slides that align with the intended presentation structure [6, 66].

Researchers have proposed automatic systems to alleviate this tedious and burdensome task. NB2Slides [66] automatically generates an entire slide deck based on a prescribed outline following the stages of data science and machine learning lifecycle. However, this approach is only applicable to those computational notebooks for building machine learning models. Furthermore, it requires the notebook to have a complete data science workflow and high-quality documentation, conditions that are seldom met in practice [50, 55]. Slide4N [57] addresses these issues by permitting users to select cells of interest to generate and refine slides. However, it assumes that the users have a clear image of the organization and content of their slides and it asks users to begin directly with the process of selecting cells. In this workflow, users may find it too demanding to hold a clear mental image of the structure of their slides to make the selection. It can also lead to ill-structured slides and bring additional burdens to refining the organization of the generated slides. To sum up, these tools do not consider the process of slides ideation—conceiving the structure and content of the slides. There is a need for additional support to facilitate the slide ideation process and to connect the outcomes of this ideation to the actual generation of slides.

To fill the gap, we present OutlineSpark, an interactive and intelligent tool that supports generating slides from computational notebooks based on the outlines written by users. Specifically, when using the tool, a user can look through the notebooks and write outlines to ideate the structure of the presentation. An outline can be a general title indicating the purpose or results of some cells such as "Data Cleaning", "Removing Outliers" and "Findings about Year Built vs Price". The outlines will be later used as the input of an automatic algorithm that can retrieve cells relevant to each item in the outline and generate the corresponding slide.

We decided upon the outline-driven workflow mainly for two reasons: 1) It is a common practice to craft outlines before creating slides, as evidenced by the prevalence of presentations starting with an agenda or outline slide [3, 44]. 2) It is also highly recommended since it aids in organizing what to present [1, 30, 48, 64]. It is worth noting that OutlineSpark allows users to manually select cells, and it will generate a corresponding item in the outline and the slide. Our goal is not to replace the previous workflow of selecting cells for slides creation, instead, we aim to complement the lack of consideration in the slides ideation process.

Centering around the outline-based workflow, OutlineSpark (Figure 1) has a set of functions and interface designs to facilitate slides ideation and generation. First, for the slides ideation, OutlineSpark provides *Notebook Overview* (Figure 1 (A)) that provides an overview of notebook cells with keywords for reminding users of the content of the notebook. It further recommends the next item in the outline by leveraging the current outline, the location where the recommendation was requested, and the analysis conducted within the notebook. Second, for the slides generation, the tool automatically retrieves relevant cells from the notebook based on the outlines and converts them into slide contents in *Slides Panel* (Figure 1 (C)). After the slides are generated, the users can refine their content and organization by modifying the outlines directly. The notebook and slides are fully linked through outlines, allowing for ideation, generation, and refinement of slides.

To evaluate the effectiveness and usability of OutlineSpark, we conducted a user study with 12 participants. Moreover, we attempt to understand users' preferences between generating slides through writing outlines, which we refer to as the *outline-based* approach, and through selecting cells, which we refer to as the *selection-based* approach. The feedback obtained from the questionnaires and interview demonstrates that OutlineSpark could help participants effectively create desired slides with less effort and is easy to use. We also observed that most participants showed a preference for outline-based slide generation, because it resonated with their regular practices, and allowed them to concentrate on crafting the narrative of their presentation rather than becoming overly fixated on individual slides. Overall, participants highly praised the process of creating slides simply by outlines. Finally, we concluded our research by discussing the lessons learned and potential future directions.

In summary, our contributions in this paper include:

- An outline-based workflow that streamlines the slides ideation and creation process from computational notebooks;
- A computational notebook plugin, OutlineSpark, that assists data scientists in creating presentation slides via outlines;
- A user study conducted to evaluate OutlineSpark and understand user preferences between outline-based and selection-based slides generation from computational notebooks.

2 RELATED WORK

2.1 Tools for Enhancing Computational Notebooks in Data Analysis

Computational notebooks have become the most popular programming environment for data scientist [23, 50]. With computational notebooks, data scientists can iterate through chunks of code, experiment with different methods, inspect intermediate results, and add documentation during exploration [15, 23, 28, 45].

While widely adopted, data scientists have encountered many problems in using computational notebooks, which has attracted a large number of researchers in the HCI field to develop tools for enhancing notebooks. These tools have covered almost every stage of data scientists' workflow, from code search [32, 33], to code management [15, 21, 22], data exploration [12, 29, 47], machine learning (ML) model development [4, 39, 41], and others [24, 59, 62]. In the domain of code search, for example, NBSearch [32] supports semantic search for relevant code cells from a large corpus of notebooks and designs novel visualizations to support interactive exploration of search results. For code management, Variolite [21] and Verdant [22] introduce features like rapid versioning and intuitive visual exploration of code history to assist in the effective management of evolving codes. In terms of exploration, examples include Lux [29] and Solas [12] that are designed to automatically recommend static visualizations for exploring data. As for ML model development, Pipeline Profiler [41] utilizes visual analytics to support the exploration and comparison of machine learning pipelines, so as to improve users' understanding of the algorithms.

While these works primarily focus on the technical tasks performed by data scientists, a limited body of research addresses the less technical but equally important communication tasks. Previous studies have emphasized that clear and effective communication

is essential for data science workers to align expectations, build trust, and share insights [30, 36, 38, 65]. However, there are numerous challenges, such as knowledge gaps, language barriers, and issues with trust-building, which complicate communication [19, 46]. These findings underscore the importance of improving communication within the data science workflow. Our study contributes to this emerging area of research by focusing on facilitating the creation of slide decks to effectively communicate analysis results from notebooks. We will delve into this topic further in subsection 2.2.

2.2 Communication Support for Computational Notebooks

With the growing complexity of data analysis work, multiple collaborators with different backgrounds are usually involved in the same analysis [19, 25, 65], and they need to frequently communicate with each other to move forward [9, 10, 27, 56]. However, computational notebooks, which represent the technical work of data scientists, are often lengthy, disorganized, and interspersed with interim notes [14, 15, 50, 54]. It's difficult for other collaborators to understand, which in turn hinders communication [15, 19, 46].

To address this critical problem, some researchers aim to curate the notebook before sharing. Adam et al. [49] proposed a technique that controls the visibility of cells using hierarchy to aid the high-level comprehension of notebooks. Code Gathering Tools [15] help find, clean, recover, and compare versions of code and generate more readable, cleaner notebooks. However, code is still the main content of notebooks, which is hard to understand [49, 57].

Another thread of research improves the readability of notebooks by facilitating the creation of documentation that provides explanations and findings in notebooks. For example, Themisto [55] applies deep-learning algorithms to generate several kinds of documentation (e.g., process and reference) for code in computational notebooks. However, it only provides a start of the sentence as a prompt and asks users to manually document findings. To alleviate the burden, InkSight [34] allows users to sketch on the charts they created for analysis to indicate data subsets of their interests and automatically generates the documentation of findings from data. However, it is not always suitable to use the notebook with documentation as the communication medium. In scenarios such as reporting to stakeholders, the content of notebooks should be filtered and reorganized to make slide decks for presentation, which demands much manual effort.

Recently, some works have attempted to close the gap between data analysis and presentation by supporting slide generation directly from computational notebooks. RISE [2] allows users to designate the role of each cell within a notebook, categorizing them as slides, sub-titles, elements to skip, and so on. Slides are then automatically generated based on these predefined roles. Although straightforward, this method requires a manual configuration process. In contrast, Notable [31] seeks to automate this process to some extent by offering an on-the-fly plugin. It auto-generates the chart findings and embellishments to help data analysts create slides during the analysis phase. Unlike Notable which supports slide creation when analyzing data, some research tries to support the slide generation after data analysis. For example, NB2Slides [66] employs a prescribed outline to distill notebook contents into templated slides. While it generates a slide deck with one click, the rigid outline constrains data scientists from freely telling their stories and makes it challenging to adapt to diverse scenarios. Furthermore, its demand for high-quality documentation is often impractical in real-world settings. Slide4N [57] mitigates these limitations by allowing users to select individual notebook cells for slide generation. However, this method demands users maintain a clear mental image of their presentation structure to make the selection, which can result in disorganized slides and additional effort in refining the organization. In summary, both NB2Slides and Slide4N overlook the slide ideation process, which is crucial for shaping the presentation structure and content. In line with the research of supporting slide generation after data analysis, OutlineSpark considers how to facilitate the ideation process and connect it to the creation of the slides. It facilitates slide creation by enabling users to craft customized outlines while automatically generating the corresponding slides.

3 DESIGN GOALS

Our tool is designed to support the slide ideation process in creating presentation slides for communicating essential analysis details and results from computational notebooks. Informed by relevant literature, we derive the following design goals.

G1: Support the slides creation process guided by outlines. Designing a presentation using outlines is a widely recognized good practice [1, 30, 48, 64]. They serve as a guide for slide creation and are commonly used, as evidenced by the prevalence of presentations starting with an agenda or outline slide [3, 44]. Thus, the tool should facilitate users in structuring outlines and the creation of the slides centered around the outlines.

G2: Retrieve relevant cells based on the outlines. Computational notebooks often suffer from disorderly cell execution and loose connections between cells [15, 53, 63]. However, creating a slide typically relies on a set of tightly interconnected cells. Following the outlines, data scientists need to identify these cells within the notebook, which is tedious and time-consuming. Thus, the tool should automatically retrieve relevant cells from the notebook by identifying the connections between cells and the outlines.

G3: Automate slides creation from relevant cells. Presentation slides typically feature human-readable, self-explanatory, and concise content, such as titles, bullet points, and charts/tables [52, 57, 64]. Cells relevant to the outlines serve as the foundational materials for creating such slides, typically, the users need to manually extract and summarize the key information from them [27, 50, 66], and arrange them on slides. Such a process is often burdensome and time-consuming. Therefore, the tool should facilitate this process with automation to simplify the creation of slides.

G4: Support easy refinement of generated slides. Allowing users to exert some control over the slide generation process can serve as a means of refining the generated slides to better meet individual preferences and requirements [16, 40, 66], and mitigate the limitations of automated methods [5, 30, 67]. Therefore, the tool should support the refinement of the generated slides with a suite of easy-to-use interactions, such as updating slides via outlines, adjusting cells used in slide generation, and modifying generated slides.

G5: Recommend outline candidates based on the notebook. Crafting an effective outline that covers the main sections or topics, is a fundamental step in creating organized and coherent slides [30, 48, 64]. It takes time for users to consume the content of the notebook cells and go back and forth between notebook cells and the outline to draft it clearly. Although there is no fully automatic algorithm to create an outline that aligns well with the intention of the users, partial automation is possible by recommending the next item in the outline. To increase the efficiency of drafting outlines, we propose that the tool can recommend items of the outline according to previous items that are written by the users and the analysis in the notebook.

