

How Do Analysts Understand and Verify AI-Assisted Data Analyses?

Ken Gu
kenqgu@cs.washington.edu
University of Washington
Seattle, WA, USA

Ruoxi Shang
rxshang@uw.edu
University of Washington
Seattle, WA, USA

Tim Althoff
althoff@cs.washington.edu
University of Washington
Seattle, WA, USA

Chenglong Wang
chenwang@microsoft.com
Microsoft Research
Redmond, WA, USA

Steven M. Drucker
sdrucker@microsoft.com
Microsoft Research
Redmond, WA, USA

ABSTRACT

Data analysis is challenging as it requires synthesizing domain knowledge, statistical expertise, and programming skills. Assistants powered by large language models (LLMs), such as ChatGPT, can assist analysts by translating natural language instructions into code. However, AI-assistant responses and analysis code can be misaligned with the analyst's intent or be seemingly correct but lead to incorrect conclusions. Therefore, validating AI assistance is crucial and challenging. Here, we explore how analysts understand and verify the correctness of AI-generated analyses. To observe analysts in diverse verification approaches, we develop a design probe equipped with natural language explanations, code, visualizations, and data tables with common data operations. Through a qualitative user study ($n=22$) using this probe, we uncover common behaviors within verification workflows and how analysts' programming, analysis, and tool backgrounds reflect these behaviors. Additionally, we provide recommendations for analysts and highlight opportunities for designers to improve future AI-assistant experiences.

KEYWORDS

Data Science, Verifying AI-Generated Output, Human-AI Interaction

ACM Reference Format:

Ken Gu, Ruoxi Shang, Tim Althoff, Chenglong Wang, and Steven M. Drucker. 2024. How Do Analysts Understand and Verify AI-Assisted Data Analyses?. 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, 21 pages. <https://doi.org/10.1145/3613904.3642497>

1 INTRODUCTION

Data analysis is challenging as practitioners must synthesize domain knowledge with computational expertise to draw reliable

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

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

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 979-8-4007-0330-0/24/05...\$15.00
<https://doi.org/10.1145/3613904.3642497>

conclusions. For practitioners, computational challenges may act as a barrier to exploring alternative and valid analysis decisions, and therefore conducting robust analyses [84, 86].

Improved capabilities of Large Language Models (LLMs) for general [18, 22, 24], data science [19, 76, 138], and visualization [30] programming tasks suggest LLM-based analysis assistants as a promising solution, enabling data analysts to *execute* and *automate* their analyses. Given the widespread deployment of assistants such as OpenAI's Code Interpreter [96], there is an emerging paradigm in how analysts perform analyses. Rather than formulate the exact transformations and operations in an analysis, analysts specify their analysis question through a natural language intent (Fig. 1 right) [85]. The AI assistant then performs the analysis task, writes code to understand the data, makes the necessary data transformations, and interprets the results. Since many analysis tasks are time-consuming [20, 54, 55, 84], even for analysts well-versed in programming, natural language interactions can be more efficient especially when abstraction of the intent is possible. Furthermore, this natural language communication can broaden the usability of AI analysis tools, making them accessible even for analysts who are non-expert programmers [106].

Yet, a significant challenge remains. AI-based assistants, while powerful, can misinterpret user intentions (see Fig. 1) [48, 79, 137] or make seemingly correct erroneous outputs [13, 47]. Since misinterpretations is inherent in the flexibility of natural language communication [85, 106, 114], errors from existing AI-based assistants are likely to persist no matter what the underlying LLM is. Moreover, because conclusions from data analyses often inform high-impact decisions in science [4, 8], business [54, 66, 67], and government [42, 124], it is important for users of AI-based analysis assistants to critically reflect on the assistant's outputs [106]. In this new paradigm, analysts' focus now shifts from writing and validating their own analyses to understanding and evaluating the analyses generated by AI assistants [79]. This can be challenging as existing AI assistants lack verification support [106, 112] and even experts can be susceptible to automation bias and overreliance [100, 132].

Recent work in the HCI community explored ways to provide the user with a better understanding of LLM-generated outputs, including those from AI-based code assistants [120, 127]. However, these studies focus only on programmers and not the specific domain of data analysis, missing a significant portion of data analysts [26, 84].

Traditional Data Analysis

The screenshot illustrates a traditional data analysis workflow. At the top, a spreadsheet window titled 'order_status' shows a table with columns 'sku', 'order_date', 'quantity', and 'order_status'. A filter dialog is open, showing a 'Filter' section with 'Returned...' selected under 'Equals' and 'And' selected under 'And/Or'. Below the spreadsheet is a computational notebook interface with the following Python code:

```

[1]: import pandas as pd
df = pd.read_excel('orders_data.xlsx')
df.head()
***
```

```

[2]: df['order_date'] = pd.to_datetime(df['order_date'])
df['order_date'].head()
```

```

[2]: 0 2021-07-18 22:38:00
1 2021-10-19 18:05:00
2 2021-11-28 22:20:00
3 2021-07-28 04:06:00
4 2021-09-28 14:50:00
Name: order_date, dtype: datetime64[ns]
```

```

[3]: # filter for orders in November
df_nov = df[df['order_date'].dt.month == 11]
df_nov['order_date'].dt.month.value_counts()
```

```

[3]: 11 29
Name: order_date, dtype: int64
```

```

[4]: # filter for returned to seller
df_filt = df_nov[df_nov['order_status'] == 'Returned to seller']
df_filt[['sku', 'order_date', 'order_status', 'quantity']]
```

```

[4]:      sku       order_date   order_status  quantity
22  SKU: DN-0wdx-vyot  2021-11-26 19:22:00  Returned to seller      1
39  SKU: 54-d265-b74k  2021-11-26 21:12:00  Returned to seller      4
```

AI-Assisted Data Analysis

The screenshot illustrates an AI-assisted data analysis workflow. At the top, an AI interface asks, "How many items purchased within the month of November were returned to seller?". Below it, a code editor interface shows the generated Python code:

```

# Convert the 'order_date' column to datetime format
orders_data['order_date'] = pd.to_datetime(orders_data['order_date'], errors='coerce')

# Filter the data for orders made in the month of November
november_orders = orders_data[orders_data['order_date'].dt.month == 11]

# Count the number of items that were returned from the filtered data
returned_items = november_orders[november_orders['order_status'].str.contains('Returned to seller')].groupby(['order_date']).size()

# Get the total number of returned items
total_returned_items = returned_items['quantity'].sum()
```

At the bottom, a results interface shows the output: "total_returned_items" with a value of 5. A note at the bottom states, "A total of 5 items purchased in the month of November were returned to the seller."

Figure 1: Analysts may now need to understand and verify AI-assisted analyses. In traditional analysis workflows, analysts specify and execute their data operations using tools such as computational notebooks (A) or spreadsheets (B). Engaged in these operations, analysts are familiar with the process and results of their work (e.g., reports, code, tables, and visualizations). However, with AI-assisted analysis, analysts can convey their intentions using natural language (e.g., *"How many items purchased within the month of November were returned to the seller?"*). The AI assistant handles the task of specifying and performing the data operations. This shift requires analysts, including those who may not be familiar with the underlying execution language the assistant uses, to understand and verify the process and results of the assistant (C). In this paper, we study the workflows analysts with varied backgrounds use to understand and verify AI-assisted data analyses.

Since data analysis involves iteratively making sense of the underlying data [69, 71, 104], analysts must juggle their mental model of the data in addition to their understanding of the AI's procedure. There is limited insight into how analysts with different programming skills and computational reasoning abilities go about this process. Current code assistants such as Open AI's Code Interpreter [96] present these steps in natural language, Python code, code comments, and code execution outputs. It is unclear to what extent these interface affordances are helpful and how analysts use these different sources of information for accessing the correctness of AI-generated analyses.

In this work, we build on the perspective of recent works in Human-Centered Explainable AI (HCXAI) [33, 68, 81, 82] in that

users' needs to understand AI-generated outputs are dependent on their backgrounds, social, and organizational contexts; we focus on data analysts as they understand AI-assisted analyses. We study the verification workflows of analysts, how different backgrounds may influence analysts' approaches, and opportunities to improve this experience. We aim to guide the design of AI assistants that empower analysts to critically reflect on AI-generated outputs.

To explore analysts' behaviors and elicit feedback for design opportunities, we developed a prototype interface that not only includes the AI's code, code comments, and natural language explanation but also additional interactive data tables and summary visualizations (Sec. 4). This design accommodated a broad spectrum

of potential artifacts that analysts may employ during verification, facilitating the observation of naturally emerging workflows. Analysts could access intermediary data tables that let them filter, search, and sort for values to explore the data and verify the AI-generated output. Using our tool as a probe, we conducted a qualitative study with 22 professional analysts (Sec. 5). We asked participants to verify AI-generated analyses that involved sequences of data transformations covering a range of real-world datasets (Table 1). These generated analyses were in response to high-level analysis queries written by data scientists [138] which were presented to participants as prompts written by an intern.

We find analysts often started their verification workflows with *procedure-oriented* behaviors (answering “*what did the AI do?*”) and shifted to *data-oriented* behaviors (answering “*does the [resulting] data make sense?*”) once they noticed issues in the AI-generated output (Table 4). Analysts adopted procedure-oriented behaviors to get a high-level understanding of the analysis steps and confirm low-level details in the data operations. Meanwhile, analysts adopted data-oriented behaviors to make sense of the data. Notably, *data artifacts* (i.e., the data tables and summary visualizations) were often vital as secondary support in *procedure-oriented* behaviors, and vice versa for *procedure artifacts* (i.e., the natural language explanation and code/code comments) (Fig. 5).

Based on these findings (Sec. 6), we make recommendations to end-user analysts interacting with AI-based analysis assistants (Sec. 7.1). Additionally, we discuss implications for system designers (Sec. 7.2). These center around fluidly connecting data-oriented and procedure-oriented artifacts, communicating data operations and AI’s assumptions about the data, and incorporating AI guidance into verification workflows.

This paper contributes the following:

- (1) Findings from a user study using our design probe that uncover common behavioral patterns in the verification of AI-generated analyses,
- (2) A set of implications for end-user analysts interested in AI-assisted data analysis, paired with design implications for tool builders to improve verification workflows.

2 MOTIVATING EXAMPLE

To illustrate the potential errors and verification workflows in AI-assisted analyses, we compare three hypothetical analysts working on the same analysis question (Fig. 1). This scenario highlights the workflow differences between traditional and AI-assisted data analysis. Jane, Kate, and Alex are data analysts at a company that sells products on a popular e-commerce platform. They want to assess how many products may have quality control issues as the year draws to a close. Specifically, they want to find out how many distinct products bought in November were later returned. To perform this analysis, they have a dataset of product orders where each row is a unique order for a product and its associated order status. The dataset contains the following key columns:

- **order_date**: the date which the order was placed
- **sku**: a unique identifier for the product
- **quantity**: the number of units of the product ordered
- **order_status**: indicating whether the order was *Delivered to Buyer* or *Returned to Seller*

Traditional Data Analysis. Jane uses a spreadsheet for her analysis (Fig. 1A). She explores the dataset in her spreadsheet tool. After grasping its structure, she zeros in on the task. She uses the spreadsheet’s interface to filter the **order_date** column for the month of November and the **order_status** column for *Delivered to Buyer*. She sees that there are two rows left, indicating two unique products were ordered in November and returned to the seller.

In contrast, Kate opts to use computational notebooks for data analysis, employing Python code to investigate the analysis question (Fig. 1B). She writes code to read the dataset and to understand the data. She then writes code to filter the **order_date** column for the month of November and the **order_status** column for *Delivered to Buyer*. In her workflow, Kate validates her operations by writing code to get a quick sense of the intermediate data produced. Like Jane, Kate arrives at the same conclusion. Throughout this process, Jane and Kate are actively engaged in the data and operations, giving them confidence in the correctness of their conclusion.

AI-Assisted Data Analysis. Alex, meanwhile, decides to leverage an AI assistant for help. He uploads the dataset and writes a natural language prompt: *How many items purchased within the month of November were returned to seller?*

The AI assistant writes and executes code to read the dataset. Next, it performs the analysis task, writing and running code before arriving at a final answer of 5. During this, the AI assistant also explains its steps in natural language. Because Alex was not directly involved in the analysis, he needs to validate its answer by inspecting the AI-generated output (Fig. 1C). He reads over the AI’s natural language explanation of its procedure and skims the code briefly. Without fully understanding the code, he deems the answer to be correct. However, upon comparing his answer with Jane’s and Kate’s, Alex realizes the AI made a mistake. Looking through the intermediate data in Kate’s work, Alex realizes the AI assistant calculated the number of units returned rather than the number of *unique* products returned.

The errors in AI-assisted data analysis are subtle and require the analyst to carefully scrutinize the AI’s procedure and associated data involved. Additionally, current AI assistants do not facilitate the exploration of data along with the AI’s code and natural language explanation, thereby constricting analysts’ verification workflows. Further, it is unclear how analysts go about verification. Therefore, this paper seeks to understand analysts’ workflows and identify opportunities for improvement.

3 BACKGROUND AND RELATED WORK

3.1 AI-Based Tools for Data Science Execution

AI-based code assistants can enable analysts to conduct analyses as end-user programmers [70, 85]. With tools such as Github Copilot and ChatGPT, analysts’ intent specified through natural language can be more expressive and approachable than those conveyed in domain-specific programming languages [27, 41]. Given the potential of AI-supported natural language programming, there is a growing body of research in the value and usability of these tools [8, 31, 52, 79, 112, 126]. These works focus on programmer

experiences with AI-based code assistants and find common challenges in comprehending and verifying the AI-generated code [79, 112, 126].