G6: Assist with quick recall of the notebook. Conceiving the structure of the slides requires users to first identify crucial information from the notebook. However, computational notebooks often suffer from lengthiness, poor formatting, and a focus on code [14, 15, 50, 54]. Readers, including oneself in the future, usually lack the interest to thoroughly read such documents to grasp the notebook's essence at a high level [49]. Thus, the tool should provide an overview of the notebook appropriately to enable a quick recall of the entire analysis.

4 OUTLINESPARK

In this section, we first present an overview of OutlineSpark (subsection 4.1). Then we introduce the interactive modules and computational modules of OutlineSpark (subsection 4.2 and subsection 4.3, respectively).

4.1 System Overview

OutlineSpark is developed based on the aforementioned design goals. It aims at assisting users in structuring and creating presentation slides from computational notebooks based on the outlines written by the users. The system architecture consists of two components: the interactive modules and the computational modules. The interactive modules decide the user interface and interaction designs, while the computational modules support the functioning of interactive modules in the back-end.

As shown in Figure 1, OutlineSpark can be positioned side by side with the notebook window, which is implemented as a JupyterLab extension. There are three interactive modules: (a) *Outline Panel* (Figure 1 (B)), which allows users to craft outlines (G1, G5) and trigger slides generation (G2, G3); (b) *Notebook Overview* (Figure 1 (A)), which provides a visual summary of the code and Markdown cells in the notebook by keywords (G6). This component also enables users to exclude or include a cell in a slide by selection (G4); and (c) *Slides Panel* (Figure 1 (C)), which renders the generated slides and allows users to refine and customize them (G3, G4).

To support the interactive modules of OutlineSpark, we have designed four computational modules: (a) *Keyword Extraction*, responsible for extracting keywords from notebook cells which are then displayed in *Notebook Overview* to assist in summarizing the notebook's content (G6); (b) *Topic Recommendation*, which extracts topics from the notebook content, and based on these extracted topics and the current outline, recommends the next outline item for users in *Outline Panel* (G5); (c) *Cell Retrieval*, which retrieves relevant cells for each item in an outline (G2); and (d) *Slide Generation*, designed for extracting essential information from notebook cells to generate slide titles, bullet points, charts, and tables on slides (G3). In the following sections, we introduce these modules in detail.

4.2 Interactive Modules

This section presents three interactive modules in the front-end user interface to support users in creating presentation slides from computational notebooks via outlines (G1).

Notebook Overview. As shown in Figure 2 (A), *Notebook Overview* concisely displays the notebook cells by keywords for a quick overview to assist users in drafting outlines (G6). Each notebook cell is represented by keywords displayed on a card with a pink left border. The height of a card increases with the amount of content in the cell, and the cards are arranged from top to bottom in accordance with the order of cells in the notebook. This card-based visual design aims to maintain consistency with the cell-based design in JupyterLab throughout the interface. Inspired by Slide4N [57] and NB2Slides [66], cells are used to generate slides in OutlineSpark. A dot is positioned on the left side of each card, with its color to indicate three possible states of a cell, namely, default (gray), focused (blue, indicating the mouse hovering over a card), or selected (pink, indicating a card chosen for slide generation). The pink bar on top of the card is designed to help users check the automatically retrieved cells by *Cell Retrieval*, which reflects the relevance of the cell to the selected outline item in *Outline Panel* (Figure 2 (B), when an outline item is clicked or edited by the users), the more relevant the wider the bar. The users can double-click on a card to bind or unbind the cell to the selected outline item, which allows them to easily refine the cells used to generate slides (G4). Several interactions are provided to facilitate seamless navigation between the notebook and OutlineSpark. When the users click on a card, the notebook window jumps to the corresponding cell and vice versa. Furthermore, hovering over a card triggers a tooltip that provides a quick overview of the cell's content, including code snippets and any associated tables or charts.

Outline Panel. After a quick overview of the notebook cells through *Notebook Overview*, users can move to *Outline Panel* to draft the outline. We design *Outline Panel* (Figure 1 (B)) to facilitate users' planning and creating presentation slides using explicit outline (G1). OutlineSpark supports writing a hierarchical outline including topics and sub-topics which are differentiated by font size, indentation, and a vertical gray bar. For example, sub-topic outline items are represented with a smaller font size and indented with a vertical bar on the left. Users can add an outline item by either pressing "Enter" on the keyboard or clicking the button on the right of each item (Figure 2 (b6)). Additionally, they can easily adjust the level and order of outline items by clicking the button or drag and drop interactions on the right of each item (Figure 2 (b7)). To get topic recommendations, users can press the space bar. When doing so, a list of recommended topics will appear below the selected outline item for users to choose from. After completing the outline, they can click the button at the top of *Outline Panel* (Figure 2 (b1)) to prompt OutlineSpark to generate a slide deck based on the outline. If the users are not satisfied with the current outline, they can further edit them. In addition to generating slides via outlines, OutlineSpark also allows users to select cells of interest

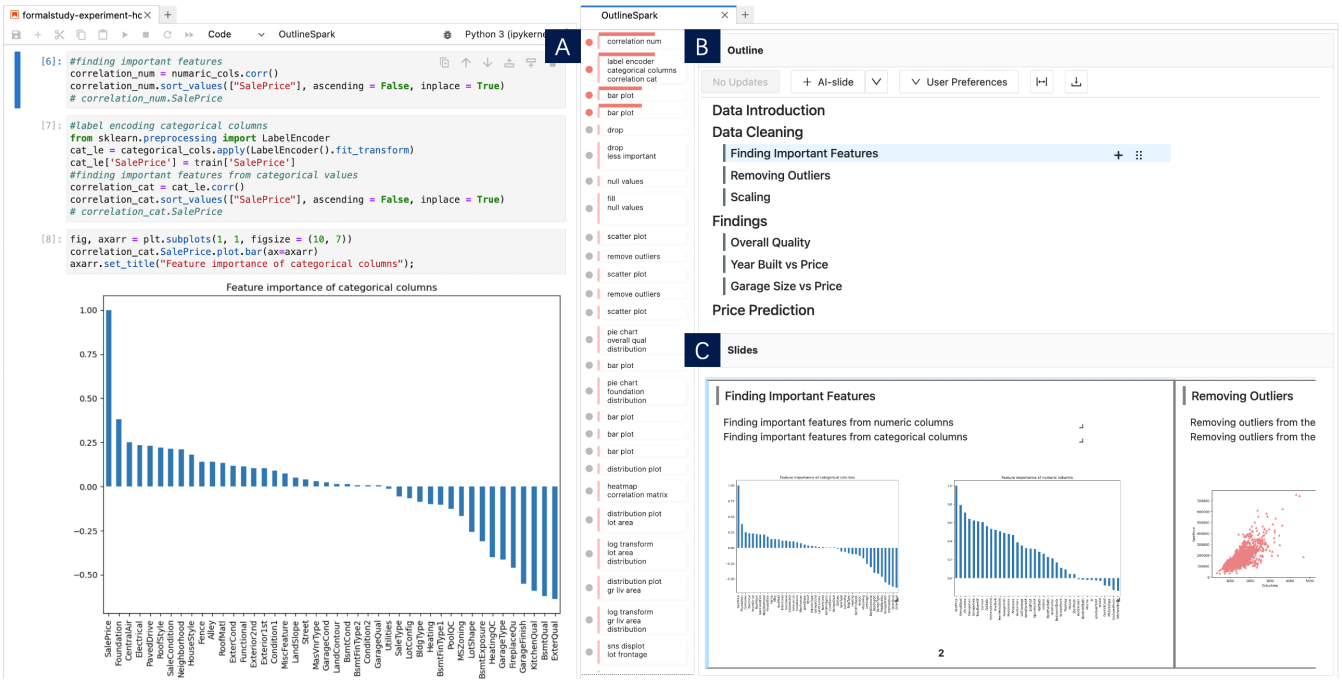

Figure 1: OutlineSpark is an interactive AI-powered JupyterLab plugin designed to facilitate the creation of presentation slides from computational notebooks using outlines. The user interface of OutlineSpark comprises three interconnected components: a *Notebook Overview* (A) that provides a visual summary of the notebook cells by relevant keywords, a *Outline Panel* (B) that enables users to craft outlines and initiate the generation of slides, and a *Slides Panel* (C) that renders the generated slides and offers users the flexibility to refine and customize them.

for slide generation, by double-clicking on the cards in *Notebook Overview* (Figure 2 (A)).

At the top toolbar of *Outline Panel*, users can change the slide generation options and parameters (G4). Specifically, they can manually create some slides by clicking "+AI-slide" (Figure 2 (b2)), adjust the number of cells retrieved (i.e., top-K relevant cells) for generating a slide (Figure 2 (b3)), control the level of detail in the generated bullet points (Figure 2 (b3)), include page numbers on slides (Figure 2 (b3)), stretch slides for presentation purposes (Figure 2 (b4)), and download the current slides as .pptx for further customization (Figure 2 (b5)).

Slides Panel. After generating the slides, users can proceed to make further refinements on *Slides Panel* (G4). *Slides Panel* (Figure 1 (C)) displays the generated slides in a left-to-right manner and offers a range of accessible interactions to empower users in refining and customizing the slides to meet their preferences. Inside each slide, users can edit any bullet point using markdown-based grammar. They can insert, resize, or remove plots and tables. Furthermore, they can adjust the layout by dragging and dropping slide contents (e.g., bullet points and charts). OutlineSpark further provides different layout templates, such as title slides and slides with one or two-column content. Users can click the "down arrow" on the *Outline Panel* (Figure 2 (b2)) to select a template. In terms of the general structure, users can manage slide order, and delete or restore slides (Figure 2 (c1)). Any changes on the slides that affect the outline, e.g., modifications to slide titles, addition, deletion, or reordering of slides, are seamlessly synchronized with the *Outline Panel* (Figure 2 (B)).

OutlineSpark provides interactive linking between all the interactive modules, allowing users to easily trace back to the cells used to generate the slides. When the user clicks on an outline item in *Outline Panel* (Figure 1 (B)), *Slides Panel* (Figure 1 (C)) automatically scrolls to the corresponding slide; *Notebook Overview* (Figure 1 (A)) highlights the retrieved cells (horizontal bar on top of a card) and selected cells (pink dots on the left side of the card), and scrolls to the first selected cell. Notably, when clicking on a slide in *Slides Panel*, *Outline Panel* and *Notebook Overview* respond in a similar manner. To help the users track their modifications, OutlineSpark highlights adjusted outline items (e.g., modification, addition, deletion, reordering, or binding new cells) in pink. For more advanced slide editing and beautification, users can export the slides in the .pptx format to modify them in Microsoft PowerPoint by clicking the download button (Figure 2 (b5)).

4.3 Computational Modules

This section describes the four computation modules that support the interactive modules. Considering the remarkable capabilities of LLMs (e.g., GPT-3.5 [42]) in understanding and generating both natural language and code [20, 60, 61], we leverage LLM (i.e., gpt-3.5-turbo-16k with temperature set to 0 for more consistent outputs) to assist users in creating the outline and converting them into slides. It is worth noting that (1) OutlineSpark's capacity for

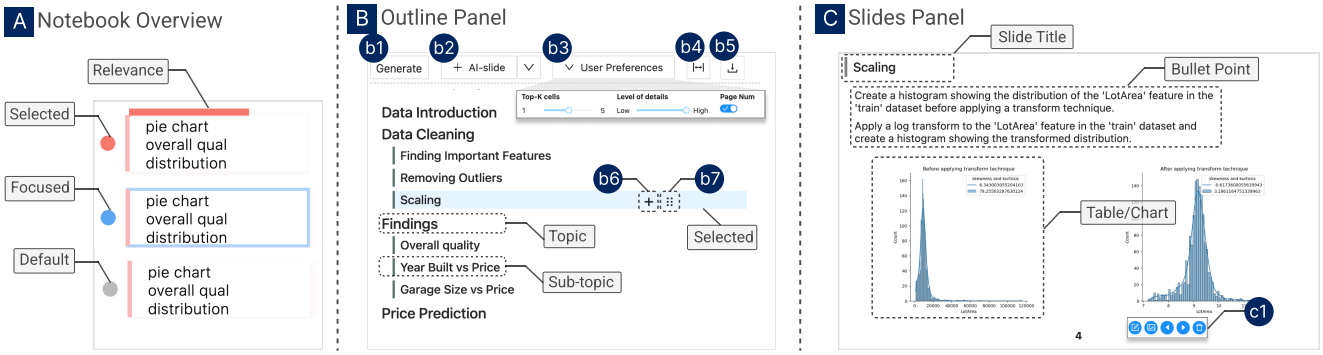

Figure 2: This figure illustrates the interactive components of OutlineSpark, including (A) *Notebook Overview*, (B) *Outline Panel*, and (C) *Slides Panel*. (A) explains the visual encodings of *Notebook Overview*. (b1)-(b7) allows users to generate slides, adjust slide generation options and parameters, edit outlines, and etc. (C) showcases the content of a slide, and (c1) enables users to refine the generated slides.

notebook length is limited by the employed LLM. According to the common length of data science notebooks [50], we opt for 16k tokens to accommodate these notebooks within the context length limits of the LLM; (2) All the prompts in computational modules are designed based on existing successful prompt engineering experiences [35, 43]. Further details about the prompts can be found in the supplementary material.

Keyword Extraction. Users often lack interest in delving into complex code when attempting to understand the notebook at a high-level [49]. To reduce the effort, OutlineSpark provides all the notebook cells to the LLM and instructs it to summarize the input (i.e., the code) of every notebook cell into at most 5 keywords and ranks them by how representative they are of the cell's content. The keywords are then rendered explicitly in *Notebook Overview* (Figure 2 (A)), allowing users to quickly grasp the complex code by keywords (G6).

Topic Recommendation. To ease the burden of drafting outlines, we instruct the LLM to generate contextually appropriate topic recommendations for users (G5). Before recommendation, OutlineSpark provides the notebook cells to LLM and prompts it to extract topics and relevant sub-topics from these cells as the candidate topics set. To ensure a user-friendly experience, OutlineSpark adopts a request-by-user design for topic recommendations, avoiding unnecessary interruptions. When requesting, OutlineSpark instructs the LLM to pick the top 10 relevant topics from the candidate topics set based on the current context of the current outline. The context depends on where the users make the request. As shown in Figure 2 (B), if requested at the sub-topic level, OutlineSpark treats the outline items of the parent topic (including itself) as context; if requested at the topic level, OutlineSpark treats all the current topic level outline items as the context.

Cell Retrieval. Finding relevant cells within a cluttered notebook can be burdensome for users, to simplify this process, OutlineSpark automatically maps notebook cells to user-created outline items (specifically, those items at the lowest level of the hierarchical outline). This involves transforming the hierarchical structure of outlines into flat outline units, each comprising two components: the outline item and its corresponding context. The context indicates the higher-level topic to which the outline item belongs, providing valuable contextual information for precise mapping. Subsequently, OutlineSpark provides the LLM with all outline units and notebook cells to ensure it makes informed mapping decisions. OutlineSpark then instructs the LLM to map each outline unit to a maximum of 5 notebook cells, taking into account the semantic relevance between the outline unit and the input of notebook cells. The retrieved cells are also assigned semantic relevance scores ranging from 0 to 1. To inform users of the mappings, the retrieved cells and their relevance scores are then visualized in *Notebook Overview* (Figure 2 (A) Relevance).

Slide Generation. Creating slides from notebook cells requires considerable effort, OutlineSpark automates the process to ease the burden (G3). Inspired by Slide4N [57], OutlineSpark tasks the LLM to convert every cell into one bullet point to summarize the content of the cell into a more human-readable and concise sentence. However, users may have varied preferences for the complexity of the generated bullet points. OutlineSpark provides two sliding bar widgets in *Outline Panel* for adjusting the generation rule. As shown in Figure 2 (b3), the left sliding bar is for selecting the number of retrieved cells to be included in a slide; the right one enables users to control the detail level of the generated bullet points. In addition to the generated bullet points, OutlineSpark synchronizes the outline into slide titles, and incorporates outputs (e.g., charts and tables, if available) from selected cells onto the slide. To provide a meaningful layout, OutlineSpark draws inspiration from previous work [58] and adopts the commonly used Parallel layout, which vertically separates each slide into three parts as shown in Figure 2 (C). At the top, the slide title is placed. Below the title, bullet points are arranged in descending order of relevance scores to the outline item. Below the bullet points, there are tables and charts, whose sizes are dynamically adjusted to avoid occlusion. OutlineSpark further arranges them from left to right.

5 USAGE SCENARIO

In this section, we present a scenario to illustrate how OutlineSpark supports structuring and creating slides from computational notebooks. Suppose Andie is a data analyst working at a real estate company. His manager tasks him with analyzing a house dataset

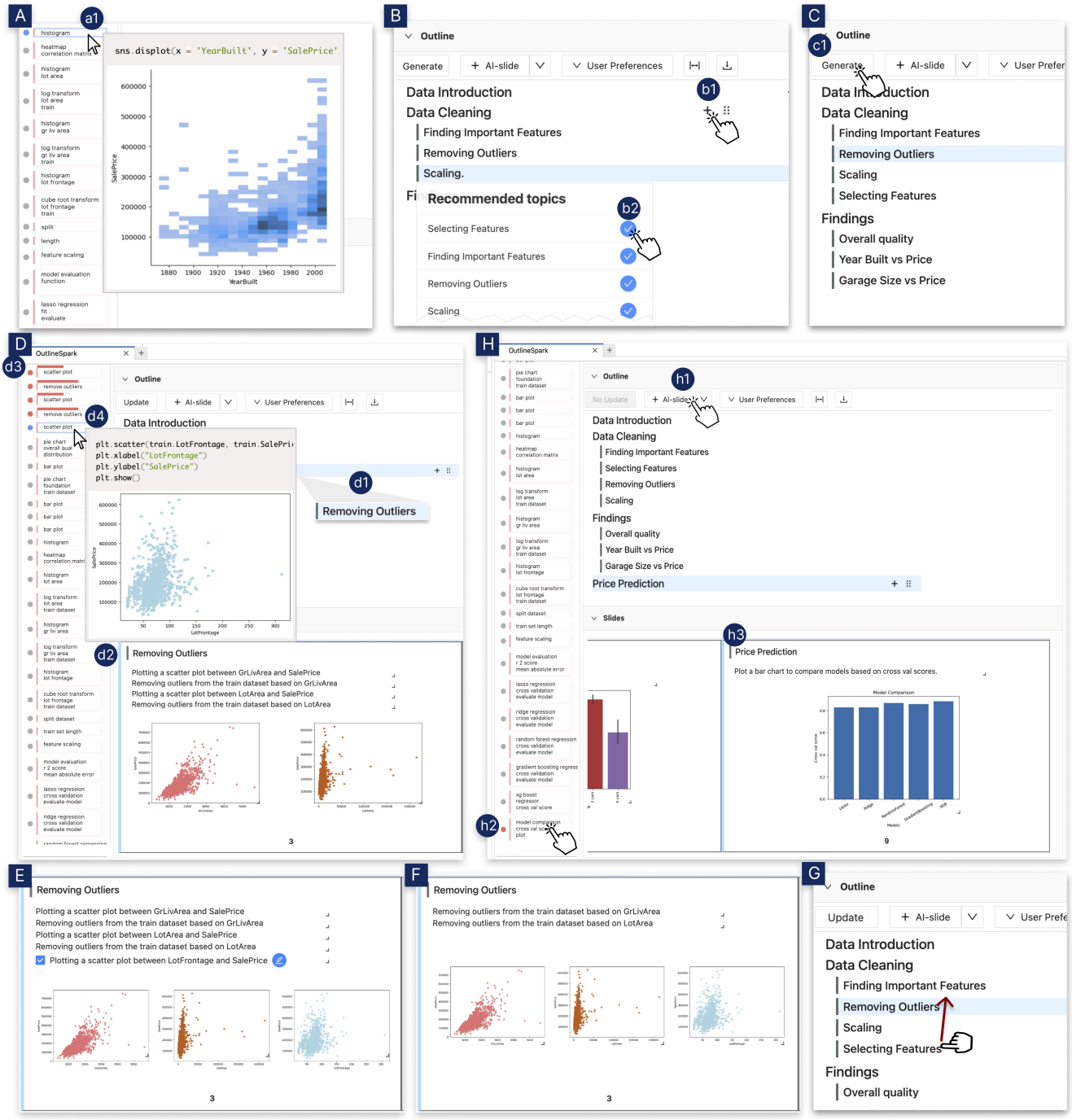

Figure 3: This figure illustrates a usage scenario in which an analyst, Andie, utilizes OutlineSpark to create slides from his computation notebook. In the process, Andie follows these steps: (A) Gain an overview of the notebooks, (B) Draft an outline to guide the slide generation, (C) Instruct OutlineSpark to generate the slides, and (D-H) Refine the generated slides.

in order to predict house prices. After completing the analysis using Jupyter notebook, Andie wants to create slides to report the analysis results to the manager.

Andie opens the notebook in JupyterLab and activates OutlineSpark from the toolbar. Then, he puts the two windows side by side to start to create slides based on his notebook. Next, he has a quick recall of what has been analyzed by browsing the keywords summarized from notebook cells in *Notebook Overview* (Figure 3 (A)). When Andie hovers over the cards in *Notebook Overview* (Figure 3 (a1)), each representing a cell in the notebook, a tooltip appears,

providing additional details about the corresponding cell, such as code snippets and the cell's output (e.g., tables and charts).