Here, we build on prior work and study the verification challenges for data analysts as end-user programmers [70]. Compared to software engineering, data analysis is more exploratory and iterative [60, 86, 104, 110] as code is not the deliverable but the outcome of the analysis [34, 54]. Importantly, making sense of the data is an integral part of data analysis [38, 71, 104, 111]. While existing work has studied how programmers validate AI-written code for general programming tasks [9], our work observes how data analysts' *sensemaking* behaviors interplay with their verification of AI-generated analyses.

AI-based analysis assistants have also received great attention. Prior work has explored the design of analytical chatbots [44, 57, 142] such as what should the responses be from such an assistant [114] and how analysts can request, specify, and refine assistance [89]. Given the attention and promise, these assistants have been integrated into popular analysis environments such as computational notebooks [39, 103] and spreadsheets [118], and released as standalone systems [30, 125]. In contrast, we examine these tools from the perspective of supporting analysts' understanding and verification of AI-generated analyses, especially as AI-assistants can specify sequences of data operations (e.g., Fig. 1 right). In particular, we contribute the first user study examining the common behaviors and challenges in verification workflows.

3.2 Understanding and Sensemaking in Data Analysis

Prior work has found that data scientists often compare conclusions with prior analyses and existing data [66, 86]. Likewise, as part of data science collaboration and communication, it is common for data analysts to interpret and understand the results of another analyst's work [26, 67, 99, 140]. During this sensemaking, analysts often need help managing, tracking, and comparing the iterative and messy nature of their computations [43, 59, 62, 131]. To support this, prior work has explored how to help analysts track the data provenance in their analyses [58, 59, 135] and visualize how their data changes throughout various data operations [105, 129, 136].

While existing research has focused on understanding human-written analyses, there are key differences between understanding analyses written by a human, and those written by an AI. People develop diverse perceptions and levels of trust in AI systems [28, 39, 51, 64], leading to distinct behaviors when evaluating AI-generated analyses. For example, McNutt et al. [89] observed that some data scientists naturally trust analyses written by a colleague and may overlook code auditing. Additionally, social factors in data analysis communication often result in communicating asynchronously [129], employing multiple platforms (i.e., Slack, Zoom, Google Docs etc.) [99], and selectively sharing information [86, 99]. These qualities are less likely to appear when interacting with an AI-assistant. Therefore, AI analysis assistants present a unique paradigm for analysis verification and sensemaking. Given the widespread deployment of AI-based tools, our work seeks to understand the intricacies of this paradigm.

3.3 Understanding and Verifying LLM Outputs

Prior work in the Human-Centered Explainable AI community has emphasized the goal that understanding AI revolves around empowering individuals to fulfill their objectives [33, 68, 81, 82]. Following this perspective, recent work has explored ways to make LLMs more interpretable as they present their own explainability challenges [31, 120, 127]. Specifically, these works have explored opportunities for AI-code assistants in programming applications and proposed different explainable AI (XAI) features to guide the programmer's understanding of the model. In contrast, our work focuses on XAI features for data analysts. As data sensemaking is inherent in data analysis [38, 71], we explore specific features that help analysts in following the AI's data operations (e.g., via the natural language explanation and associated data tables).

We note that understanding and verifying AI-generated outputs can involve additional follow-up prompts to the AI assistant [77, 85]. In this work, we focus on understanding the usage of supporting artifacts and disentangle the verification workflow using these artifacts from prompt writing (Sec. 4.1). Works studying abstraction matching [85], prompt writing [139], and repair of AI-generated programs [6, 94] are related but distinct.

4 METHOD: DESIGN PROBE AND PREPARED TASKS

In this work, we aim to study how analysts verify AI-generated analyses with features that support data sensemaking. In doing so, we wanted to minimize the impact of variations in the dataset, task difficulty, prompt expertise, and AI responses. To facilitate analysts' natural sensemaking workflows when working with data, we chose to purpose-build a design probe [14] and provide additional data artifacts based on recommendations from prior work [5, 10, 23, 59, 61, 91, 135]. We made this decision because existing assistants, such as Code Interpreter [96], do not allow easy exploration of data and only provide natural language explanations, code, and execution outputs. In this section, we discuss the key design considerations involved in our study (Sec. 4.1) before outlining the specific design of our probe (Sec. 4.2) and the preparation of our study tasks (Sec. 4.3).

4.1 Design Considerations

In building our design probe and formulating our user study, we identified three central design considerations that involved different trade-offs outlined below. These decisions are guided by our goal to make reliable observations on analysts' *verification* workflows with different *supporting artifacts*, while minimizing distractions from other aspects of the data analysis workflow.

Using familiar and approachable supporting artifacts. A crucial decision in implementing our design probe revolved around the choice of data artifacts. One option is to incorporate customized feature-rich artifacts (e.g., those in *Datamations* or *DITL* [105, 128]) and verification interfaces (e.g., incorporating visualization tools like *Voyager* [133, 134] or *PygWalker* [53] for verificaiton). Alternatively, we could opt for integrating a core set of simpler, more user-friendly, and approachable data artifacts. These two options contrast with each other in terms of *the richness of actions analysts*

can achieve with the interface and the time and effort analysts need to spend to learn to perform these actions.

Ultimately, we opted for familiar and approachable artifacts as our study focus is to observe verification workflows: we determined that the challenges tied to learning and interpreting customized visual elements would limit our ability to observe analysts' verification workflows [37, 73, 78]. Likewise, while generic artifacts and interfaces may overlook some features experienced analysts could leverage, they let us focus on analyzing analysts' workflow across a broad audience and still gather concrete feedback on the features they want.

Focus on one turn of the human-AI interaction. Another choice that shaped our probe and study was whether or not to allow follow-up prompts to the AI as part of the verification workflow. On one hand, choosing to allow follow-up prompts enables analysts to use them in their verification. However, besides interpreting the AI's current output, analysts would need to grapple with the challenge of formulating prompts [90, 139]. On the other hand, opting against follow-up prompts mitigates the impact of prompt expertise but at the cost of confining analysts' actions. In our study, we chose to zoom in on one turn of the human-AI interaction and disallow follow-up prompts. As every new prompt triggers a corresponding response from the AI assistant, continuous interpretation and verification of AI-generated outputs is inevitable. In reducing the scope, we chose to thoroughly observe how analysts understand the AI's outputs and interact with supporting artifacts. Similarly, aiding analysts in understanding the AI-generated output through artifacts not only helps them rectify errors in immediate interactions but also craft more effective prompts in future engagements.

Prepare standardized task materials. The choice of datasets and verification tasks was also important to consider. One option is to allow analysts to work with their own datasets while interacting with the AI in real-time to verify the AI's outputs. Alternatively, we could pre-prepare the natural language prompts, associated datasets, and AI-generated responses. These choices underscore a trade-off between matching analysts' natural workflows and mitigating variability associated with generative AI and dataset complexity. In our study, to control potential confounding factors, we chose to prepare all task materials. Although this approach may differ from the scenario where analysts verify their own datasets and analyses, it closely aligns with situations where analysts need to familiarize themselves with another analyst's analysis [72, 99, 140]. To mitigate concerns related to dataset unfamiliarity, we carefully prepared approachable and realistic datasets from diverse domains, used prompts written by real-world data scientists, and provided additional background descriptions of the dataset and analysis motivation (Sec. 4.3). Given our chosen approach, it would be interesting to apply the lessons learned in this study to analysts working on their own datasets and tasks (Sec. 7.3).

4.2 Design of the Probe

Data analysts need to *inspect* the data (i.e., the data quality, format, columns, values etc.) and *engage* with the data (e.g., understanding relationships between columns, looking for outliers etc.) as part of data sensemaking activities [71]. Therefore, we designed our

probe with facilities for understanding the data. This includes data tables, and summaries and histograms of all the fields in the data. To support sensemaking while making components approachable, our probe incorporates abstractions of modalities found in popular data science tools: the code, code comments, natural language explanations, data tables, and visualizations [1–3, 29, 32, 63, 135].

User Interface. The probe's interface follows a multi-panel design. The main panel shows the chat interface between the user and the assistant (Fig. 2 left). The user's prompt and subsequent response from the assistant are shown. In the assistant's response, following Code Interpreter [96] and explanation practices recommended and found favorable in prior work [32, 77, 120], the interface shows the assistant's natural language explanation of its procedure, associated code, code comments, and execution outputs (Fig. 2A and B). These are interleaved together such that all artifacts are available at any time. In addition, because understanding the sequence of data manipulations is vital to understanding the output [105, 136], the probe includes descriptions of any intermediate data computed as a result of an atomic data wrangling operation (Fig. 2C) [56, 136].

Inspired by prior work detailing the importance of contextualized intermediate data artifacts for analysis provenance [5, 10, 23, 59, 61, 91, 135], the probe includes a side data panel (Fig. 2 right) that allows analysts to view all relevant data in the AI-generated analysis. Having this as a side panel is helpful as it suggests the information is of global scope [89]. Additionally, by keeping the data separate from the AI-generated analysis, the probe minimizes any interference to analysts following the AI's procedure in the main panel. In the side panel, data tables are shown (i.e., the original data and the results of intermediate calculations) as collapsible panels. Each collapsible panel includes a raw data table and a summary visualization of the table from [93]. The summary visualization offers a quick visual summary of the overall table, the values and distribution of each column, and basic descriptive statistics.

User Interactions. In the main panel, the probe's interactions prioritize easy access to the full/high level abstraction of the procedure conveyed through the AI's natural language explanation. Because code and table descriptions can obstruct and interrupt this view, the code and intermediate data descriptions in the main panel are initially hidden which users can reveal via a show/hidden button. Analysts can also access the side data panels within the flow of the AI's explanation via in-context buttons which open and close the corresponding panel (Fig. 2E1 and E2).

In the side panel, analysts can perform basic operations on each table such as sorting columns and filtering rows based on specific values in the column(s). To support easy navigation between the data tables and AI's procedure in the main panel [23, 135], clicking on a data table scrolls the main panel to the corresponding context of that data table.

4.3 Preparation of Study Tasks

To prepare realistic analysis verification tasks and AI-written errors representative of those made by state-of-the-art AI assistants, we used the ARCADE benchmark dataset from Yin et al. [138]. ARCADE features multiple rounds of natural language intent to code problems (i.e. a natural language query, dataset, and code solution)

A What is the least possible age of students having high adaptivity?

First, I will inspect the contents of the uploaded file to understand its structure and the data it contains. After that, I can determine the least possible age of students with high adaptivity. Let's start by loading and viewing the first few rows of the dataset.

B

```
# Filter the dataset for students with an "Adaptivity Level" of "High"
high_adaptivity_students =
    students_data[students_data["Adaptivity Level"] == "High"]

# Identify the minimum age from the subset of students with high adaptivity
least_age_high_adaptivity =
    high_adaptivity_students["Age"].min()

least_age_high_adaptivity
```

C

show/hide intermediate data description

The code generates the following intermediate data:

- high_adaptivity_students (students with an "Adaptivity Level" of "High")

```
high_adaptivity_students =
    students_data[students_data["Adaptivity Level"] == "High"]
```

D1

The dataset contains various attributes related to students, such as gender, age, education level, institution type, IT expertise, location, and adaptivity level.

To determine the least possible age of students with high adaptivity, I'll filter the dataset for students with an "Adaptivity Level" of "High" and then identify the minimum age from that subset.

E1

The least possible age of students with a high adaptivity level is between 11 and 15 years.

F

Original Data **D2**

Gen...	Age	Education Level	Monthly Internet Usage (GB)	Adaptivity Level	Class Dur...
Boy	21-25	University	34		
Girl	21-25	University	36		
Girl	16-20	College	41		
Girl	11-15	School	57		
Girl	16-20	School	33		
Boy	11-15	School	43		
Boy	11-15	School	25		
Boy	11-15	School	24		
Boy	16-20	College	48		
Boy	11-15	School	8		

G

high_adaptivity_students **E2**

Column	Snapshot	Missing	Mean	Median	SD
Gender	2 categories	0.0%	-	-	-
Age	5 categories	0.0%	-	-	-
Education Level	3 categories	0.0%	-	-	-
Monthly Internet Usage (GB)	1 100 112	0.0%	38	36	21
Adaptivity Level	1 category	0.0%	-	-	-
Class Duration	2 categories	0.0%	-	-	-
Institution Type	2 categories	0.0%	-	-	-
IT Student	2 categories	0.0%	-	-	-
Location	2 categories	0.0%	-	-	-

Figure 2: Probe Interface. The analyst's prompt and the assistant's response are shown in the left panel. The AI's response includes its natural language explanation (A), the code and code comments involved in its calculations (B), and a description of any intermediate data (C). The original data table and intermediate data table(s) are accessible via buttons interleaved in the AI's response (D1 and E1) and when clicked point to their corresponding data pane in the right panel (D2 and E2). These panes can also be opened directly in the right panel. In each pane, analysts can view the raw data table in the *Dataset* tab with sort and filter functionality (F). Analysts can also view a visualization showing the distribution and basic descriptive statistics of each column in the *Summary* tab (G).

situated in computational notebooks¹ written by data scientist annotators. In our task preparation, we ran queries from ARCADE on Code Interpreter [96] and carefully selected a subset of queries that contained errors, had good coverage across a range of domains and error types, and were approachable to analysts (details in the

appendix). These queries and Code Interpreter responses formed the backbone of our tasks.