Then, Andie proceeds to draft an outline using *Outline Panel* (Figure 3 (B)). The tool distinguishes different levels of outline through variations in font size and indentation. He first drafts an outline with three topics: "Data Introduction", "Data Cleaning", and "Findings". Then he plans to illustrate the topic "Data Cleaning" with more sub-topics. By clicking the "+" button (Figure 3 (b1)), he manages to add sub-topics "Finding Important Features", "Removing Outliers", and "Scaling" under the topic. After that, he would like to check if he missed some critical content. By focusing on the last sub-topic and pressing the space bar, he gets some recommended topics from OutlineSpark and finally selects "Selecting Features" which he thinks complements the current content (Figure 3 (b2)). Following the above practice, he specifies the sub-topics for each topic when it is necessary. Gradually, an outline forms for generating the slides (Figure 3 (C)). Next, he clicks the "Generate" button (Figure 3 (c1)). For each topic and sub-topic at the lowest level of the outline, OutlineSpark generates a corresponding slide. OutlineSpark selects the top-K (a parameter that can be adjusted by the user) cells relevant to each sub-topic and renders a slide deck in *Slides Panel* (Figure 3 (d2)). After browsing each slide, he finds most slides present the information he intends to convey.

However, Andie notices that the slide corresponding to the topic "Removing Outliers" lacks some related charts (Figure 3 (d2)). He vaguely remembers noticing three relevant charts while scanning through the notebook in the beginning. To check with this, Andie clicks on the slide in *Slides Panel*. Upon the action, *Slides Panel* highlights the card that represents the slide with a blue left border (Figure 3 (d2)), while *Outline Panel* highlights the corresponding outline item with a blue background (Figure 3 (d1)). Simultaneously, *Notebook Overview* automatically scrolls to the first cell used to generate this slide (Figure 3 (d3)). Additionally, in *Notebook Overview*, the cards that represent the cells selected for slide generation are highlighted by pink dots at the left and pink bars at the top. The width of a pink bar indicates the relevance of a cell to the outline item. By examining these cells and several cells nearby them, Andie identifies a cell that displays a "scatter plot". When hovering, a tooltip appears (Figure 3 (d4)), providing additional details about the corresponding cell. After reviewing the cell, Andie selects it and clicks on "Update". Within moments, as shown in Figure 3 (E), the content of the slide is updated with a new bullet point, "Plotting a scatter plot between LotFrontage and SalePrice," and the scatter plot from the notebook cell. However, Andie doesn't want to include several bullet points, so he deletes them (Figure 3 (F)).

Next, Andie decides to have a final review of the structure of his presentation. He finds that placing "Selecting Features" immediately after "Finding Important Features" would be more appropriate. Thus, he adjusts their orders by the dragging and dropping interaction in *Outline Panel* (Figure 3 (G)). Andie then clicks on the "Update" button, prompting OutlineSpark to re-order the slides to align with the revised outline. Furthermore, he notices that he has forgotten to discuss the performance of the models in his outline. As he already knows where the relevant cells are, he directly clicks the "+ AI-slide" button on *Outline Panel* (Figure 3 (h1)) and scrolls in *Notebook Overview*. With a simple click, OutlineSpark automatically scrolls down the notebook window to the actual cell, enabling Andie to view the detailed code. He then selects the cell (Figure 3 (h2)) and clicks the "Generate" button. Within a second, a slide with a title, a bullet point, and a chart is rendered on *Slides Panel* (Figure 3 (h3)). However, the title doesn't fit his style, he then renamed it to "Price Prediction" in *Slides Panel*. Now he is satisfied with the structure and general content of the slides. Finally, he makes slight modifications to improve the wording and style of the slides.

6 USER STUDY

To evaluate the usability and effectiveness of OutlineSpark, we conducted a user study with 12 participants. We do not compare our system with a baseline. That's because the most comparable system is Slide4N [57] which generates slides based on cells selected by the users. However, as we have discussed, our work is not to replace such an interaction but to complement it with the outline-based approach to facilitate slides ideation and streamline the slides ideation and creation process. Thus, the evaluation of our tool will be primarily reflected by the feedback from participants. Furthermore, we attempt to understand participants' preferences between the two different interaction approaches, as well as whether different situations would affect their preferences.

6.1 Participants

We recruited 12 participants (5 females, 7 males; aged 24.7 ± 2.6) through social media and word of mouth (denoted as P1-P12). They are postgraduate researchers from diverse backgrounds, including human-computer interaction, visualization, recommendation systems, and ocean engineering. Participants self-reported their familiarity with Jupyter Notebook with a rating of 5.08 ± 1.38, where 1 represents "No Experience" and 7 represents "Expert". Moreover, they were familiar with creating slides for presentation, with frequency of 1 to 5 times per week.

6.2 Task

In our user study, inspired by the task in Slide4N [57] and NB2Slides [66], participants were required to create a slide deck for a 10-minute presentation. They were told that the target audiences could be business or technical audiences and the presentation content could consist of anything within the notebook that participants deemed important for presentation. To simulate the realistic scenario wherein participants would create slides following data analysis, they were tasked with familiarizing themselves with the provided notebook first before proceeding to slide creation. The slide deck was required to contain between 5 and 10 slides, both to gauge participants' proficiency in using OutlineSpark and to manage the time constraints of the study. To better understand users' preferences, during slide creation, participants were required to adopt two types of interactions for guiding slide generation, i.e., creating outlines and selecting cells.

6.3 Data

We selected the House Prices Prediction[1] notebook from Kaggle for the experiment, which is commonly employed to assess the effectiveness and usability of tools [34, 57, 66]. We removed all

[1]https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques

markdown cells to avoid the participants being affected by these cells and following the structure of those cells to organize the presentations. Furthermore, prior research indicates that most data analysts typically avoid documentation due to its time-consuming nature and potential to disrupt analytical flow [34, 55]. The resulting notebook comprised 42 code cells.

6.4 Procedure

All studies were conducted through one-to-one in-person meetings, each lasting about one hour. Prior to the user study, we briefly introduced the study procedure and gained consent from participants for video recording the whole process. The user study was segmented into four distinct phases: a training session, an experiment session, a post-study questionnaire session, and a post-study semi-structured interview. During the training session, we first briefly introduced the components of OutlineSpark and its related interactions. Participants were then required to interact with OutlineSpark to create a slide deck for a sample notebook for around 15 minutes, or until they felt familiar with it. The sample notebook is about Titanic data with 11 cells. The subsequent experiment session lasted around 25 minutes, where participants were given the experiment notebook and asked to finish a slide deck with 5-10 slides using OutlineSpark. This session concluded only when participants indicated satisfaction with their created slides, followed by a brief presentation of the slides. They then proceeded to complete two post-study questionnaires. The first was the System Usability Scale (SUS) [7], a widely recognized method for assessing tool usability, as shown in Figure 6. The second was a 7-point Likert scale questionnaire aimed at evaluating the effectiveness of OutlineSpark, where 1 signifies "strongly disagree" and 7 denotes "strongly agree", as shown in Figure 5. The study ended with a semi-structured interview, focusing on the advantages and disadvantages of the OutlineSpark as well as a discussion on two interactions of creating slides, i.e., creating outlines and selecting cells (detailed questions can be found in the supplementary material). Each participant received a compensation of $13.50 for completing the user study.

7 USER STUDY RESULTS

In this section, we first report participants' questionnaire responses regarding the effectiveness and usability of OutlineSpark. Then, we discuss preferences between outline-based and selection-based slide generation. The last part is about participants' qualitative feedback on OutlineSpark. As shown in Figure 4, we also present some outlines created by the user study participants.

7.1 Questionnaire Results

The quantitative results of our user study reflect participants' ratings on both the effectiveness and usability of OutlineSpark.

Regarding effectiveness, Figure 5 depicts the distributions of ratings, as well as the medians (MDs) and interquartile ranges (IQRs) for all participants (detailed ratings can be found in the supplementary material). Overall, most of the participants expressed satisfaction with OutlineSpark. Specifically, 11 out of 12 participants highly appreciated the outline-centered design that supported ideation and creation of presentation slides from computational notebooks (Q1), with a median rating of 6.5 ($IQR = 1$). They agreed that the

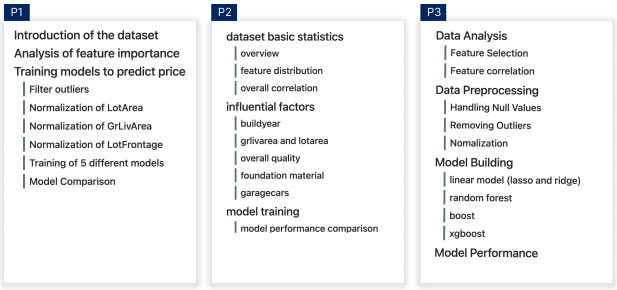

Figure 4: This figure showcases three sample outlines crafted by participants (P1, P2 and P3) during the user study.

essential interface designs and functions: (1) the outline panel (Q3), (2) the notebook cell retrieval (Q5), (3) the slides generation (Q6, Q7), and (4) the overview of cells (Q2) were useful and met their expectations. However, the rating for *Topic Recommendation* (Q4) was relatively lower as most participants gave a neutral score and one participant (P10) gave a negative score of 2. We will elaborate on the reasons in subsection 7.3 Topic Recommendation. Furthermore, we noticed that most of the participants weakly agreed that OutlineSpark supported sufficient customization of the generated slides (Q8) as OutlineSpark can not support rich editing like mature commercial software (e.g., Microsoft PowerPoint) does.

In terms of usability, OutlineSpark achieved a score of 85.2, surpassing that of 95% of applications according to Sauro and Lewis [51]. Notably, during the interview, all participants highly appreciated the seamless integration between the notebook window and the tool, as well as the linking among the three interactive modules. As P3 mentioned, *"The linking between the slide, the outline, and the notebook cells [in Notebook Overview] makes the refinement of the generated slides much easier."*

7.2 Preferences between Outline-based and Selection-based Slide Generation

In this section, we first present our findings by analyzing the slide creation process in the user study. Then, we present feedback from participants in the interview, regarding the outline-based and selection-based approaches for slide generation.

7.2.1 Characteristics of the Slide Creation Process. Our analysis of the recorded videos revealed 6 main activities participants engaged with during the slide creation process. These activities encompassed outline-based slide initiation, outline-based slide refinement, selection-based slide initiation, selection-based slide refinement, manual slide refinement (i.e., editing text and figures on slides), and other (e.g., looking through the user interface). We manually labeled the 12 recorded videos based on these activities and their duration, and the results were visualized in Figure 7 (more details can be found in the supplementary material). We initially present an overall summary of our findings before delving into the temporal sequencing.