In total, we prepared one tutorial task and 10 main tasks, 2 of which did not contain any errors, spanning 9 unique datasets (Table 1). For each task, we used the generated code from Code Interpreter's output to prepare intermediate data tables and table summary visualizations. To provide analysts with sufficient information, we prepared additional intermediate data tables for all

¹these notebooks used datasets uploaded to Kaggle and encompassed data wrangling and EDA operations

TID	Task	Natural Language Prompt	Error Type and Summary	PIDs
T0 [121]	Tutorial	What is the least possible age of students having high adaptivity?	<i>Data</i> —AI and took the minimum of age-range by alphanumeric order.	Everyone
T1 [15]	Amazon Orders	How many items purchased within the month of November were returned to seller?	<i>Prompt</i> —AI counted the quantity of products returned, instead of the number of unique products.	P1, P7, P15, P20
T2 [46]	Big Basket1	Show a list of the top five rated Nivea products.	<i>Data</i> —AI missed "Nivea Men" products in the data.	P1, P2, P5, P12
T3 [46]	Big Basket2	How expensive are gourmet products compared to beverage products in average? Show the value as a percentage of beverage products.	<i>Correct</i>	P1, P2, P5, P12, P22
T4 [108]	Bollywood	Show the top 5 movies with the highest percentage return on investment.	<i>Calculation</i> —AI mentioned but did not ignore rows with bad budget values. The intern did not filter for only Bollywood movies.	P2, P6, P10, P15, P21
T5 [11]	Flights1	What is the city that has the highest number of incoming flights? How many incoming flights does each airline have for that city with one week left until departure? Show the city name, airline and number of incoming flights.	<i>Calculation & Data</i> —AI performed the wrong order of filtering. It also incorrectly counted incoming flights based on booking rows rather than by unique flight codes.	P4, P8, P10, P14, P17
T6 [11]	Flights2	What is the least expensive time of the day to depart to Chennai in Economy class across all airlines? Show the time of the day and the fare price for each airline.	<i>Prompt</i> —For each airline, the AI calculated the minimum price of all flights during a time of day rather than the minimum of the average price of flights during a time of day.	P4, P8, P10, P14, P17, P20
T7 [92]	Hotels	Which hotels had a worse ranking this year than in 2021? Show the hotel name, location and the difference in ranking from last year.	<i>Data</i> —AI misinterpreted a "2021" flag column as representing the rank in 2021.	P3, P4, P6, P8, P12, P16
T8 [65]	Movie Content	What is the number of shows viewed in distinct languages within each genre as a percentage of the total number of shows within each genre? Show the genres as an index and languages as columns	<i>Calculation</i> —Inverted the percentage calculation. Summing across genres for a given language should add up to 100 but the AI's calculation resulted in summing across languages for a genre adding up to 100.	P5, P7, P11, P19, P22
T9 [117]	Netflix	Who is the actor who worked with the same director the most?	<i>Correct</i>	P6, P9-11, P13, P18, P19, P21
T10 [7]	TV	What are the top 5 most selling television frequencies? Show the frequencies with their counts	<i>Data</i> —The AI cleaned dirty frequency data but it neglected correct data that existed in other columns.	P7, P13, P20, P22

Table 1: User Study Tasks. These tasks and natural language prompts were derived from Yin et al. [138]. The prompts were written by data scientist annotators, who were instructed to be natural, concise, and avoid unnecessary elaborations. AI errors encompassed poor interpretations of the prompt and data, and incorrect calculations. Although creating more precise prompts can potentially reduce errors, our study is not centered on this aspect.

major transformations, often breaking up chained function calls in the code. We then verified with multiple expert data scientists that these intermediate data tables were sufficient for the task. To ease dataset understanding and help participants focus on verification, we reduced the number of columns for excessively complex datasets. We made sure to leave enough extra columns irrelevant to the immediate analysis query such that the task remains non-trivial. In addition, we provided a clear ground truth motivation and a more detailed description of the analysis query. In the rest of the paper, to differentiate the two, we refer to the detailed description of the analysis query as the *analysis goal*. Finally, we provided a description of the dataset and what each column in the dataset represented.

In summary, we compiled the following primary materials for each task: the natural language query from ARCADE, the unmodified code, code comments, and natural language explanation response from Code Interpreter, the prepared data tables and visualizations (Fig. 2 right), and a detailed description of the motivation, analysis goal, and dataset.

5 USER STUDY

Using our probe and prepared tasks, we conducted a lab study to observe analysts' verification workflows of AI-generated analyses. Our goal was not to evaluate our specific interface but rather to understand behaviors that emerge when verification artifacts are available. Three research questions guide our study design and analysis.

PID	Role	Exp.	Primary Analysis Tools	Analysis Frequency	Coding Comfort	Tasks Shown
P01	Data Scientist	6 Y	Code/Computational Notebooks	Daily	Very Comfortable	[T1, T2, T3]
P02	Software Engineer	4 Y	Data Visualization Tools	Daily	Very Comfortable	[T2, T3, T4]
P03	Consultant	26 Y	Spreadsheets	Daily	Somewhat Comfortable	[T7]
P04	Data Analyst	10 Y	Data Visualization Tools	Weekly	Somewhat Comfortable	[T5, T6, T7]
P05	Software Engineer	10 Y	Code/Computational Notebooks	Weekly	Very Comfortable	[T2, T3, T8]
P06	Program Manager	12 Y	Spreadsheets	Weekly	Very Comfortable	[T4, T7, T9]
P07	Program Manager	6 Y	Spreadsheets	Monthly	Neutral	[T1, T8, T10]
P08	Software Engineer	7 Y	Code/Computational Notebook	Weekly	Very Comfortable	[T5, T6, T7]
P09	Program Manager	3 Y	Spreadsheets	Daily	Neutral	[T9]
P10	Tech. Strategist	15 Y	Spreadsheets	Monthly	Neutral	[T4, T5, T6, T9]
P11	Program Manager	11 Y	Data Visualization Tools	Daily	Somewhat Comfortable	[T8, T9]
P12	Program Manager	5 Y	Spreadsheets	Daily	Neutral	[T2, T3, T7]
P13	Cloud Architect	5 Y	Spreadsheets	Daily	Somewhat Comfortable	[T9, T10]
P14	Program Manager	0.5 Y	Code/Computational Notebook	Weekly	Very Comfortable	[T5, T6]
P15	Program Manager	10 Y	Spreadsheets	Daily	Somewhat Comfortable	[T1, T4]
P16	Program Manager	15 Y	Spreadsheets	< Monthly	Never Coded Before	[T7]
P17	Software Engineer	15 Y	Code/Computational Notebook	Weekly	Very Comfortable	[T5, T6]
P18	Program Manager	7 Y	Data Visualization Tools	Daily	Somewhat Comfortable	[T9]
P19	Finance Manager	13 Y	Spreadsheets	Daily	Neutral	[T8, T9]
P20	Program Manager	3 Y	Data Visualization Tools	Daily	Somewhat Comfortable	[T1, T6, T10]
P21	Customer Manager	8 Y	Spreadsheets	Daily	Very Comfortable	[T4, T9]
P22	Architect	5 Y	Code/Computational Notebook	Weekly	Very Comfortable	[T3, T8, T10]

Table 2: Participants reported using a variety of tools to perform data analysis in their normal workflows and came from a variety of roles and teams. Over half (13/22) of the participants reported being somewhat comfortable (knowing programming basics and can write simple programs) or less with programming.

- **RQ1 - Behaviors:** What patterns of behaviors do analysts follow in a verification workflow?
- **RQ2 - Artifacts:** What artifacts do analysts use and for what purposes?
- **RQ3 - Background:** How are analysts' backgrounds reflected in their behaviors and artifact usage?

Participants. We recruited participants from a large data-driven software company and advertised an opportunity to work with AI-generated analyses. Participants were recruited via email based on data analysis interest groups and organization mailing lists. We selected participants with data analysis experience and a diversity of backgrounds, including how often they performed data analysis, the types of tools they used for analysis, and their self-reported comfort with programming (Table 2). Participants were compensated with a \$30 USD Amex gift certificate.

Study Procedure. We conducted our study remotely with participants sharing their screens over a video conferencing tool. Because our design probe was built as a web application, participants loaded the interface on their computer with their choice of browser.

The study was structured into three parts lasting approximately 60 minutes total: a tutorial phase (~10 min), an activity phase (~40 minutes) in which participants worked through the prepared tasks, and an semi-structured interview phase (~10 minutes). In the tutorial phase, we presented participants with the following scenario:

Imagine having an intern who is provided with a dataset to address an analysis problem. The enthusiastic intern decides to use an AI assistant to complete the problem, correctly uploading the relevant dataset and crafting a prompt that is their interpretation of the original analysis

problem. The AI assistant produces an output but the intern is unsure that it is correct.

We clarified that the input to the AI assistant is only the intern's prompt and the uploaded dataset. In particular, we directed analysts not to focus on the quality of the prompt, but on the AI-generated output in response to the prompt. We mention that although the code executes, there might be issues with the information the AI uses or how it interprets the intern's prompt. Next, we presented the participants with the intern's prompt and the assistant's response in our interface (Fig. 2). We asked participants to determine if the AI assistant's answer correctly addressed the initial problem, and if not, point to reasons why they think there was a mistake. To eliminate the challenge of participants having to craft their own prompts—a task that presents its own set of difficulties [139]—we chose to assign the responsibility of prompt authoring to the intern. This choice also dissociated participants' personal preferences for how they would write prompts which is not the core focus of our study.

After introducing the analysis scenario, we gave participants a brief walk-through of the interface (Fig. 2). We then had participants complete a fixed tutorial task to familiarize themselves with the interface (Task 0 in Table 1). The task intentionally contained an error in which the AI assistant misinterpreted the values of a column and poorly sorted the answer. This strategy aimed to make participants cognizant of potential assistant errors and to encourage vigilance in subsequent tasks.

Next, in the activity phase, participants read the analysis goal and reviewed the work of the assistant and intern. We assigned tasks to participants to ensure that, collectively, all tasks were

Label	Definition
Start	Start of a verification workflow
Data Only	Using data artifacts only
Data + Procedure	Mainly focusing on the data and using procedure artifacts as secondary support
Procedure Only	Using procedure artifacts only
Procedure + Data	Mainly focusing on the procedure and using data artifacts as secondary support
Notice	Noticing an issue that may affect the correctness of the AI-generated result
Confirm	Confirming or rejecting that the AI-generated analysis has an error

Table 3: Definition of labels used to annotate participants' verification workflows. **Data Only** and **Data + Procedure** are behaviors under *data-oriented* behaviors while **Procedure Only** and **Procedure + Data** are behaviors within *procedure-oriented* behaviors.

covered by the participants (Table 1). For all tasks in the study, the motivation, analysis goal, and description of the dataset were always available in a left side panel (see appendix for an example). To make sure the participants fully understood the analysis goal, we discussed any areas they found unclear. We also clarified any questions about the dataset or analysis goal while participants were working through the task. To reduce time pressure and encourage a realistic workflow, we emphasized the exploratory nature of our study, allowing participants to work at their own pace until they were satisfied with their conclusion. After completing each task, participants were asked to complete a brief survey that asked for a confirmation as to whether the AI's generation was erroneous and their reasoning.

While participants were completing the tasks, we encouraged them to think aloud and describe their reasons for interacting with different components of the interface. If they remained relatively silent, we regularly prompted them to speak about their workflow. We provided minimal help beyond clarifying the analysis goal or semantics of the data.

Finally, in the semi-structured interview phase, we asked open-ended questions to understand participants' backgrounds and prior experiences, overall verification workflows, and rationales behind their behaviors.

Transcript and Workflow Analysis. To identify recurring themes, two authors reviewed the participants' screen recordings, engaged in several rounds of open coding, and had multiple meetings to refine their observations. The themes we observed highlighted variations in participants' workflows and the artifacts they used. In the analysis and results, we distinguish between *data artifacts* (those in Fig. 2 right) and *procedure artifacts* (those in Fig. 2 left). Data artifacts pertain to the original, intermediate, and result data (i.e., the data table and summary visualization). Conversely, procedure artifacts encompass the natural language explanation, code/comments, and the AI's explanation and interpretation of the result.

Additionally, to systematically understand participants' verification workflows, we delineated these into sequences of primary

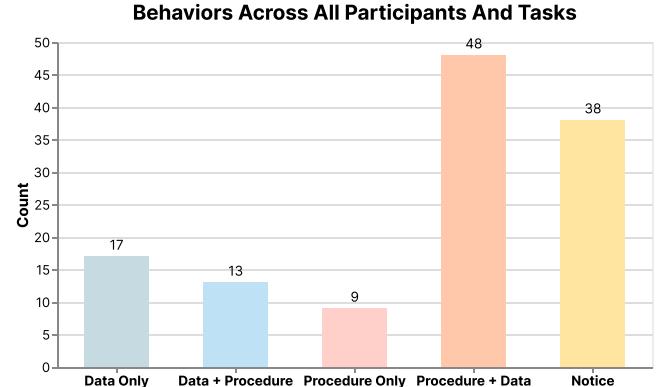


Figure 3: Participants often followed procedure-oriented behaviors.

observations. Informed by the themes from open coding, the two authors who developed the themes drafted an initial set of observation labels (e.g., *read data table* or *raising a consideration*). Both authors annotated a few participants' workflows independently before meeting to resolve ambiguities and unclear interpretations. These discussions facilitated a refinement of the observation labels by broadening their scope. Using the final refined set of labels (Table 3), they rewatched the recordings and annotated all end-to-end verification workflows in the study (see Fig. 4 for examples), including observations of the artifacts being used. Each workflow starts when participants began verifying the AI-generated analysis (after understanding the task and data) and ends when they confirmed or rejected the existence of an error. We list all participants' verification workflows in the appendix.

6 RESULTS

Participants engaged in a variety of different steps to identify potential issues and ultimately confirm or reject the AI's analysis. These were guided by two high-level behaviors participants had within a verification workflow: *procedure-oriented behaviors* and *data-oriented behaviors* (Fig. 3).

Within a verification workflow, participants with procedure-oriented behaviors were primarily focused on validating the AI's procedure, possibly drawing upon data artifacts to aid their understanding. In contrast, participants exhibiting data-oriented behaviors gave precedence to scrutinizing the data involved in the analysis, relegating the AI's procedure to a supplementary role. The distinction between these behaviors lies in the participants' intention. Participants following procedure-oriented behaviors were concerned with *what does the AI do?* Meanwhile participants following data-oriented behaviors focused on *does the data make sense?* This was clear from observing their interaction patterns and verbal comments. For example, while some participants engaged extensively with intermediate datasets, their main goal in those moments was to confirm the correctness of the AI's procedure; this behavior was labeled as a procedure-oriented behavior with the use of data artifacts as secondary support (i.e., **Procedure + Data**). In this section, we provide a detailed account of findings pertinent to our research questions.

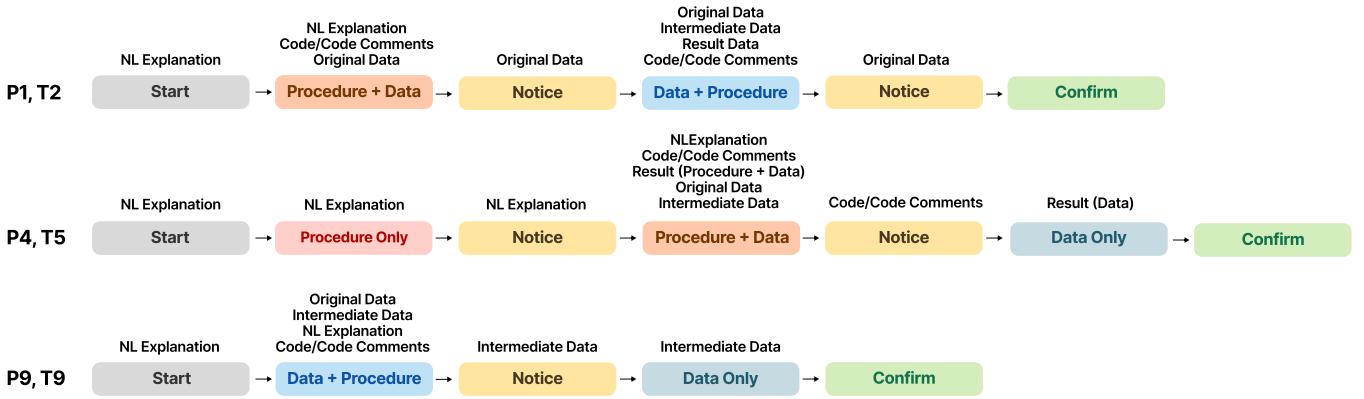


Figure 4: Examples of end-to-end verification workflows. We show a sample of interesting end-to-end verification workflows and associated artifacts using our labels in Table 3. The labels help get a sense of the overall workflow and capture relevant behavioral patterns. For example, in task (T5), we observed P4 starting out (**Start**) focusing only on the natural language explanation (**Procedure Only**) before noticing an issue (**Notice**) from the explanation. Their behavior then shifts slightly as they include data artifacts (**Procedure + Data**) before noticing a subsequent issue in the code (**Notice**). Finally, they check the result data (**Data Only**) to confirm an error in the AI’s analysis (**Confirm**). Overall, we observed 52 verification workflows in our study (39 of which involved errors) with an average length of 4.40 (std=1.42) labels. Four of these had two **Notice** patterns occur, 30 had one **Notice** pattern occur and the rest had none.

6.1 What patterns of behaviors do analysts follow in a verification workflow?

Overall, we observed most participants (18/22) followed all or parts of the AI’s procedure at some point during their verification workflows (44/52). In workflows in which participants did not follow the AI’s procedure, they only used the analysis goal, intern’s prompt, and data (original and intermediate) artifacts. Likewise, most participants (18/22) had data-oriented behaviors in their verification workflows (25/52). Participants generally exhibited different behaviors at different moments in their workflows to focus on different parts of the AI-generated response.

6.1.1 Analysts initially gravitate towards procedure-oriented behaviors. Participants usually had procedure-oriented behaviors initially during their verification workflows (41/52). These behaviors were relatively time-consuming and similar to when programmers thoroughly examine the code’s logic when they validate suggestions from an AI programming assistant. However, in contrast to the programming assistant setting, procedure-oriented behaviors were rarely as brief as the quick verification methods (e.g., visual inspection and pattern matching) programmers most commonly employ [9, 80].

6.1.2 Analysts shift towards data-oriented behaviors once they noticed an issue. Upon identifying a concern, participants’ strategies diversified, delving deeper to explore and validate the perceived issue (Table 4). Of the 31 times in which participants noticed an issue but had not yet confirmed an error, 17 times (15/22 participants) they continued their verification workflow with data-oriented behaviors. In particular, 13 of 17 of these occurrence were when participants switched from procedure-oriented behaviors to data-oriented behaviors after noticing an issue.

Patterns	Count	Occurrences in Verification Workflows
PD – N – D	9	T1 [P1], T2 [P5, P12], T4 [P2], T5 [P4], T6 [P14], T8 [P22], T9 [P6], T10 [P22]
PD – N – PD	6	T3 [P5, P12], T6 [P4, P14], T7 [P12], T10 [P20]
PD – N – C	5	T1 [P7], T2 [P2], T4 [P6], T5 [P10, P14]
PD – N – DP	3	T2 [P1], T4 [P21], T10 [P13]
P – N – PD	3	T5 [P4, P8], T8 [P11]
DP – N – D	3	T4 [P15], T7 [P16], T9 [P9]
DP – N – C	2	T2 [P1], T10 [P13]
P – N – P	2	T7 [P4, P6]

Table 4: Participants’ behaviors around noticing an issue. We report patterns around noticing an issue that occurred more than once in our study. Participants often followed procedure-oriented behaviors before noticing an issue and used a variety of data-oriented behaviors and procedure-oriented behaviors afterwards to validate their concerns.

Some participants (P2, P15, P16, P20, P22) identified the source of the issue through eyeballing different data tables, while others had data-oriented behaviors to check more closely the potential source of an issue (P4, P5). In T2, P5 correctly noticed that there was no consideration for *Nivea Men*. To explore this further, they focused on the original data table, filtering for rows in which the brand was *Nivea* and applying additional filters to spot check specific products [T2]. Similarly, to affirm their suspicions of an error, P4 investigated the final result table after noticing the AI may not have grouped rows by *flight code* (Fig. 4 second row) [T5].

6.1.3 Analysts adopt procedure-oriented behaviors to get a high-level understanding of the analysis steps. Many participants looked over the AI’s procedure to get a high-level understanding of the

analysis steps (P1, P2, P4, P8, P11, P17, P18, P20). For instance, in T9, P18 wanted to understand how the AI arrived at a particular actor-director pair in the result: “*I’m trying to read the code and I’m trying to understand ... like how is it generating that pair and if there is any error in that process.*” Sometimes, participants gained confidence in the AI’s correctness from this high-level understanding (P1, P2, P11, P20). For example, P1 confirmed their understanding after reading the AI’s procedure one last time: “*seems reasonable to me.*” P11 was reassured by the AI’s overall procedure matching their own: “*the steps it described is exactly what I would have done but in SQL.*” In other cases, participants’ focus on the procedure helped them notice and subsequently confirm a particular step the AI did wrong (P4, P8). For example, P4 and P8 both noticed from the NL explanation that the AI’s order of analysis steps were incorrect [T5]. In T6, similar to P11, P4 broke down the general steps on their own and realized the AI missed calculating the average after inspecting the AI’s procedure.

6.1.4 Analysts adopt procedure-oriented behaviors to confirm details in the data operations. Participants also focused on lower-level details involved in the AI’s data operations (P2, P7, P8, P9, P10, P11, P13, P15, P17, P22). Participants checked functions used in the code, logic around the data operations, and specific result values. P13 was especially vigilant making sure that “*the spelling was correct*” in the data columns mentioned in the code, and that “*syntactically ... (they) appear to be correct.*” P19 was also following the AI’s data operations to see how it handled unusual values the AI had identified earlier in its procedure. Focusing on low-level details was one way participants gained confidence in the AI or noticed issues (P7, P8, P10, P11). For instance, in T9, P10 focused on the AI’s data operations in the code which helped them conclude the AI had indeed found the highest count. P11, while not the most comfortable with Python, used code comments to translate the AI’s procedure into familiar data operations, noticing the AI’s procedure was inconsistent with the analysis goal description [T8].

However, participants sometimes formed incorrect conclusion due to their reliance on the AI’s procedure (P15, P17). In T5, P17 was initially skeptical of the result table due to insufficient information presented, but gained confidence that the result “*looked more or less correct*” after cross referencing the AI’s NL explanation and code with the data. P15 manually reviewed the original data and used the AI’s NL explanation and code to form an incorrect expectation of the output. They confirmed that the AI was correct when the result data table matched their expectation [T1].

6.1.5 Analysts adopt data-oriented behaviors to better understand the structure of and relationships in the data. Participants in data-oriented behaviors focused on the data in a variety of ways. Some participants (P3, P7, P15, P19) were keen on understanding the data (original and intermediate) and the specific columns that were pertinent to the analysis goal. For instance, P7 adopted data-oriented behaviors using only data tables at the beginning of their process, exploring the column of interest in the analysis goal and subsequently identifying a data quality issue in the frequency column [T10]. Meanwhile, for P9, a prototypical data-oriented participant, their data-oriented behaviors involved comparing all data tables [T9]. Specifically, they delved into the directors and actors intermediate data tables to gather an expectation of what they would

expect in the subsequent merged data table (Fig. 4 third row). After, they checked if specific actors in the result data existed in the merged data. In the process, they spent significant time forming an expectation of the result data, transitioning back and forth between all data tables.

6.2 What artifacts do analysts use and for what purposes?

Across all participants, both data and procedure artifacts were used consistently in their verification workflow (Fig. 5). Data artifacts were frequently employed to support procedure-oriented behaviors (17/22 participants and in 41/52 workflows), while procedure artifacts were utilized during data-oriented behaviors (10/22 participants and in 13/52 workflows). Within the data, participants engaged with the intermediate data roughly equally as often as the original data and result data. Similarly, for procedure artifacts, participants leveraged code and code comments as often as the natural language explanation. However, these preferences for artifacts were not consistent between participants, which we discuss in section 6.3.

6.2.1 Analysts use data artifacts for sensemaking. Unsurprisingly, most participants leveraged data artifacts to understand and interpret the data, a standard practice in data analysis practices [5, 10, 38]. For instance, P7 used the data summary visualizations to understand the data, and expressed liking “*the summary stats to be able to do a quick view of the data and understand what it is instead of having to manually check everything.*” P3, meanwhile, meticulously examined the original, intermediate, and result data to have a comprehensive understanding of the analysis. Consequently, several participants expressed appreciation for the data table summary and visualizations provided in the AI’s procedure (P5, P7, P13, P17, P20).

Through closely examining the data tables, participants were able to identify issues with the AI-generated analyses (P1, P3, P7, P9, P13, P15, P16). For instance, from the data tables, P1 noticed that there was also the *Nivea Men* brand in the dataset which the AI missed (Fig. 4 first row), ultimately leading them to confirm that the AI-generated analysis contained an error. Likewise, for P9, their exploration of the data tables led them to perceive a discrepancy with the number of rows in merged data table and the count from a quick mental calculation. These observations align with established guidelines in data analysis practices and implementations of data analysis tools, and underscore the importance of data visualizations alongside the analysis procedure [5, 10, 23, 59, 61, 91, 135].