As shown in Figure 7 (A), participants averagely allocated 42.2% and 23.8% of their time to outline-based and selection-based slide initiation and refinement, respectively. These results underscored the necessity of combining both interaction methods for effective

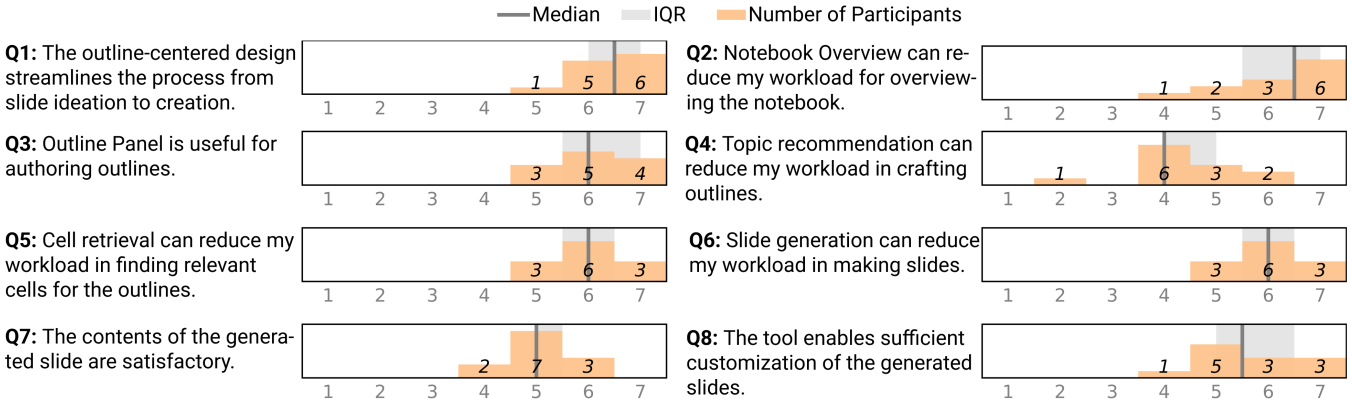

Figure 5: This figure shows participants' ratings on the effectiveness of OutlineSpark on a 7-point Likert Scale (1 = "strongly disagree" and 7 = "strongly agree").

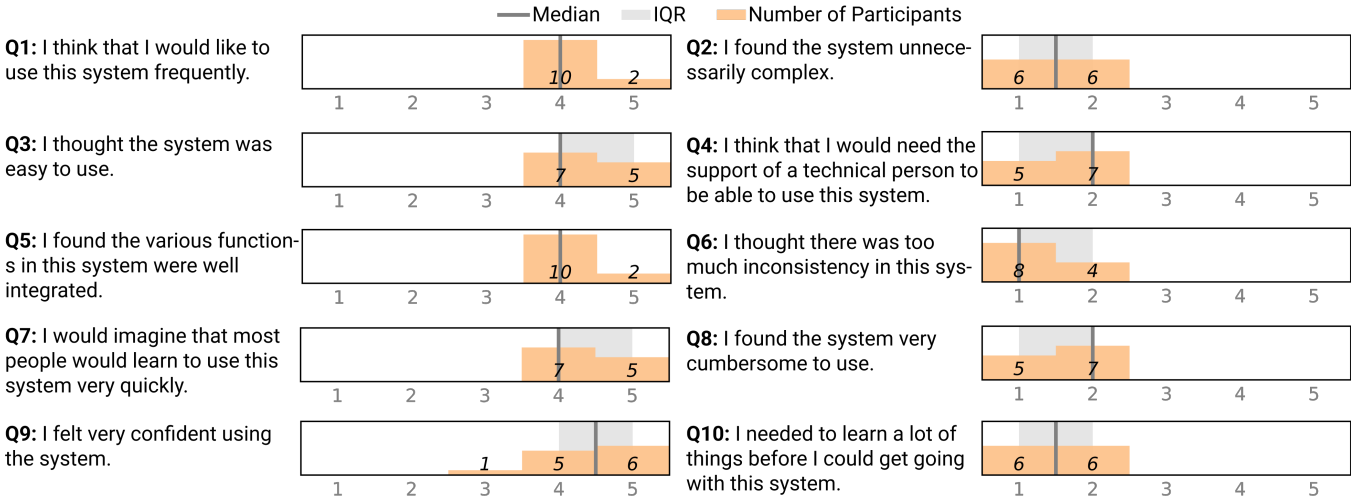

Figure 6: This figure presents participants' ratings on the usability of OutlineSpark on a 5-point Likert Scale (1 = "strongly disagree" and 5 = "strongly agree").

slide creation. Specifically, outline-based interaction predominantly facilitated slide initiation and was infrequently employed for refinement. Conversely, selection-based interaction was utilized primarily for slide refinement and less commonly for initiation.

Interestingly, we found participants' preferences between outline-based and selection-based approaches were quite consistent throughout the slide creation task. As shown in Figure 7 (B), 10 out of 12 participants initially utilized outlines to plan slides as indicated by the long duration of outline-based slide initiation, followed by a mix of selection, manual editing, and outlines for slide refinement. Additionally, outline-based slide initiation happened even in later stages, focusing on adding slides through the incorporation of outline items, albeit with shorter duration. Selection-based refinement typically preceded manual refinement, signaling participants' inclination to refine slides through cell selection before manual adjustments. However, no discernible pattern was identified for the timing of outline-based slide refinement. In the meantime, two exceptions were observed: P9 did not write a complete outline first to generate all slides, instead, P9 progressively added each item into the outline and generated a slide each time (Figure 7 (B)-P9); P10 predominantly used selection to initiate slides one by one (Figure 7 (B)-P10). They both mentioned their habits of iteratively organizing materials for slide creation. Additionally, P9 stated, *"I don't trust LLM is capable of generating high quality slides for my entire outline at one time."*

We further confirm our findings in terms of how slides are created (i.e., by outline, selection, or a mix of both) using boxplots with medians (MDs) marked by red horizontal lines (Figure 8). In general, participants created 6 to 13 pages of slides in the user study (Figure 8 (A)). As shown in Figure 8 (B), the most prevalent approach involved a mix of outline-based and selection-based interactions, constituting a median proportion of 65% of the slides created in this manner, followed by outline (MD = 23%) and selection (MD = 8%).

7.2.2 Feedback on Advantages and Disadvantages. We present feedback from participants in the interview, regarding the advantages and disadvantages of outline-based and selection-based approaches.

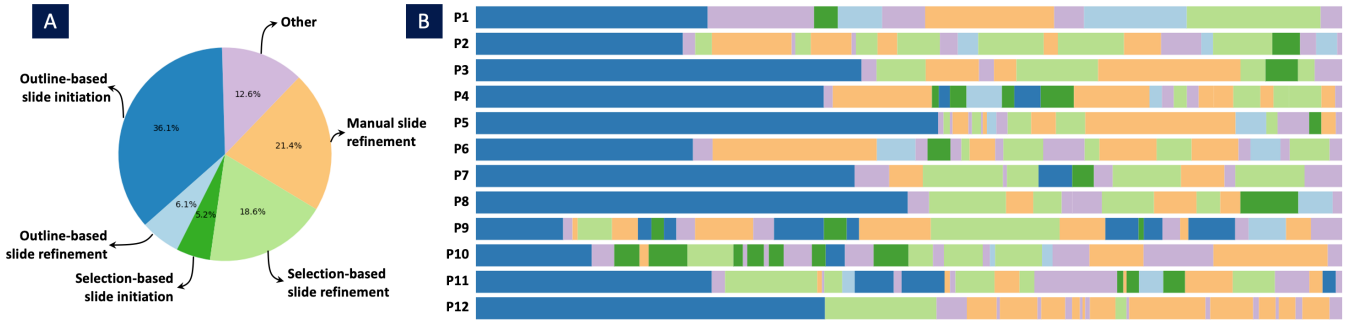

Figure 7: The figure showcases: (A) the average proportion of time spent on each activity of slide creation task over 12 participants (P1-P12) in the user study; (B) temporal sequence of activities in 12 separate slide creation sessions, each lasting about 15 to 30 minutes. To facilitate comparison, we normalized the duration of activities within each session for (A) and (B). These activities encompassed outline-based slide initiation, outline-based slide refinement, selection-based slide initiation, selection-based slide refinement, manual slide refinement (i.e., editing text and figures on slides), and other (e.g., looking through the user interface).

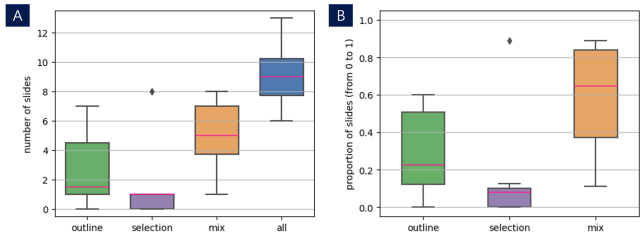

Figure 8: The figure illustrates the distribution of slide types (i.e., created by outline, selection, or a mix of both) across the 12 decks of slides created by our participants in the user study by boxplots: (A) number of slides; (B) proportion of slides. The red horizontal lines indicate the median values.

Outline-based Slides Generation. Participants praised the outline-based slide generation mainly for the following reasons. First, this approach helps them focus on structuring the slides instead of letting a single slide take too much attention or energy, as highlighted by 9 out of 12 participants (except P2, P6, and P9). They expressed that the outline-based approach allowed them to focus on the general and essential content of the slides, and free them from diving into a single slide. As P4 explained: *"Outlines serve as the backbone of the story delivered by slides. Personally, I prefer to spend more time refining the overall structure to make the entire story more logical. If I focus too much on slide details, it prevents me from achieving that."* Second, it fits their existing slide creation habits (P1, P5, P6, P7 and P12). As P7 said: *"Drafting an outline before creating slides is a common and good practice, which I follow in my day-to-day work."* Additionally, four participants (P1, P2, P5, P11) noted that the outline-based is more efficient for them. P1 mentioned that *"I just need to organize my story through outlines. OutlineSpark would help me find relevant cells from the notebook and convert them into slides. This process can be quite tedious if done manually. Although selecting cells individually to create slides is also helpful, I prefer not to search for these cells throughout the entire notebook."* P5 also said, *" When creating slides from notebooks, I have to find the associated cells used for slide creation. However, due to the cluttered nature of my notebooks, the cells before or after a specific cell may not necessarily be connected. This makes it quite time-consuming for me to locate these cells. "* Furthermore, participants P4 and P12 expressed that the outline-based approach provided them with a sense of control throughout the slide creation process.