6.2.2 Analysts frequently use data artifacts for both quick checks and detailed calculations to support procedure-oriented behaviors. Some participants glanced at the data artifacts to quickly check things they noticed in the AI’s procedure (P6, P8, P22). For instance, P22 noticed that the AI’s calculation for the average price of gourmet food was not a number, prompting them to check the original data table and discover the AI used the wrong column name [T2]. Similarly, participants used data artifacts to conduct data sanity checks to verify the procedure (P1, P3, P4, P5, P19, P20, P21, P22). For instance, P19 wanted to see if percentages in a column added up to 100: “*If the percentages tied to 100%, I would feel confident just at a glance that you know everything and action was allocated to*

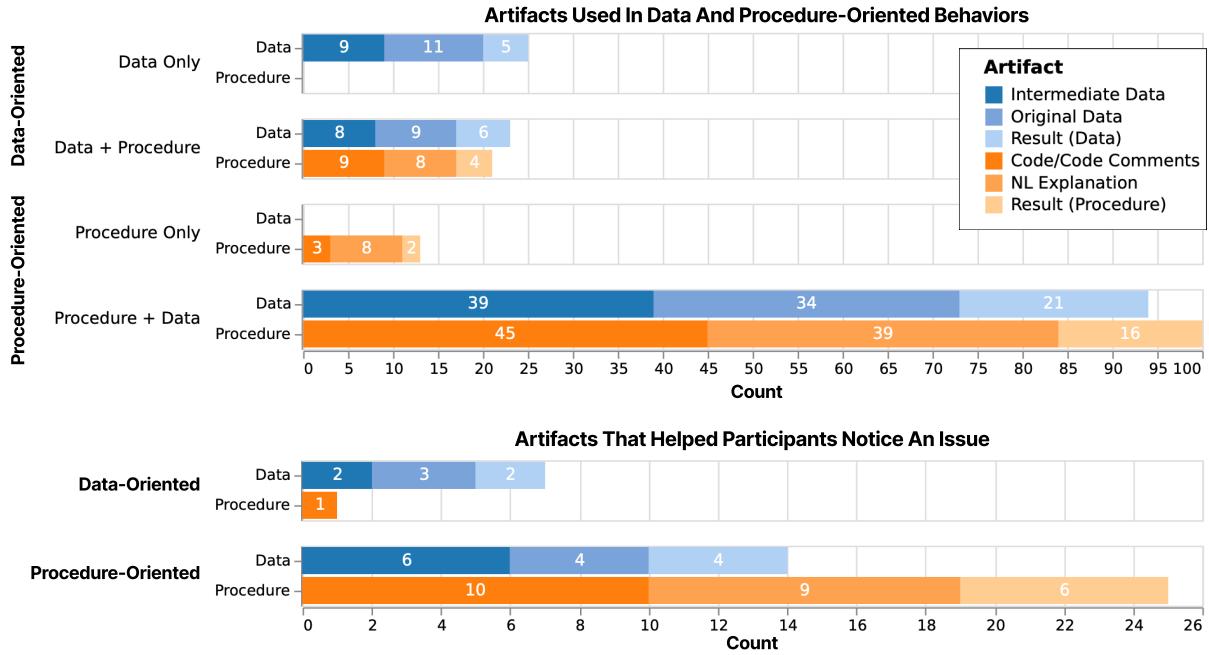


Figure 5: Both data and procedure artifacts were used to support participant's primary behaviors. For each type of behavior illustrated in Fig. 3, we tally the unique artifacts involved, counting each artifact once per occurrence in a behavior. This distribution shows that participants extensively used data artifacts to support procedure-oriented behaviors and used procedure artifacts to support data-oriented behaviors (Top). The intermediate data, original, data, code, and natural language explanation were all pivotal for analysts' to notice an issue (Bottom).

one of those languages” [T8]. P20, meanwhile, went back and forth between the code and table to verify the procedure, stating, “just like understanding what the code did and then each of the intermediate steps in seeing like ... it looks like it joined it” [T6].

Alternatively, participants also tried to replicate the AI’s data operations using the supported functionalities in the data tables to compare with the result in the AI’s procedure and verify what the AI did (P2, P9, P12, P14, P22). For example, P22 filtered on the frequency column following the AI’s steps to see how many rows were filtered out [T10].

These approaches align with common AI-assisted programming verification methods [9, 80], resembling visual inspection and pattern matching for quick data checks, and code execution for replicating the AI’s data operations. However, distinct from AI-assisted programming, data (in addition the procedure) serves as a central artifact for verification.

6.2.3 Analysts use procedure artifacts to find discrepancies in the procedure. Some participants were keen on verifying the consistency between the high-level analysis steps in the NL explanation and the specific data operations in the code (P5, P11, P17, P22). Meanwhile, other participants compared the AI’s Natural Language (NL) explanation with their own common knowledge. For example, P10 felt that “2800 flights felt like a lot” for a day [T5], leading them to investigate the AI’s procedure. Similarly, from inspecting the AI’s explanation of the result, P6 noticed “Jurassic World is not a Bollywood movie” [T4]. These discrepancies prompted them to

question the AI’s data operations and understanding of the dataset. Moreover, participants used procedure artifacts to verify the AI’s interpretation of the prompt and original dataset (P4, P12). As a result, in T7, both P4 and P12 noticed issues with the AI’s understanding of the data, stating an incorrect interpretation of a column.

6.2.4 Analysts use procedure artifacts for tracing data provenance to support data-oriented behaviors. Many participants (P1, P3, P13, P15, P16) used the NL explanation and code to trace the provenance of data. P1 in T2 is a typical example (Fig. 4 first row). P1 started out with procedure-oriented behaviors but upon perceiving an issue in the original data table where more than 5 items had a rating of 5, transitioned into data-oriented behaviors. They scrutinized the original data table but became confused when the result data table displayed values less than 5. This contradicted their initial observation where more than five products had a 5 rating in the original data. To resolve this, they turned to the intermediate data and the AI’s procedure, tracing the steps in the code that derived the result table.

Participants also noticed issues when using procedure artifacts to support their understanding of the provenance of the data (P1, P9, P13, P15, P16, P17, P21). For instance, P17 wanted to explore why the intermediate data table contained all kinds of departure time. They checked the code to see where it did the filtering on the *Departure Time* and realized the AI missed taking into account the time of the day when calculating the minimum of prices [T6]. These behaviors reiterate the importance of tracking data provenance in analysis

tools and highlight opportunities to adopt and adapt methods in existing implementations [58, 62, 101, 105, 109, 128, 135].

6.3 How are analysts' backgrounds reflected in their behaviors and artifact usage?

In the post-task interviews, we explored participants' prior experiences with data analysis, coding, and existing analysis tools. We noted a range of experiences that might be associated with participants' behaviors and interactions with artifacts.

6.3.1 Prior Experience With Data. Some participants with data-oriented behaviors mentioned their prior experience working with data (P6, P9, P11, P16, P17, P18, P22). For instance, P15, who had previous experience working with large datasets, expressed comfort with directly manipulating the datasets, stating, “*I'm used to working with hundreds of thousands of rows, so the size of the spreadsheet wouldn't have been a problem.*” In both tasks P15 did, they only followed data-oriented behaviors.

These prior analysis experiences also led participants to be vigilant about the data (P6, P9, P11, P22). P6, who self-identified as a “*data guy*”, was inclined to dive directly into the data and scrutinize the tables and numbers. P11, an electrical engineer by training, also favored a data-oriented approach. They compared the study tasks to their past experiences designing systems, explaining how they always “*went straight to the data*” and asked “*hey, can I trust that it's quality data?*” P9, with a background in finance and accounting, was accustomed to handling datasets that were “*routinely between 100 and 500,000 rows*.” In the study, P9 focused on validating the data, underscoring the importance of ensuring data integrity and rational calculations. Similarly, P22 cited their prior work experiences as informing their analysis verification approach: “*we were maniacal about not drawing the wrong conclusion.*” During their verification workflows, they demonstrated a methodical approach to problem-solving, routinely considering “*lots of caveats and checks.*”

Moreover, we noticed participants' prior experiences forming conclusions and insights from data shaped their behavior in the study tasks. Participants (P1, P6, P9, P19, P22) who started their verification workflow by proactively devising a plan mentioned being accustomed to formulating error hypotheses, setting expectations, and actively identifying problems in the data. For example, P19 commented how in their own data analysis workflow they were always on the lookout for “*weird stuffs*” like data classes, data types, and anomalies. This was reflected in their systematic approach when faced with our study tasks.

6.3.2 Prior Experience With Code. While participants' experiences working with data sometimes led them to being more data-oriented, a couple participants' coding comfort made them inclined towards procedure-oriented behaviors and examining the code (P5, P8). For instance, P8, a software engineer, naturally tried to verify the correctness of the code: “*as a developer, just let me look at the code.*” Similarly, P5, compared the tasks to doing a “*code review.*”

Interestingly, participants who were not familiar or comfortable with Python but had a familiarity of data operations and/or data-oriented programming languages (e.g., SQL and Visual Basics for Applications) managed to leverage their knowledge to understand the AI's procedure on the data (P10, P15, P19, P20, P21). For example,

P4 looked for key words in the code to understand the the AI's operations. For P10 in T9, despite having no prior experience with Python, they were able to devise the analysis steps in SQL and cross-check them against the AI's explanation and code comments:

“If I were to do this in SQL Server with the table that you gave me, I would have created ... select unique ID where director and popped the table out ... So basically, I'm translating Python into the technologies I know to see if the AI is doing what I need them to do.”

However, while able to parse key words or main points from the code, a lack of understanding of the code hindered some participants' confidence (P2, P12, P14, P18, P20, P21). For example, P20 found T5 “*difficult because (they didn't) fully understand the code.*” For P18, an incomplete comprehension in T9 led them to focus more on the original data as opposed to the intermediate data since they were unsure how the intermediate data was generated: “*in the intermediate steps, I am not confident if the slicing and the dicing is being done the way it should be.*” To bridge this gap, a couple participants mentioned liking code documentation or being able to easily translate the code to a more familiar languages (P11, P21). For P12 and P21 in particular, a lack of code comprehension led them to prefer the NL explanations over the code. P12 found the explanations to be “*a very concrete and precise way that makes sense.*”

6.3.3 Prior Experience With Existing Analysis Tools. Our study aimed to include participants with a diverse usage of existing analysis tools (Table 2). While there were ways to verify the AI-generated output using our design probe (Fig. 2), some participants expressed a preference for functionality they were familiar with from existing tools that were unsupported by our interface (P12, P13, P15, P18, P21). For example, P12 mentioned wanting to use conditional sorting on the columns, and P17 wanted support for comparison operators in data filtering. Several participants also expressed a desire to have pivot tables (P11, P17, P18, P21), making it easier to understand the data. Moreover, while many participants expressed appreciation for the visual summaries of the data tables (P5, P7, P13, P17, P20), some participants (P4, P18, P21) expressed low trust in the data artifacts because they were unsure how the artifacts were created in the first place. P21 in particular stated:

“The (visualization) summary tab was kind of useless to me... I didn't really trust it and and it was unclear to me how the the data was being summarized.”

7 DISCUSSION

In this work, we examined how data analysts understand and verify AI-generated analyses. We developed a design probe, prepared realistic analysis queries and AI-generated analyses, and conducted a user study observing analysts' verification workflows and common behaviors. In this section, we synthesize the results from our user study and share implications for data analysts using AI-based analysis tools (Sec. 7.1) and system designers building these tools (Sec. 7.2).

7.1 Implications for Data Analysts

7.1.1 The Role of Data Literacy in Analysis Verification. Our findings suggest that while a lack of familiarity with one particular programming language (i.e., Python) is not a barrier for verifying AI-generated analyses, prior experience with data operations in other languages (e.g., SQL or Visual Basics for Applications) can aid in understanding the AI's procedure (Sec. 6.3.2). Likewise, knowledge and experience in data analysis are often indispensable for grasping AI-generated analyses, mirroring the requirement for computational thinking skills when working with AI-based code assistants [112]. Here, we highlight two recommendations for analysts to be more effective in verifying AI-generated analyses.

Know Common Data Operations. We observed participants closely examined the AI's data operations and sometimes performed these operations on the datasets directly to compare with AI's result (Sec. 6.1.4 and 6.2.2). Their actions involved understanding common operations such as filtering, sorting, and merging the data. Similarly, understanding aggregation operations, such as sum, average, count, were essential to verify how the AI grouped data and performed calculations on those groups.

Develop Strategies for Quick Data Validations. Our observations revealed analysts often employed quick sanity checks on the data to assess AI-generated outputs (Sec. 6.2.1 and 6.2.2). Such strategies involved eyeballing values, manual summation of rows, using visualizations to check the value distributions, spot checking result values against original data table, and comparing things against common knowledge. These were effective for analysts to either build confidence in the results or identify unexpected patterns. The sanity checks often involved forming expectations around the result and then comparing them with the AI-generated output. Any discrepancies signaled potential issues that demanded further inspection. Given the effectiveness, analysts should actively identify quick sanity checks and externalize their expectations [111]. However, as there may be data issues hidden by seemingly reasonable patterns [25], analysts should employ a variety of different sanity checks.

7.1.2 The Role of AI Assistants in Data Analysis. Participants' varied experiences working with data and analysis tools (Sec. 6.3) led to a wide range of comments about their preferred use of general AI-based analysis assistants. Here, we discuss how AI assistants could be used in data analysis.

Some participants commented how AI-assistance would be helpful for getting them a partial answer that might guide them in their analysis (P1, P6, P7, P14, P20, P21). For instance, P6 mentioned how the AI can help them consider alternative steps: "*It'll help me say 'oh maybe I didn't think of this way of solving this and I can now go down that path.'*" AI assistants can help inspire ideas for analysis steps similar to when programmers follow an exploration mode in AI-assisted programming [9]. Presenting alternative approaches is especially critical in analysis assistance as analysts can often be limited in their consideration of analysis approaches [84, 86], thereby affecting the robustness of their subsequent analysis conclusions [16, 113, 116]. Thus, AI analysis assistance can improve analysis quality by broadening the analysis decision space analysts consider [40, 86].

Similarly, participants noted that assistants can expedite their analysis workflow (P1, P13, P20), corroborating the findings from prior research on AI-code assistants [31, 52, 126, 143]. However, to ensure the robustness of analysis conclusions, it is crucial to balance the acceleration of analysis with measures that surface potential errors, underlying data assumptions, and alternative approaches [39].

Finally, some participants expressed a preference to reserve AI assistance for peripheral tasks such as error-checking or report generation rather than the central analysis. Utilizing AI assistance for crafting reports [130, 141] and similar subsidiary activities offers a low stakes environment for analysts to evaluate the quality of AI-generated outputs within data analysis workflows. Such use cases can also facilitate a deeper understanding of the assistant's capabilities and limitations.

7.1.3 Understanding the Bounds of the AI and the Tool. Participants sometimes expressed doubt on the intermediate data artifacts as it was unclear what artifacts had the potential to be a result of AI "hallucinations" [47] and which were consistent with the data and procedure (Sec. 6.3.3). As the boundary blurs between AI-generated artifacts and tool-generated artifacts, analysts should clearly distinguish what is and what is not produced by AI.

Tool artifacts compiled from code is deterministic (i.e., the same summary visualization will be produced given the same code and input data) [112]. The AI-generated outputs and their associated errors are stochastic in nature, varying each time the AI is prompted to conduct the analysis. Knowing the provenance of artifacts used to support verification—what is tool-generated (e.g., the visual summary of data) and AI-generated (e.g., the intermediate data feeding into the visualization)—is important to assess the reliability of the information associated with these artifacts. Additionally, understanding the strengths and limitations of the AI [106, 120] is crucial to knowing what types of mistakes to look for.

7.2 Implications for System Designers

Informed by our study findings of verification workflows and noted challenges, we highlight four design implications for designers of AI-based analysis assistants.

7.2.1 Fluidly connecting data-oriented and procedure-oriented artifacts at various levels of granularity. Our study on analysts' verification workflows indicated the use of data-oriented and procedure-oriented artifacts was often intimately tied to each other, requiring analysts to switch between the two (e.g., intermediate data table and the code snippet that generated this data). Therefore, tools should facilitate a seamless connection between the data and procedure. Integrating the data with the procedure aligns with existing implementations in data science and machine learning tools [12, 23, 63, 107, 135]. Likewise, as analysts prefer different functionality based on their prior experiences (Sec. 6.3.3), data components should be interactive, reusable, and customizable to cater to specific preferences and applications [12, 23, 107].

These connections should also prioritize verification behaviors. Participants often explored AI-generated analyses at different levels of abstraction (Sec. 6.1.3 and 6.1.4) and transitioned between artifacts for quick glances of specific information (e.g., column

names) or in-depth examinations involving multiple steps (Sec. 6.2.2 and 6.2.4). Therefore, tools should allow analysts to specify the scope of their current verification actions and surface the most relevant information. For example, when an analyst selects the text of a column name or cell value in the procedure, the tool can quickly highlight the corresponding data (and vice-versa). If the analyst selects a code snippet containing multiple data operations, the tool can highlight the corresponding data differences between the operations [128]. Furthermore, to support analysts in tracing the data provenance of a given intermediate data (Sec. 6.2.4), the tool can highlight just the data operations in the procedure that produced it [43, 58]. To support these features on the back-end, the AI can dynamically augment its text response to include semantically relevant connections to the data. Recent systems have demonstrated the efficacy of augmenting the AI's output for AI-assisted writing and exploration [49, 74].

7.2.2 Communicating the semantics of data operations. We observed AI-generated code to be a common part of analysts' verification workflows in procedure-oriented and data-oriented behaviors (Sec. 6.2.3 and 6.2.4). However, a lack of comprehension of the code and the exact data operations impacted analysts' confidence in their verification findings and even pushed them to alternative behaviors (Sec. 6.3.2). Nevertheless, participants who were able to translate the AI's code into familiar data operations were able to understand the procedure and not be significantly impacted. Therefore, consistent with implications from prior work [39, 68], tools should provide multiple methods to communicate the operations in the AI's procedure.

For instance, participants suggested translating the Python code into programming languages they were comfortable with. This translation can be achieved with neural code translators [75] or existing AI-code assistants [36, 95] at the code snippet level. If translated at the operation level, it is essential to ensure faithfulness to the original operator. Alternatively, the code can be translated into systematic and consistent natural language utterances representing the data operations. On a limited set of operations and data problems, Liu et al. [85] found this approach improved users' understanding of the code-generating model. Additionally, besides code and natural language, tools can communicate data operations through visualizations such as those in *Datamations* [105], and *SOMNUS* [136]. Likewise, tools can show differences in intermediate data between calculations to further enhance analysts' understanding of the operations [128].

7.2.3 Communicating AI's assumptions and interpretations of the data. We observed participants often noticed errors faster in tasks where the AI clearly stated its assumptions about the data (e.g., the data column semantics, assumed data types, and relationships between data columns etc.) and its interpretation of the prompt (P6, P10, P12, P22). For example, in T7, the AI clearly stated its interpretation of what the data in key columns represented: “*2021: (Assuming) Rank of the hotel in 2021*” and “*Past_rank: Past rank (unclear for which year)*.” However, we noticed similar assumptions were not always stated in other tasks.

Therefore, tools should consistently communicate the assumptions and interpretations of the dataset and analysis, a well-regarded best practice for data analysis [35]. This implication narrows down

a broader recommendation for AI-based code assistants to articulate the generated code within the context of a programmer's specific task and environment [112]. One strategy to facilitate regular communication could be to prompt the AI system to enumerate its assumptions explicitly. Another avenue, as exemplified by LLMs such as GPT-4 being good evaluators for tasks like natural language summarization and visualization [30, 83, 87], can involve employing a separate agent to evaluate the assistant-generated analysis, thereby listing any assumptions, interpretations, or discrepancies.

7.2.4 Incorporating AI guidance into verification workflows. In a world where AI can generate data analyses, analysis verification becomes an increasingly important skill [122]. In our study, we noticed variations in participants' verification approaches, with more experienced participants proactively formulating systematic plans (Sec. 6.3.1). As LLMs show promise for making data analysis approachable to broader audiences, there is an opportunity for tools to guide analysts in their verification workflows.

For example, to help with quick data checks (Sec. 6.2.2), LLMs can generate test cases. For instance, if a field represents age, then only values between 0 and around 120 might be appropriate. Furthermore, identifying missing values (which might be 0) might help in the analysis process. However, different than prior work where generated test cases were for general programming problems [21, 123], test cases for data analyses need to consider the semantics and domain of the data. Specifically, as outputs from data analysis are meant for human understanding and involve complex data structures (e.g., data tables, plots), there can be alternative ways to present the answer (e.g., different views of the same data table) [45, 76, 138]. Likewise, since AI-generated analyses can involve sequences of data operations, it is pertinent to create test cases for both the final output and intermediate steps.

Additionally, as a key part of verification involves checking for inconsistencies (Sec. 6.2.3), our findings suggest opportunities for AI self-reflection [50, 102, 115, 119] to help analysts identify potential discrepancies, formulate specific checklists, and improve their data literacy (Sec. 7.1.1). This reflection should be contextualized with external knowledge of existing best analysis practices and the contents of the working analysis. In incorporating the AI's self-reflection, the interface can visually highlight specific sections of the analysis that require verification. Such visual warnings can help reduce analysts' overconfidence in the AI's procedures [74]. However, given the failure modes of LLMs [13, 47], it is crucial to communicate to the analyst the parts of guidance that is AI generated.

7.3 Limitations and Future Work

We note several limitations in our study and discuss opportunities for future work.

First, the analysis task queries involved in the study tasks primarily involved sequences of data transformations. Other aspects of data analysis (e.g., data visualization, statistical modeling, machine learning, etc.) were not explored. However, our study still provided insight into analysis queries in common analysis workflows, as participants (P9, P22) mentioned the issues they encountered in the study tasks were ones they commonly faced in their own data

analyses. Our study is a first step to uncovering verification workflows of AI-generated outputs where data and manipulations on data are centrally involved. Future work should explore analysts' verification of AI-generated data visualizations, statistical analyses, and machine learning pipelines [26].

Second, since our study contained a small number of verification tasks while using real responses generated by state-of-the-art nondeterministic models, components of the AI's explanation (e.g., listing its assumptions of the data and columns) and specific data operations involved in the data analysis task (i.e., filtering for data or merging data) were not controlled for across tasks. Future work could experiment with how varying components of the AI's explanation or analysis task impact analysts' verification processes.

Third, as discussed in our study design considerations (Sec. 4.1) participants were not the ones gathering the data and coming up with the analysis goal. As a result, prior to them verifying the AI-generated analyses, they first spent time understanding the data and analysis goal, a process that may not be as pertinent in conducting their own data analysis with AI assistants. However, all tasks used in the study involved real-world datasets with analysis prompts written by data scientists (Sec. 5). In addition, we quickly clarified any questions regarding the analysis goal to facilitate analysts' verification. Future work could explore verification workflows when analysts work with an AI assistant on their own data and analysis.

Finally, study participants were not responsible for creating the AI prompts (Sec. 4.1). While more elaborate prompts may be argued for reducing the errors presented in our study, writing these prompts is challenging [90, 139]. Brevity of the prompt can be in tension with a precise one that reflects all aspects of the analysis [138]. Therefore, given that the misalignment between the analyst's intent and AI's actions is inherent in natural language communication [85, 88, 98], subtle AI-generated analysis errors (at least those due to misalignment) and the need for verification workflows persist. In a few cases, our participants (P6, P10, P22) noticed issues directly from language ambiguities in the intern's prompt. As a logical next step, future research should study how expertise in prompt creation influence analysts' interactions with AI assistants and how these analysts integrate prompts into their verification and repair strategies.

8 CONCLUSION

In this paper, we explore data analysts' behaviors when verifying AI-generated analyses. In a controlled user study with a design probe and prepared tasks, we find analysts frequently interleaved *procedure-oriented* and *data-oriented* behaviors, using data artifacts in support of procedure-oriented behaviors and procedure artifacts in support of their data-oriented behaviors. We also find analysts' prior analysis and programming experiences influenced their approach and understanding of the AI's output. Based on our findings, we synthesize recommendations for data analysts and highlight opportunities for future tools to enhance the verification experience, including ways to incorporate AI guidance in verification workflows.

ACKNOWLEDGMENTS

We thank analysts from Microsoft for their eager participation in our study and anonymous reviewers for their insightful comments. We also thank Bongshin Lee, Dan Marshall, Dave Brown, Robert DeLine, Priyan Vaithilingam and others from Microsoft Research for their valuable feedback throughout the project.

REFERENCES

- [1] 2023. *JupyterLab*. <https://jupyterlab.readthedocs.io/en/stable/> Accessed: 2023-09-02.
- [2] 2023. *RStudio: Integrated Development for R*. <https://www.rstudio.com/> Accessed: 2023-09-02.
- [3] 2023. *Tableau Software*. <https://www.tableau.com/> Accessed: 2023-09-02.
- [4] Alexander A. Aarts, Joanna E. Anderson, Christopher J. Anderson, Peter Raymond Attridge, Angela S. Attwood, et al. 2015. Estimating the reproducibility of psychological science. *Science* 349 (2015). <https://api.semanticscholar.org/CorpusID:218065162>
- [5] Sara Alspaugh, Nava Zokaei, Andrea Liu, Cindy Jin, and Marti A. Hearst. 2019. Fuzting and Moseying: Interviews with Professional Data Analysts on Exploration Practices. *IEEE Transactions on Visualization and Computer Graphics* 25 (2019), 22–31. <https://api.semanticscholar.org/CorpusID:52068378>
- [6] Zahra Ashktorab, Mohit Jain, Qingzi Vera Liao, and Justin D. Weisz. 2019. Resilient Chatbots: Repair Strategy Preferences for Conversational Breakdowns. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (2019). <https://api.semanticscholar.org/CorpusID:85503944>
- [7] Md Waquar Azam. 2022. TELEVISION DATASET 2022. Kaggle. [https://www.kaggle.com/datasets/mdwaquarazam/](https://www.kaggle.com/datasets/mdwaquarazam)
- [8] Monya Baker. 2016. 1,500 scientists lift the lid on reproducibility. *Nature* 533 (2016), 452–454.
- [9] Shraddha Barke, Michael B. James, and Nadia Polikarpova. 2022. Grounded Copilot: How Programmers Interact with Code-Generating Models. *Proceedings of the ACM on Programming Languages* 7 (2022), 85 – 111.
- [10] Andrea Batch and Niklas Elmquist. 2018. The Interactive Visualization Gap in Initial Exploratory Data Analysis. *IEEE Transactions on Visualization and Computer Graphics* 24 (2018), 278–287. <https://api.semanticscholar.org/CorpusID:963684>
- [11] Shubham Bathwal. 2022. Flight Price Prediction. Kaggle. <https://www.kaggle.com/datasets/shubhambathwal/flight-price-prediction>
- [12] Alex Bauerle, Ángel Alexander Cabrera, Fred Hohman, Megan Maher Welsh, David Koski, et al. 2022. Symphony: Composing Interactive Interfaces for Machine Learning. *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (2022). <https://api.semanticscholar.org/CorpusID:246996743>
- [13] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (2021). <https://api.semanticscholar.org/CorpusID:232040593>
- [14] Kirsten Boehner, Janet Vertesi, Phoebe Sengers, and Paul Dourish. 2007. How HCI interprets the probes. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1077–1086.
- [15] Pranali Bose. 2022. Amazon Seller - Order Status Prediction. Kaggle. <https://www.kaggle.com/datasets/pranalibose/amazon-seller-order-status-prediction>
- [16] Nate Brezna, Eike Mark Rinke, Alexander Wuttke, Muna Adem, Jule Adriaens, et al. 2022. Observing many researchers using the same data and hypothesis reveals a hidden universe of uncertainty. *Proceedings of the National Academy of Sciences of the United States of America* 119 (2022).
- [17] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, et al. 2020. Language Models are Few-Shot Learners. *ArXiv abs/2005.14165* (2020). <https://api.semanticscholar.org/CorpusID:218971783>
- [18] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, John A. Gehrke, Eric Horvitz, et al. 2023. Sparks of Artificial General Intelligence: Early experiments with GPT-4. *ArXiv abs/2303.12712* (2023).
- [19] Shubham Chandel, Colin B. Clement, Guillermo Serrato, and Neel Sundaresan. 2022. Training and Evaluating a Jupyter Notebook Data Science Assistant. *ArXiv abs/2201.12901* (2022). <https://api.semanticscholar.org/CorpusID:246430316>
- [20] Souti Chattopadhyay, I. V. R. K. V. Prasad, Austin Z. Henley, Anita Sarma, and Titus Barik. 2020. What's Wrong with Computational Notebooks? Pain Points, Needs, and Design Opportunities. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (2020). <https://api.semanticscholar.org/CorpusID:210927488>
- [21] Bei Chen, Fengji Zhang, A. Nguyen, Daoguang Zan, Zeqi Lin, et al. 2022. CodeT: Code Generation with Generated Tests. *ArXiv abs/2207.10397* (2022). <https://api.semanticscholar.org/CorpusID:250920542>
- [22] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, et al. 2021. Evaluating Large Language Models Trained on Code. *ArXiv abs/2107.03374* (2021).