Besides the advantages, participants pointed out potential improvements of the outline-based slide generation. First, 4 out of 12 participants (P1, P3, P4, P9) expressed that drafting outlines from scratch could be cumbersome, particularly for casual occasions (P9). They suggested the inclusion of a recommended draft outline at the beginning, such as from markdown cells in the notebook, which they can then refine to meet their preferences. Second, while most participants found the provided levels (i.e., topic and sub-topic) of outline sufficient, 3 participants suggested including more levels of outline for increased flexibility and granularity.

Selection-based Slides Generation. When it comes to the advantages of the selection-based approach, there are four reasons. First, the selection-based approach was recognized as being more accurate compared to the outline-based approach in terms of identifying relevant cells. Several participants (P1, P9, P10, P12) said that when selecting cells, the generated slides can better meet their expectations in some cases. Second, the selection-based approach was considered as a complementary role to the outline-based approach. Some participants used this approach mainly to refine the generated slides (P1, P2, P5, and P7), such as dealing with inaccurately retrieved cells, seeking inspiration, or addressing any missing points they noticed in the outline. Interestingly, three participants (P2, P6, and P10) mentioned that sometimes they knew which cells to use for a slide, but struggled with summarizing them as a topic in the outline. In such cases, they found the selection-based approach was beneficial. Third, the selection-based approach was aligned with some participants' habits of iteratively organizing materials to create slides (P9 and P10). Rather than setting up outlines from the beginning, they preferred a more iterative approach. Additionally, this approach was considered more efficient in certain cases. P9 and

P10 noted that when the notebook or slides were simple, it took less time to select cells compared to typing out the outlines.

As for disadvantages, the most mentioned one was its general low efficiency. P12 expressed: *"This approach focused on creating slides directly from notebook cells but provided minimal support for slide ideation, which should occupy most of the time when making slides."* P1 and P3 highlighted the burden of having to manually find and select cells from the notebook to create slides one by one.

7.3 OutlineSpark Streamlines Slide Ideation and Creation from Computational Notebooks

Next, we report the feedback of our participants regarding some critical functions and designs of OutlineSpark.

Keyword Extraction. The inclusion of keywords in *Notebook Overview* was appreciated by all participants, as it allowed them to gain an overview of the notebook's content without delving into the actual code. As P6 said: *"I don't want to read the code, I think it's overwhelming when making slides. The keywords really help me grasp a big picture of the notebook quickly."* Three participants (P2, P4, and P10) echoed with p6. Furthermore, P12 mentioned that the keywords served as a source of inspiration when drafting outlines. Additionally, participants found the keywords helpful in locating specific cells when refining the generated slides, further streamlining the slide creation process. While participants praised the keywords for effectively summarizing the content of each cell, four participants (P2, P3, P5, and P8) suggested an improvement. They recommended including X and Y labels in the keywords for charts to help them understand what the chart is about, rather than just displaying keywords such as "bar plot", "histogram", etc.

Topic Recommendation. The topic recommendation was not as favored by participants as other functions. Two participants (P5 and P6) reported that the recommended topics were too specific to the notebook content, while they preferred more general ones. Furthermore, P2 and P7 suggested that they forgot this function as it is not explicitly indicated in the interface and thus the usage of it is unintuitive. Participant P10 gave a negative score of 2 and indicated that the recommendations did not consider the context of what P10 had written in the outline. However, after reviewing the recorded video, we found it was the latency of LLM that resulted in the failure to present the latest recommendations to P10. On the other, 5 participants expressed appreciation for the topic recommendation, as it alleviated certain burdens of drafting outlines. They turned to the recommendation to get inspiration on what could be added (P7: *"I can turn to the recommendation for help when I have no idea what else to present."*) and to check if important content had been missing. Interestingly, P12 used the recommended topics to refine his outlines. For example, he transformed "Outliers" into "Removing Outliers" and "Important Features" into "Finding Important Features." To sum up, we observed that the recommendation feature played an assistant role for participants, which aligned with the design of allowing users to trigger the recommendation only when desired.

Cell Retrieval. When asked about retrieving relevant cells from the notebook, 9 out of 12 participants expressed appreciation for its ability to alleviate the manual process of locating cells when creating slides. P5 stated, *"OutlineSpark really helps me find desired cells from the notebook. My notebooks are often long and messy, and I usually have difficulty locating cells."* Although the retrieved cells were not perfect, participants needed to refer back to check the retrieved cells if the slides didn't meet their expectations, they still valued OutlineSpark's assistance in locating cells from the notebook. P1 specifically mentioned, *"I don't need to look through the whole notebook. Even if the retrieved cells don't match the one I need, it does narrow down the search space."* And this was echoed by P12.

Slide Generation. In terms of slide generation from notebook cells, all the participants thought it reduced the workload in making slides. According to P1, *"The generated slides match what I want, and I only need to make slight adjustments."* P9 and P11 echoed with P1. Moreover, P8 and P10 appreciated that OutlineSpark saved their time as it automatically arranged titles, bullet points, and charts/tables on slides. However, participants did provide some suggestions regarding the generated bullet points. Both P4 and P10 suggested that while OutlineSpark effectively summarized each cell, it would be better to merge similar cells into a single bullet point to avoid repetition. Additionally, P2 and P8 expressed the need for automatically extracting insights from the output of cells, particularly for those outputting charts.

8 SLIDES QUALITY ASSESSMENT

We evaluated slide quality based on presenters' self-assessment using Q7 in Figure 5; however, these self-impressions may not align with how audiences perceive the slides. Inspired by Slide4N [57], we invited 8 participants to rate the 12 slide decks created by our participants in the user study, followed by a short interview regarding the justifications for ratings. Ratings covered five aspects: overall satisfaction, clarity of structure, ease of understanding the content, slide layout, and aesthetics.

These participants (4 females, 4 males; aged 25.3 ± 2.1) were recruited through social media and word of mouth. Among them, 6 were postgraduate researchers (referred to as peer reviewers, R1-R6), and 2 were university faculties (referred to as expert reviewers, R7-R8) with extensive experience in presentation slide preparation, delivery, and evaluation. They came from diverse fields, including data science, human-computer interaction, visualization, machine learning, and computer vision. The studies were conducted through one-to-one online meetings, each lasting about 30 minutes. Each participant received a compensation of $7 for completing the study.

The results of the ratings are illustrated in the boxplots in Figure 9 (detailed ratings can be found in the supplementary material). Overall, both peer and expert reviewers expressed satisfaction with the slides created with OutlineSpark, yielding a median rating of 6 for peers and 5 for experts (overall satisfaction). While peers' ratings on other aspects remained high, experts rated relatively lower. Upon closer examination, it was observed that R8 gave lower ratings on all aspects, especially on slide layout and aesthetics. When asked about his concerns, R8 expressed, *" In terms of layout, the text and figures on the slides are detached [when there are multiple figures], requiring additional time for the audiences to match them. Aesthetically, the text lacks some highlights, making the slides appear dull over time, and it is challenging for the audience to quickly identify the message the presenter intends to convey. Moreover, some bar charts on certain slides are too large, with small accompanying text,*

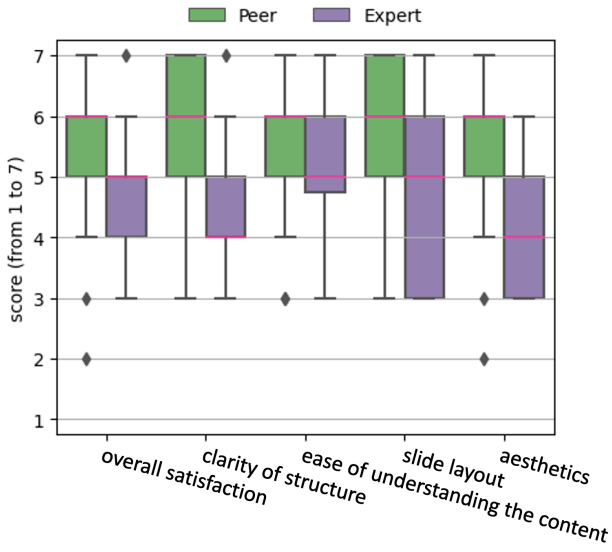

Figure 9: The figure displays audience ratings for the 12 slides created by OutlineSpark in the user study. Ratings are based on a 7-point Likert Scale (1 = "strongly disagree," 7 = "strongly agree") across five aspects. The red horizontal lines indicate the median ratings.

resulting in a lack of cohesion in the slides." Similarly, two peers (R5 and R6) suggested improving the aesthetic of the generated slides. As stated by R6: *"While aesthetics is not the main consideration when rating the overall satisfaction of slides, certain options can be added to further polish the preset neat slides provided by OutlineSpark, such as text highlighting, fonts, and colors."* To sum up, the generation of slides can be improved in terms of the aesthetics and layout.

9 DISCUSSION

This section first discusses the lessons learned from the research in Section 9.1. Then we point out the limitation and potential future work in Section 9.2.

9.1 Design Lessons

In this section, we present the design lessons learned from our research that can serve as inspiration for the design of future tools.

Outline-based slides creation enhances users' ideation and eliminates their workload. Crafting outlines before creating slides is a recommended practice [1, 30, 48, 64]. Building on this concept, we design OutlineSpark that supports this workflow for creating slides from computational notebooks. With OutlineSpark, users only need to conceptualize the message they want to convey using outlines, while the tool handles the labor-intensive aspects of slide creation, such as locating relevant cells, distilling key information, and arranging content on slides, to help them to transfer outlines into slides. In this way, OutlineSpark further bridges the gap between data analysis and presentation. During our user study, most participants appreciated this workflow, as it aligned with their existing habits and allowed them to plan slides at a high level, resulting in well-structured slides with little effort. Consequently, users can dedicate more time to the most critical aspects of the presentation—ideation and planning—rather than getting bogged down with tedious details and tasks, such as locating cells. We believe future tools can extend the outline-based workflow to other scenarios where individuals gather a set of materials for creating communication materials, such as data articles and videos.

Outline-based and selection-based interaction should be combined for high efficiency. OutlineSpark provides two types of interactions for users to specify their intent in slide content, i.e., writing an outline or selecting cells. We found both of them have their own advantages and limitations. The outline-based approach offers flexibility, facilitating users' ideation and allowing users to freely express their ideas. However, the comprehensibility of these outlines can sometimes pose challenges for the AI system, resulting in suboptimal retrieval that requires further user refinement. On the other hand, the selection-based approach offers higher accuracy as users directly choose specific cells for slide generation. However, this method can become burdensome when dealing with lengthy and messy notebooks which are common [15, 50]. In our user study, we noticed that users often wrote outlines to build their slides rapidly and then leveraged the selection-based approach to fine-tune the generated slides. They combined both interaction approaches in the task for a more convenient slide creation experience. Based on the design lesson, we would like to suggest that future tools should consider combining them when designing slide creation tools.

OutlineSpark facilitates effective collaboration between humans and AI. OutlineSpark adopts a Human-AI collaboration workflow to facilitate the creation of presentation slides from computational notebooks through outlines. The AI assistance in OutlineSpark encompasses two aspects: supporting outline creation and transforming outlines into slides. When using OutlineSpark, users first craft outlines with the aid of AI, such as keywords in *Notebook Overview* and recommended topics in *Outline Panel*. Subsequently, the AI generates a deck of slides. Finally users refine them to meet their expectations. In our user study, all participants highly appreciated the level of AI assistance in reducing manual work in this tedious task. Some even expressed the view that they did not expect full automation in slide creation, even in the time of LLM. They believed that the core elements of the task should be controlled by humans, such as the outline in our case, while AI could assume responsibility for handling certain tedious and repetitive tasks. Considering that individuals have diverse preferences when it comes to slide creation, a promising future direction is to incorporate users' previous slides as input to the AI. By leveraging this historical data, AI could learn from users' past slide designs and generate slides that align with their preferences. This approach would not only generate more personalized slides but also reduce the workload required for slide refinements.

9.2 Limitations and Future Work

In this section, we discuss the limitations and future work of our research, regarding the retrieval accuracy, functionalities, and the evaluations of OutlineSpark.

Retrieval Accuracy. With prolonged use of OutlineSpark, users may overly trust in AI assistance. However, the AI's support may

introduce errors, potentially misleading or being overlooked by users. For instance, the LLM may fail to retrieve cells the users expected as evident in the user study (relevant failed cases can be found in the supplementary material). It may encounter challenges in comprehending abstract outline items that require additional inference or exhibiting inconsistency in responding to similar or identical queries, leading to retrieved cells that over-represent or under-represent the outlines. Furthermore, the performance of OutlineSpark may be influenced by how organized the notebook is. An unorganized notebook may have excess or lengthy code cells. OutlineSpark may generate redundant content in the slides when excess cells are retrieved or when only parts of a lengthy cell are what the users desire while OutlineSpark retrieves the whole cell. To improve the performance of OutlineSpark, we plan to investigate methods of cleaning computational notebooks [11, 15, 22], segmenting a lengthy cell for sub-cell retrieval, and enhancing AI's understanding of users' ways of expressing intentions by learning from historical correct pairs of outlines and desired notebook cells.

Functionalities. The functionalities of OutlineSpark can be further extended. First, presentation slides in practice may involve multi-layered structures. However, OutlineSpark currently provides only two levels of outlines (i.e., topic and sub-topic), which is insufficient in such cases (also pointed out by three participants in the user study). Future studies could explore adaptive prompts, allowing OutlineSpark to adapt to various levels of cell prioritization based on outline depth and complexity. Additionally, conducting an empirical study to explore types of user-crafted outline items would be beneficial. Second, OutlineSpark currently can't access Python kernel. It restricts the ability to generate bullet points for chart and table findings that are stored as variables in the kernel, which is considered a point for future improvement by user study participants (P2 and P8). Following the work on insight generation for charts and tables [13, 31, 34, 58], OutlineSpark can be enhanced to generate such bullet points. Moreover, participants held different opinions regarding the function of recommending potential topics in the outline panel. While the recommendation feature was deemed helpful by some participants, others expressed concerns regarding its alignment with their intentions (P5 and P6) and its intuitiveness (P2 and P7). To cater to different user preferences, future work could consider leveraging insights from users' previous outlines to provide more tailored topic recommendations. Additionally, enhancing intuitiveness could be achieved by incorporating a placeholder for blank outline items, such as including a prompt like "Press space for topic recommendation". Furthermore, participants R5, R6, and R8 desired the AI-generated slides to be more aesthetically pleasing. While OutlineSpark supports markdown-based editing for styling, it was less convenient than tools like Microsoft PowerPoint [37]. In response, users can export the generated slides as .pptx files for further editing and beautification in PowerPoint. Simultaneously, OutlineSpark could be enhanced to maintain text-visual layout coherence, thereby reducing users' editing time.

Evaluations. In the user study, participants were tasked with creating a slide deck for a notebook using OutlineSpark. The evaluation can be improved regarding four perspectives. First, while we assessed the effectiveness of OutlineSpark's main features, we did not evaluate AI assistance concerning trust, accuracy, and perceived cognitive load. Examining these aspects could offer additional insights into the tool's overall efficiency. Second, the quality of the slides could be further evaluated. While we assessed the quality of slides created with OutlineSpark from the perspective of both presenters and audiences, a more comprehensive evaluation could be achieved by comparing these AI-generated slides with manually crafted ones (without AI assistance, e.g., PowerPoint), which may provide additional insights into differences in slide quality and efficiency. Third, a more long-term evaluation would be beneficial. Following previous work [50, 55], we simulated real-world notebooks by removing markdown cells and working with 42 code cells. However, real-world analysis in computational notebooks can be more complex. It would be valuable to gather feedback from users who utilize OutlineSpark in their daily work over an extended period. Last, we acknowledge that the coverage of participants in our user study was limited. We found that the number of participants was relatively small and no data scientists from the industry were involved. To better assess the value of OutlineSpark in real-world settings, future work is needed to conduct a long-term and comparative study with a larger and more diverse pool of participants.

10 CONCLUSION

This paper introduced OutlineSpark, an interactive and intelligent plugin to streamline the process of creating slides from computational notebooks via outlines. Utilizing the capabilities of large language models, OutlineSpark automates the process by allowing users to define the presentation's structure and content through customized outlines. The tool then automatically retrieves relevant notebook cells, extracts essential information, and organizes it within the slides. A user study demonstrated that OutlineSpark streamlined the slides ideation and creation process from computational notebooks. Furthermore, participants found that outline-based slide creation aligned well with their existing practices of crafting presentations and allowed them to concentrate on conceptualizing the narrative logic. In summary, OutlineSpark has taken a step forward in bridging the gap between computational notebooks and presentation slides.

ACKNOWLEDGMENTS

We would like to thank the reviewers and all participants in our studies for their valuable input. The research was partially supported by the National Natural Science Foundation of China (Grant No. 62172289) and the Hong Kong Research Grants Council (GRF16210722).

REFERENCES

[1] Robert RH Anholt. 2010. *Dazzle'em with style: The art of oral scientific presentation.* Elsevier, San Diego, California, USA.
[2] Damian Avila. 2019. RISE. https://github.com/damianavila/RISE
[3] Lawrence Bergman, Jie Lu, Ravi Konuru, Julie MacNaught, and Danny Yeh. 2010. Outline wizard: presentation composition and search. In *Proceedings of the 15th International Conference on Intelligent User Interfaces* (Hong Kong, China) *(IUI '10).* Association for Computing Machinery, New York, NY, USA, 209–218. https://doi.org/10.1145/1719970.1719999
[4] Avinash Bhat, Austin Coursey, Grace Hu, Sixian Li, Nadia Nahar, Shurui Zhou, Christian Kästner, and Jin L.C. Guo. 2023. Aspirations and Practice of ML Model Documentation: Moving the Needle with Nudging and Traceability. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23).* Association for Computing Machinery, New York, NY, USA, Article 749, 17 pages. https://doi.org/10.1145/3544548.3581518

[5] Matthias Blohm, Glorianna Jagfeld, Ekta Sood, Xiang Yu, and Ngoc Thang Vu. 2018. Comparing attention-based convolutional and recurrent neural networks: Success and limitations in machine reading comprehension. *arXiv preprint arXiv:1808.08744* abs/1808.08744 (2018).

[6] Matthew Brehmer and Robert Kosara. 2021. From jam session to recital: Synchronous communication and collaboration around data in organizations. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2021), 1139–1149.

[7] John Brooke. 1996. SUS: A Quick and Dirty Usability. *Usability evaluation in industry* 189, 3 (1996), 189–194.

[8] Souti Chattopadhyay, Ishita Prasad, Austin Z. Henley, Anita Sarma, and Titus Barik. 2020. What's Wrong with Computational Notebooks? Pain Points, Needs, and Design Opportunities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376729

[9] Fanny Chevalier, Melanie Tory, Bongshin Lee, Jarke van Wijk, Giuseppe Santucci, Marian Dörk, and Jessica Hullman. 2018. From analysis to communication: Supporting the lifecycle of a story. In *Data-Driven Storytelling*. AK Peters/CRC Press, Boca Raton, FL, USA, 151–183.

[10] David Donoho. 2017. 50 years of data science. *Journal of Computational and Graphical Statistics* 26, 4 (2017), 745–766.

[11] Ian Drosos, Titus Barik, Philip J. Guo, Robert DeLine, and Sumit Gulwani. 2020. Wrex: A Unified Programming-by-Example Interaction for Synthesizing Readable Code for Data Scientists. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376442

[12] Will EPPerson, Doris Jung-Lin Lee, Leijie Wang, Kunal Agarwal, Aditya G. Parameswaran, Dominik Moritz, and Adam Perer. 2022. Leveraging Analysis History for Improved In Situ Visualization Recommendation. *Computer Graphics Forum* 41, 3 (2022), 145–155. https://doi.org/10.1111/cgf.14529 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14529

[13] Heng Gong, Xiaocheng Feng, Bing Qin, and Ting Liu. 2019. Table-to-Text Generation with Effective Hierarchical Encoder on Three Dimensions (Row, Column and Time). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Association for Computational Linguistics, Stroudsburg, PA, USA, 3141–3150.

[14] Philip Jia Guo. 2012. *Software tools to facilitate research programming*. Stanford University, Stanford, California, USA.

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

[16] Jeffrey Heer. 2019. Agency plus automation: Designing artificial intelligence into interactive systems. *Proceedings of the National Academy of Sciences* 116, 6 (2019), 1844–1850.

[17] Project Jupyter. 2015. *Project Jupyter: Computational Narratives as the Engine of Collaborative Data Science*. the Helmsley Trust, the Gordon and Betty Moore Foundation and the Alfred P. Sloan Foundation. https://blog.jupyter.org/project-jupyter-computational-narratives-as-the-engine-ofcollaborative-data-science-2b5fb94c3c58

[18] Project Jupyter. 2018. *JupyterLab: the next generation of the Jupyter Notebook*. Project Jupyter. https://blog.jupyter.org/jupyterlab-thenext-generation-of-the-jupyter-notebook-5c949dabea3

[19] DaYe Kang, Tony Ho, Nicolai Marquardt, Bilge Mutlu, and Andrea Bianchi. 2021. ToonNote: Improving Communication in Computational Notebooks Using Interactive Data Comics. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 727, 14 pages. https://doi.org/10.1145/3411764.3445434

[20] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and individual differences* 103 (2023), 102274.