- [23] Frederick Choi, Sajjadur Rahman, Han Jun Kim, and Daz Zhang. 2023. Towards Transparent, Reusable, and Customizable Data Science in Computational Notebooks. *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems* (2023). <https://api.semanticscholar.org/CorpusID:257687372>
- [24] Aankanksha Chowdhery, Sharai Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, et al. 2022. PaLM: Scaling Language Modeling with Pathways. *ArXiv* abs/2204.02311 (2022).
- [25] Michael Correll, Mingwei Li, Gordon Kindlmann, and Carlos Scheidegger. 2018. Looks good to me: Visualizations as sanity checks. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 830–839.
- [26] Anamaria Crisan, Brittany Fiore-Gartland, and Melanie K. Tory. 2020. Passing the Data Baton : A Retrospective Analysis on Data Science Work and Workers. *IEEE Transactions on Visualization and Computer Graphics* 27 (2020), 1860–1870. <https://api.semanticscholar.org/CorpusID:222351819>
- [27] Robert DeLine. 2021. Glinda: Supporting Data Science with Live Programming, GUIs and a Domain-specific Language. *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (2021). <https://api.semanticscholar.org/CorpusID:233987681>
- [28] Michael Ann Devito, Jeremy P. Birnholtz, Jeffrey T. Hancock, Megan French, and Sunny Xun Liu. 2018. How People Form Folk Theories of Social Media Feeds and What it Means for How We Study Self-Presentation. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (2018). <https://api.semanticscholar.org/CorpusID:5048366>
- [29] Jacob Diamond-Reivich. 2020. Mito: Edit a Spreadsheet. Generate Production Ready Python.. In *LIVE: Workshop on Live Programming*.
- [30] Victor C. Dibia. 2023. LIDA: A Tool for Automatic Generation of Grammar-Agnostic Visualizations and Infographics using Large Language Models. *ArXiv* abs/2303.02927 (2023).
- [31] Victor C. Dibia, Adam Fourney, Gagan Bansal, Forough Pourabz-Sangdeh, Han Liu, et al. 2022. Aligning Offline Metrics and Human Judgments of Value of AI-Pair Programmers. *ArXiv* abs/2210.16494 (2022). <https://api.semanticscholar.org/CorpusID:253237523>
- [32] 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. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (2020). <https://api.semanticscholar.org/CorpusID:212684638>
- [33] Upol Ehsan and Mark O. Riedl. 2021. Explainability Pitfalls: Beyond Dark Patterns in Explainable AI. *ArXiv* abs/2109.12480 (2021). <https://api.semanticscholar.org/CorpusID:237940863>
- [34] Will Epperson, Yi Wang, Robert DeLine, and Steven Mark Drucker. 2022. Strategies for Reuse and Sharing among Data Scientists in Software Teams. *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (2022), 243–252. <https://api.semanticscholar.org/CorpusID:248726301>
- [35] Andy P. Field, Jeremy Miles, and Zoë Field. 2012. Discovering statistics using R, 1st Edition. <https://api.semanticscholar.org/CorpusID:45575760>
- [36] GitHub. 2022. GitHub Copilot. <https://github.com/features/copilot>. Accessed: Sept 12, 2023.
- [37] Lars Grammel, Melanie Tory, and Margaret-Anne D. Storey. 2010. How Information Visualization Novices Construct Visualizations. *IEEE transactions on visualization and computer graphics* 16 6 (2010), 943–52. <https://api.semanticscholar.org/CorpusID:8568910>
- [38] Garrett Grolemund and Hadley Wickham. 2014. A Cognitive Interpretation of Data Analysis. *International Statistical Review* 82 (2014). <https://api.semanticscholar.org/CorpusID:53622653>
- [39] Ken Gu, Madeleine Grunde-McLaughlin, Andrew M. McNutt, Jeffrey Heer, and Tim Althoff. 2023. How Do Data Analysts Respond to AI Assistance? A Wizard-of-Oz Study. *ArXiv* abs/2309.10108 (2023). <https://api.semanticscholar.org/CorpusID:262054482>
- [40] Ken Gu, Eunice Jun, and Tim Althoff. 2022. Understanding and Supporting Debugging Workflows in Multiverse Analysis. *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (2022). <https://api.semanticscholar.org/CorpusID:252780673>
- [41] Sumit Gulwani and Mark Marron. 2014. NLyze: interactive programming by natural language for spreadsheet data analysis and manipulation. *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data* (2014). <https://api.semanticscholar.org/CorpusID:13004424>
- [42] Keiran Hardy and Alana Maurushat. 2017. Opening up government data for Big Data analysis and public benefit. *Comput. Law Secur. Rev.* 33 (2017), 30–37. <https://api.semanticscholar.org/CorpusID:63875487>
- [43] Andrew Head, Fred Hohman, Titus Barik, Steven Mark Drucker, and Robert DeLine. 2019. Managing Messes in Computational Notebooks. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (2019).
- [44] Gan Keng Hoon, Loo Ji Yong, and Goh Kau Yang. 2019. Interfacing Chatbot with Data Retrieval and Analytics Queries for Decision Making. *Lecture Notes in Mechanical Engineering* (2019). <https://api.semanticscholar.org/CorpusID:198329911>
- [45] Junjie Huang, Chenglong Wang, Jipeng Zhang, Cong Yan, Haotian Cui, et al. 2022. Execution-based Evaluation for Data Science Code Generation Models. *ArXiv* abs/2211.09374 (2022). <https://api.semanticscholar.org/CorpusID:253581341>
- [46] Suraj Jha. 2022. BigBasket Entire Product List (28K datapoints). Kaggle. <https://www.kaggle.com/datasets/surajjha101/bigbasket-entire-product-list-28k-datapoints>
- [47] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, et al. 2022. Survey of Hallucination in Natural Language Generation. *Comput. Surveys* 55 (2022), 1 – 38. <https://api.semanticscholar.org/CorpusID:246652372>
- [48] Ellen Jiang, Edwin Toh, Alejandra Molina, Kristen Olson, Claire Kayacik, et al. 2022. Discovering the Syntax and Strategies of Natural Language Programming with Generative Language Models. *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (2022). <https://api.semanticscholar.org/CorpusID:248419806>
- [49] Peiling Jiang, Jude Rayan, Steven W. Dow, and Haijun Xia. 2023. Graphologue: Exploring Large Language Model Responses with Interactive Diagrams. *ArXiv* abs/2305.11473 (2023). <https://api.semanticscholar.org/CorpusID:258823121>
- [50] Shuyang Jiang, Yuhao Wang, and Yu Wang. 2023. SelfEvolve: A Code Evolution Framework via Large Language Models. *ArXiv* abs/2306.02907 (2023). <https://api.semanticscholar.org/CorpusID:259076266>
- [51] Jiuyun Jung, Sihang Qiu, Alessandro Bozzon, and Ujwal Gadiraju. 2022. Great Chain of Agents: The Role of Metaphorical Representation of Agents in Conversational Crowdsourcing. *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (2022). <https://api.semanticscholar.org/CorpusID:248419779>
- [52] Eirini Kalliamvakou. 2022. *Research: Quantifying GitHub Copilot's Impact on Developer Productivity and Happiness*. <https://github.blog/2022-09-07-research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/> Accessed on November 16, 2023.
- [53] Kanaries. 2023. PygWalker. <https://kanaries.net/home/pygwalker>. Accessed December 1, 2023.
- [54] Sean Kandel, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. 2012. Enterprise Data Analysis and Visualization: An Interview Study. *IEEE Transactions on Visualization and Computer Graphics* 18 (2012), 2917–2926.
- [55] Eser Kandogan, Aruna D. Balakrishnan, Eben M. Haber, and Jeffrey S. Pierce. 2014. From Data to Insight: Work Practices of Analysts in the Enterprise. *IEEE Computer Graphics and Applications* 34 (2014), 42–50. <https://api.semanticscholar.org/CorpusID:6438612>
- [56] Stephen Kasica, Charles Berret, and Tamara Munzner. 2020. Table Scraps: An Actionable Framework for Multi-Table Data Wrangling From An Artifact Study of Computational Journalism. *IEEE Transactions on Visualization and Computer Graphics* 27 (2020), 957–966. <https://api.semanticscholar.org/CorpusID:221516111>
- [57] Jan-Frederik Kassel and Michael Rohs. 2018. Valletto: A Multimodal Interface for Ubiquitous Visual Analytics. *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems* (2018). <https://api.semanticscholar.org/CorpusID:5083557>
- [58] Mary Beth Kery, Amber Horvath, and Brad A. Myers. 2017. Variolite: Supporting Exploratory Programming by Data Scientists. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (2017). <https://api.semanticscholar.org/CorpusID:2174858>
- [59] 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. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (2019). <https://api.semanticscholar.org/CorpusID:140210955>
- [60] Mary Beth Kery and Brad A. Myers. 2017. Exploring exploratory programming. *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (2017), 25–29. <https://api.semanticscholar.org/CorpusID:21574188>
- [61] Mary Beth Kery and Brad A. Myers. 2018. Interactions for Untangling Messy History in a Computational Notebook. *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (2018), 147–155. <https://api.semanticscholar.org/CorpusID:53080227>
- [62] 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. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (2018). <https://api.semanticscholar.org/CorpusID:5060661>
- [63] Mary Beth Kery, Donghao Ren, Fred Hohman, Dominik Moritz, Kanit Wongsuphasawat, et al. 2020. mage: Fluid Moves Between Code and Graphical Work in Computational Notebooks. *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (2020). <https://api.semanticscholar.org/CorpusID:221836345>
- [64] Pranav Khadpe, Ranjay Krishna, Fei-Fei Li, Jeffrey T. Hancock, and Michael S. Bernstein. 2020. Conceptual Metaphors Impact Perceptions of Human-AI Collaboration. *Proceedings of the ACM on Human-Computer Interaction* 4 (2020), 1 – 26. <https://api.semanticscholar.org/CorpusID:221005643>