[21] Mary Beth Kery, Amber Horvath, and Brad Myers. 2017. Variolite: Supporting Exploratory Programming by Data Scientists. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 1265–1276. https://doi.org/10.1145/3025453.3025626

[22] Mary Beth Kery, Bonnie E. John, Patrick O'Flaherty, Amber Horvath, and Brad A. Myers. 2019. Towards Effective Foraging by Data Scientists to Find Past Analysis Choices. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300322

[23] Mary Beth Kery, Marissa Radensky, Mahima Arya, Bonnie E John, and Brad A Myers. 2018. The story in the notebook: Exploratory data science using a literate programming tool. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. ACM, New York, United States, 1–11.

[24] Mary Beth Kery, Donghao Ren, Fred Hohman, Dominik Moritz, Kanit Wongsuphasawat, and Kayur Patel. 2020. 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* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 140–151. https://doi.org/10.1145/3379337.3415842

[25] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. 2016. The emerging role of data scientists on software development teams. In *Proceedings of the 38th International Conference on Software Engineering* (Austin, Texas) *(ICSE '16)*. Association for Computing Machinery, New York, NY, USA, 96–107. https://doi.org/10.1145/2884781.2884783

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

[27] Sean Kross and Philip Guo. 2021. Orienting, framing, bridging, magic, and counseling: How data scientists navigate the outer loop of client collaborations in industry and academia. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–28.

[28] Sean Kross and Philip J Guo. 2019. Practitioners teaching data science in industry and academia: Expectations, workflows, and challenges. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. ACM, New York, USA, 1–14.

[29] Doris Jung-Lin Lee, Dixin Tang, Kunal Agarwal, Thyne Boonmark, Caitlyn Chen, Jake Kang, Ujjaini Mukhopadhyay, Jerry Song, Micah Yong, Marti A. Hearst, and Aditya G. Parameswaran. 2021. Lux: always-on visualization recommendations for exploratory dataframe workflows. *Proc. VLDB Endow.* 15, 3 (nov 2021), 727–738. https://doi.org/10.14778/3494124.3494151

[30] Haotian Li, Yun Wang, Q. Vera Liao, and Huamin Qu. 2023. Why is AI not a Panacea for Data Workers? An Interview Study on Human-AI Collaboration in Data Storytelling. arXiv:2304.08366 [cs.HC]

[31] Haotian Li, Lu Ying, Haidong Zhang, Yingcai Wu, Huamin Qu, and Yun Wang. 2023. Notable: On-the-fly Assistant for Data Storytelling in Computational Notebooks. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 173, 16 pages. https://doi.org/10.1145/3544548.3580965

[32] Xingjun Li, Yuanxin Wang, Hong Wang, Yang Wang, and Jian Zhao. 2021. NBSearch: Semantic Search and Visual Exploration of Computational Notebooks. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, Vol. abs/2102.01275. ACM, New York, USA, 1–14.

[33] Xingjun Li, Yizhi Zhang, Justin Leung, Chengnian Sun, and Jian Zhao. 2023. EDAssistant: Supporting Exploratory Data Analysis in Computational Notebooks with In Situ Code Search and Recommendation. *ACM Transactions on Interactive Intelligent Systems* 13, 1 (2023), 1–27.

[34] Yanna Lin, Haotian Li, Leni Yang, Aoyu Wu, and Huamin Qu. 2024. InkSight: Leveraging Sketch Interaction for Documenting Chart Findings in Computational Notebooks. *IEEE Transactions on Visualization and Computer Graphics* 30, 1 (2024), 944–954. https://doi.org/10.1109/TVCG.2023.3327170

[35] Vivian Liu and Lydia B Chilton. 2022. Design Guidelines for Prompt Engineering Text-to-Image Generative Models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 384, 23 pages. https://doi.org/10.1145/3491102.3501825

[36] Yaoli Mao, Dakuo Wang, Michael Muller, Kush R Varshney, Ioana Baldini, Casey Dugan, and Aleksandra Mojsilović. 2019. How data scientistswork together with domain experts in scientific collaborations: To find the right answer or to ask the right question? *Proceedings of the ACM on Human-Computer Interaction* 3, GROUP (2019), 1–23.

[37] Microsoft. 2024. *Microsoft PowerPoint*. Microsoft. https://www.microsoft.com/en-us/microsoft-365/powerpoint

[38] Michael Muller, Ingrid Lange, Dakuo Wang, David Piorkowski, Jason Tsay, Q Vera Liao, Casey Dugan, and Thomas Erickson. 2019. How data science workers work with data: Discovery, capture, curation, design, creation. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. ACM, New York, USA, 1–15.

[39] David Munechika, Zijie J. Wang, Jack Reidy, Josh Rubin, Krishna Gade, Krishnaram Kenthapadi, and Duen Horng Chau. 2022. Visual Auditor: Interactive Visualization for Detection and Summarization of Model Biases. In *2022 IEEE Visualization and Visual Analytics (VIS)*. IEEE, New York, NY, USA, 45–49. https://doi.org/10.1109/VIS54862.2022.00018

[40] Andrew Myers. 2022. *In Human-Centered AI, the Boundaries Between UX and Software Roles Are Evolving*. Stanford University. https://hai.stanford.edu/news/human-centered-ai-boundaries-between-ux-and-software-roles-are-evolving

[41] Jorge Piazentin Ono, Sonia Castelo, Roque Lopez, Enrico Bertini, Juliana Freire, and Claudio Silva. 2020. Pipelineprofiler: A visual analytics tool for the exploration of automl pipelines. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2020), 390–400.

[42] OpenAI. 2023. GPT-3.5. https://platform.openai.com/docs/models/gpt-3-5

[43] OpenAI. 2023. GPT best practices. https://platform.openai.com/docs/guides/gpt-best-practices

[44] Yi-Hao Peng, Peggy Chi, Anjuli Kannan, Meredith Ringel Morris, and Irfan Essa. 2023. Slide Gestalt: Automatic Structure Extraction in Slide Decks for Non-Visual Access. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 829, 14 pages.

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

[46] David Piorkowski, Soya Park, April Yi Wang, Dakuo Wang, Michael Muller, and Felix Portnoy. 2021. How AI Developers Overcome Communication Challenges in a Multidisciplinary Team: A Case Study. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW1, Article 131 (apr 2021), 25 pages. https://doi.org/10.1145/3449205

[47] Deepthi Raghunandan, Zhe Cui, Kartik Krishnan, Segen Tirfe, Shenzhi Shi, Tejaswi Darshan Shrestha, Leilani Battle, and Niklas Elmqvist. 2022. Lodestar: Supporting Independent Learning and Rapid Experimentation Through Data-Driven Analysis Recommendations. arXiv:2204.07876

[48] Garr Reynolds. 2011. *Presentation Zen: Simple ideas on presentation design and delivery.* New Riders, California, USA.

[49] Adam Rule, Ian Drosos, Aurélien Tabard, and James D Hollan. 2018. Aiding collaborative reuse of computational notebooks with annotated cell folding. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 1–12.

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

[51] Jeff Sauro and James R Lewis. 2016. *Quantifying the user experience: Practical statistics for user research.* Morgan Kaufmann, San Francisco, CA, USA.

[52] Athar Sefid, Prasenjit Mitra, and Lee Giles. 2021. SlideGen: an abstractive section-based slide generator for scholarly documents. In *Proceedings of the 21st ACM Symposium on Document Engineering* (Limerick, Ireland) *(DocEng '21)*. Association for Computing Machinery, New York, NY, USA, Article 11, 4 pages. https://doi.org/10.1145/3469096.3474939

[53] Krishna Subramanian, Johannes Maas, and Jan Borchers. 2020. Tractus: Understanding and supporting source code experimentation in hypothesis-driven data science. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–12.

[54] Aurélien Tabard, Wendy E Mackay, and Evelyn Eastmond. 2008. From individual to collaborative: the evolution of prism, a hybrid laboratory notebook. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*. Association for Computing Machinery, New York, NY, USA, 569–578.

[55] April Yi Wang, Dakuo Wang, Jaimie Drozdal, Michael Muller, Soya Park, Justin D Weisz, Xuye Liu, Lingfei Wu, and Casey Dugan. 2022. Documentation matters: Human-centered AI system to assist data science code documentation in computational notebooks. *ACM Transactions on Computer-Human Interaction* 29, 2 (2022), 1–33.

[56] Dakuo Wang, Q Vera Liao, Yunfeng Zhang, Udayan Khurana, Horst Samulowitz, Soya Park, Michael Muller, and Lisa Amini. 2021. How much automation does a data scientist want? *arXiv preprint arXiv:2101.03970* abs/2101.03970 (2021).

[57] Fengjie Wang, Xuye Liu, Oujing Liu, Ali Neshati, Tengfei Ma, Min Zhu, and Jian Zhao. 2023. Slide4N: Creating Presentation Slides from Computational Notebooks with Human-AI Collaboration. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 364, 18 pages. https://doi.org/10.1145/3544548.3580753

[58] Yun Wang, Zhida Sun, Haidong Zhang, Weiwei Cui, Ke Xu, Xiaojuan Ma, and Dongmei Zhang. 2019. Datashot: Automatic generation of fact sheets from tabular data. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 895–905.

[59] Zijie J Wang, Katie Dai, and W Keith Edwards. 2022. Stickyland: Breaking the linear presentation of computational notebooks. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. Association for Computing Machinery, New York, NY, USA, 1–7.

[60] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent Abilities of Large Language Models. arXiv:2206.07682

[61] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

[62] Nathaniel Weinman, Steven M. Drucker, Titus Barik, and Robert DeLine. 2021. Fork It: Supporting Stateful Alternatives in Computational Notebooks. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 307, 12 pages. https://doi.org/10.1145/3411764.3445527

[63] John Wenskovitch, Jian Zhao, Scott Carter, Matthew Cooper, and Chris North. 2019. Albireo: An Interactive Tool for Visually Summarizing Computational Notebook Structure. In *2019 IEEE Visualization in Data Science (VDS)*. IEEE, New York, NY, USA, 1–10. https://doi.org/10.1109/VDS48975.2019.8973385

[64] Edward D Zanders, Edward Zanders, and Lindsay MacLeod. 2018. *Presentation skills for scientists: a practical guide.* Cambridge University Press, Cambridge, England.

[65] Amy X. Zhang, Michael Muller, and Dakuo Wang. 2020. How do Data Science Workers Collaborate? Roles, Workflows, and Tools. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW1, Article 22 (may 2020), 23 pages. https://doi.org/10.1145/3392826

[66] Chengbo Zheng, Dakuo Wang, April Yi Wang, and Xiaojuan Ma. 2022. Telling Stories from Computational Notebooks: AI-Assisted Presentation Slides Creation for Presenting Data Science Work. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 53, 20 pages. https://doi.org/10.1145/3491102.3517615

[67] Ingrid Zukerman and Diane Litman. 2001. Natural language processing and user modeling: Synergies and limitations. *User modeling and user-adapted interaction* 11 (2001), 129–158.