- [65] Owais Khan. 2022. R.I.S.E. – Research. Innovate. Solve. copilot. Kaggle. <https://www.kaggle.com/datasets/owaikhan9654/rise-research-innovate-solve-copilot>
- [66] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. 2016. The Emerging Role of Data Scientists on Software Development Teams. *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)* (2016), 96–107. <https://api.semanticscholar.org/CorpusID:7977224>
- [67] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. 2018. Data Scientists in Software Teams: State of the Art and Challenges. *IEEE Transactions on Software Engineering* 44 (2018), 1024–1038. <https://api.semanticscholar.org/CorpusID:53280229>
- [68] Sunnie S. Y. Kim, Elizabeth Anne Watkins, Olga Russakovsky, Ruth C. Fong, and A. Monroy-Hernández. 2022. "Help Me Help the AI": Understanding How Explainability Can Support Human-AI Interaction. *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (2023). <https://api.semanticscholar.org/CorpusID:252780815>
- [69] Gary Klein, Jennifer K. Phillips, Erica Rall, and Deborah A. Peluso. 2007. A Data-Frame Theory of Sensemaking.
- [70] Amy J. Ko, Robin Abraham, Laura Beckwith, Alan F. Blackwell, Margaret M. Burnett, et al. 2011. The state of the art in end-user software engineering. *ACM Computing Surveys (CSUR)* 43 (2011), 1 – 44. <https://api.semanticscholar.org/CorpusID:9435548>
- [71] Laura M. Koesten, Kathleen Gregory, Paul T. Groth, and Elena Paslaru Bontas Simperl. 2019. Talking datasets: Understanding data sensemaking behaviours. *Int. J. Hum. Comput. Stud.* 146 (2019), 102562. <https://api.semanticscholar.org/CorpusID:208176144>
- [72] Laura M. Koesten, Emilia Kacprzak, Jeni Tennison, and Elena Paslaru Bontas Simperl. 2019. Collaborative Practices with Structured Data: Do Tools Support What Users Need? *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (2019). <https://api.semanticscholar.org/CorpusID:140240430>
- [73] Bum Chul Kwon, Brian D. Fisher, and Ji Soo Yi. 2011. Visual analytic roadblocks for novice investigators. *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)* (2011), 3–11. <https://api.semanticscholar.org/CorpusID:8617319>
- [74] Philippe Laban, Jesse Vig, Marti A. Hearst, Caiming Xiong, and Chien-Sheng Wu. 2023. Beyond the Chat: Executable and Verifiable Text-Editing with LLMs. *ArXiv abs/2309.15337* (2023). <https://api.semanticscholar.org/CorpusID:262938226>
- [75] Marie-Anne Lachaux, Baptiste Rozière, Lowik Chanusso, and Guillaume Lampple. 2020. Unsupervised Translation of Programming Languages. *ArXiv abs/2006.03511* (2020). <https://api.semanticscholar.org/CorpusID:219401607>
- [76] Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, et al. 2022. DS-1000: A Natural and Reliable Benchmark for Data Science Code Generation. *ArXiv abs/2211.11501* (2022).
- [77] Himabindu Lakkaraju, Dylan Slack, Yuxin Chen, Chenhao Tan, and Sameer Singh. 2022. Rethinking Explainability as a Dialogue: A Practitioner's Perspective. *ArXiv abs/2202.01875* (2022). <https://api.semanticscholar.org/CorpusID:246607834>
- [78] Sukwon Lee, Sung-Hee Kim, Ya-Hsin Hung, Heidi Lam, Youn ah Kang, et al. 2016. How do People Make Sense of Unfamiliar Visualizations?: A Grounded Model of Novice's Information Visualization Sensemaking. *IEEE Transactions on Visualization and Computer Graphics* 22 (2016), 499–508. <https://api.semanticscholar.org/CorpusID:12291419>
- [79] Jenny Liang, Chenyang Yang, and Brad A. Myers. 2023. Understanding the Usability of AI Programming Assistants. *ArXiv abs/2303.17125* (2023). <https://api.semanticscholar.org/CorpusID:257833548>
- [80] Jenny T Liang, Chenyang Yang, and Brad A. Myers. 2023. A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges. <https://api.semanticscholar.org/CorpusID:257833548>
- [81] Qingzi Vera Liao, Dan Gruen, and Sarah Miller. 2020. Questioning the AI: Informing Design Practices for Explainable AI User Experiences. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (2020). <https://api.semanticscholar.org/CorpusID:210064344>
- [82] Qingzi Vera Liao, Microsoft Research, Canada Kush, Richa Varshney, and Kush R. Varshney. 2021. Human-Centered Explainable AI (XAI): From Algorithms to User Experiences. *ArXiv abs/2110.10790* (2021). <https://api.semanticscholar.org/CorpusID:239050385>
- [83] Stephanie C. Lin, Jacob Hilton, and Owain Evans. 2022. Teaching Models to Express Their Uncertainty in Words. *Trans. Mach. Learn. Res.* 2022 (2022). <https://api.semanticscholar.org/CorpusID:249191391>
- [84] Jiali Liu, Nadia Boukhelifa, and James R. Eagan. 2020. Understanding the Role of Alternatives in Data Analysis Practices. *IEEE Transactions on Visualization and Computer Graphics* 26 (2020), 66–76.
- [85] Michael Xieyang Liu, Advait Sarkar, Carina Negreanu, Benjamin G. Zorn, J. Williams, et al. 2023. "What It Wants Me To Say": Bridging the Abstraction Gap Between End-User Programmers and Code-Generating Large Language Models. *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (2023). <https://api.semanticscholar.org/CorpusID:258107840>
- [86] Yang Liu, Tim Althoff, and Jeffrey Heer. 2019. Paths Explored, Paths Omitted, Paths Obscured: Decision Points & Selective Reporting in End-to-End Data Analysis. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (2019).
- [87] Yang Liu, Dan Iter, Yichong Xu, Shuo Wang, Ruochen Xu, et al. 2023. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. *ArXiv abs/2303.16634* (2023). <https://api.semanticscholar.org/CorpusID:257804696>
- [88] Ewa Luger and Abigail Sellen. 2016. "Like Having a Really Bad PA": The Gulf between User Expectation and Experience of Conversational Agents. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (2016). <https://api.semanticscholar.org/CorpusID:1036498>
- [89] Andrew M Mcnutt, Chenglong Wang, Robert DeLine, and Steven Mark Drucker. 2023. On the Design of AI-powered Code Assistants for Notebooks. *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (2023). <https://api.semanticscholar.org/CorpusID:26274637>
- [90] Swaroop Mishra and Elnaz Nour. 2022. HELP ME THINK: A Simple Prompting Strategy for Non-experts to Create Customized Content with Models. *ArXiv abs/2208.08232* (2022).
- [91] Dominik Moritz, Danyel Fisher, Bolin Ding, and Chi Wang. 2017. Trust, but Verify: Optimistic Visualizations of Approximate Queries for Exploring Big Data. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (2017). <https://api.semanticscholar.org/CorpusID:15419853>
- [92] Chadner Navarro. 2022. Travel+Leisure World's Best Hotels 2022. Kaggle. <https://www.kaggle.com/datasets/narmelan/travelleisure-worlds-best-hotels-2022>
- [93] ObservableHQ. 2023. Summary Table. <https://observablehq.com/@observablehq/summary-table> Accessed: July 30, 2023.
- [94] Theo X. Olaussen, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. 2023. Demystifying GPT Self-Repair for Code Generation. *ArXiv abs/2306.09896* (2023). <https://api.semanticscholar.org/CorpusID:259187989>
- [95] OpenAI. 2022. ChatGPT: Conversational AI Language Model. <https://chat.openai.com>. Accessed on June 1, 2023.
- [96] OpenAI. 2023. Chat with GPT-4 Code Interpreter. <https://chat.openai.com/?model=gpt-4-code-interpreter>. Accessed August 26, 2023.
- [97] OpenAI. 2023. GPT-4 Technical Report. *ArXiv abs/2303.08774* (2023).
- [98] Fatma Özcan, Abdul Quamar, Jaydeep Sen, Chuan Lei, and Vasilis Eftymiou. 2020. State of the Art and Open Challenges in Natural Language Interfaces to Data. *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (2020). <https://api.semanticscholar.org/CorpusID:218881987>
- [99] Rock Yuren Pang, Ruotong Wang, Joely Nelson, and Leilani Battle. 2022. How Do Data Science Workers Communicate Intermediate Results? *2022 IEEE Visualization in Data Science (VDS)* (2022), 46–54. <https://api.semanticscholar.org/CorpusID:252124317>
- [100] Raja Parasuraman and Dietrich Manzey. 2010. Complacency and Bias in Human Use of Automation: An Attentional Integration. *Human Factors: The Journal of Human Factors and Ergonomics Society* 52 (2010), 381 – 410. <https://api.semanticscholar.org/CorpusID:2279803>
- [101] Deok Gun Park, Mohamed Suhaib, Minsheng Zheng, Cody Dunne, Eric D. Ragan, et al. 2021. StoryFacets: A design study on storytelling with visualizations for collaborative data analysis. *Information Visualization* 21 (2021), 3 – 16. <https://api.semanticscholar.org/CorpusID:237102555>
- [102] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, et al. 2023. Generative Agents: Interactive Simulacra of Human Behavior. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (2023). <https://api.semanticscholar.org/CorpusID:258040990>
- [103] Chris Perry and Shrestha Basu Mallick. 2023. AI-powered coding, free of charge with Colab. <https://blog.google/technology/developers/google-colab-ai-coding-features/>
- [104] Peter Pirolli. 2007. The Sensemaking Process and Leverage Points for Analyst Technology as Identified Through Cognitive Task Analysis.
- [105] Xiaoying Pu, Sean Gross, Jake M. Hofman, and Daniel G. Goldstein. 2021. Datamations: Animated Explanations of Data Analysis Pipelines. *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (2021). <https://api.semanticscholar.org/CorpusID:233987105>
- [106] Sruti Srinivasa Srinavasa Ragavan, Zhitao Hou, Yun Wang, Andrew D. Gordon, Haidong Zhang, et al. 2022. GridBooks: Natural Language Formulas for the Spreadsheet Grid. *27th International Conference on Intelligent User Interfaces* (2022). <https://api.semanticscholar.org/CorpusID:247585151>
- [107] Sajjadur Rahman, Peter Griggs, and cCaugatay Demiralp. 2020. Leam: An Interactive System for In-situ Visual Text Analysis. *ArXiv abs/2009.03520* (2020). <https://api.semanticscholar.org/CorpusID:221534816>
- [108] Shivani Rana. 2022. Bollywood Movies Box-Office Collection 2022. Kaggle. <https://www.kaggle.com/datasets/shivaniranana63/bollywood-movies-boxoffice-collection-2022>
- [109] Adam Rule, Ian Drosos, Aurélien Tabard, and James Hollan. 2018. Aiding Collaborative Reuse of Computational Notebooks with Annotated Cell Folding. *Proceedings of the ACM on Human-Computer Interaction* 2 (2018), 1 – 12. <https://api.semanticscholar.org/CorpusID:52258235>

- [110] Adam Rule, Aurélien Tabard, and James D. Hollan. 2018. Exploration and Explanation in Computational Notebooks. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (2018). <https://api.semanticscholar.org/CorpusID:5048947>
- [111] Daniel M. Russell, Mark Stefik, Peter Pirolli, and Stuart K. Card. 1993. The cost structure of sensemaking. *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems* (1993). <https://api.semanticscholar.org/CorpusID:207177544>
- [112] Advait Sarkar, Andrew D. Gordon, Carina Negreanu, Christian Poelitz, Sruti Srinivasa Ragavan, et al. 2022. What is it like to program with artificial intelligence?. In *Annual Workshop of the Psychology of Programming Interest Group*. <https://api.semanticscholar.org/CorpusID:251554706>
- [113] Martin Schweinsberg, Michael Feldman, Nicola Staub, Olmo van den Akker, Robbie C. M. Aert, et al. 2021. Same data, different conclusions: Radical dispersion in empirical results when independent analysts operationalize and test the same hypothesis. *Organizational Behavior and Human Decision Processes* (2021).
- [114] Vidya Setlur and Melanie K. Tory. 2022. How do you Converse with an Analytical Chatbot? Revisiting Gricean Maxims for Designing Analytical Conversational Behavior. *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (2022). <https://api.semanticscholar.org/CorpusID:247054720>
- [115] Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection. *ArXiv abs/2303.11366* (2023). <https://api.semanticscholar.org/CorpusID:257636839>
- [116] Raphael Silberzahn, Eric Luis Uhlmann, Daniel P. Martin, Pasquale Anselmi, Frederik Aust, et al. 2018. Many Analysts, One Data Set: Making Transparent How Variations in Analytic Choices Affect Results. *Advances in Methods and Practices in Psychological Science* 1 (2018), 337 – 356.
- [117] Victor Soeiro. 2022. Netflix TV Shows and Movies. Kaggle. <https://www.kaggle.com/datasets/victorsoeiro/netflix-tv-shows-and-movies>
- [118] Colette Stallbaumer. 2023. Introducing Microsoft 365 Copilot—A whole new way to work. <https://www.microsoft.com/en-us/microsoft-365/blog/2023/03/16/introducing-microsoft-365-copilot-a-whole-new-way-to-work/>
- [119] Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. 2023. Cognitive Architectures for Language Agents. *ArXiv abs/2309.02427* (2023). <https://api.semanticscholar.org/CorpusID:261556862>
- [120] Jiao Sun, Qingzi Vera Liao, Michael J. Muller, Mayank Agarwal, Stephanie Houdé, et al. 2022. Investigating Explainability of Generative AI for Code through Scenario-based Design. *27th International Conference on Intelligent User Interfaces* (2022). <https://api.semanticscholar.org/CorpusID:246705915>
- [121] Md Mahmudul Hasan Suzan and Nishat Ahmed Samrin. 2022. Students Adaptability Level in Online Education. Kaggle. <https://www.kaggle.com/dmdmahmudulhasansuzan/students-adaptability-level-in-online-education>
- [122] Xinning Tu, James Y. Zou, Weijie Su, and Linjun Zhang. 2023. What Should Data Science Education Do with Large Language Models? *ArXiv abs/2307.02792* (2023). <https://api.semanticscholar.org/CorpusID:259361007>
- [123] Michele Tufano, Dawn Drain, Alexey Svyatkovskiy, Shao Kun Deng, and Neel Sundaresan. 2020. Unit Test Case Generation with Transformers and Focal Context. <https://api.semanticscholar.org/CorpusID:235165921>
- [124] Barbara Ubaldi. 2013. Open Government Data: Towards Empirical Analysis of Open Government Data Initiatives. <https://api.semanticscholar.org/CorpusID:260737241>
- [125] Unknown. 2023. Create Models and Automate Data Workflows with AI. <https://www.datagran.io>
- [126] Priyan Vaithilingam, Tianyi Zhang, and Elena L. Glassman. 2022. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. *CHI Conference on Human Factors in Computing Systems Extended Abstracts* (2022). <https://api.semanticscholar.org/CorpusID:247255943>
- [127] Helena Vasconcelos, Gagan Bansal, Adam Fourney, Qingzi Vera Liao, and Jennifer Wortman Vaughan. 2023. Generation Probabilities Are Not Enough: Exploring the Effectiveness of Uncertainty Highlighting in AI-Powered Code Completions. *ArXiv abs/2302.07248* (2023). <https://api.semanticscholar.org/CorpusID:256846746>
- [128] April Yi Wang, Will Epperson, Robert DeLine, and Steven Mark Drucker. 2022. Diff in the Loop: Supporting Data Comparison in Exploratory Data Analysis. *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (2022). <https://api.semanticscholar.org/CorpusID:248419893>
- [129] April Yi Wang, Anant Mittal, Christopher A. Brooks, and Steve Oney. 2019. How Data Scientists Use Computational Notebooks for Real-Time Collaboration. *Proceedings of the ACM on Human-Computer Interaction* 3 (2019), 1 – 30. <https://api.semanticscholar.org/CorpusID:207946488>
- [130] Fengjie Wang, Xuye Liu, Oujing Liu, Ali Neshati, Tengfei Ma, et al. 2023. Slide4N: Creating Presentation Slides from Computational Notebooks with Human-AI Collaboration. *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (2023). <https://api.semanticscholar.org/CorpusID:258216753>
- [131] Jiawei Wang, Tzu yang Kuo, Li Li, and Andreas Zeller. 2020. Assessing and Restoring Reproducibility of Jupyter Notebooks. *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (2020), 138–149. <https://api.semanticscholar.org/CorpusID:229703261>
- [132] Christopher D. Wickens, Benjamin A. Clegg, Alex Z. Vieane, and Angelia Sebok. 2015. Complacency and Automation Bias in the Use of Imperfect Automation. *Human Factors: The Journal of Human Factors and Ergonomics Society* 57 (2015), 728 – 739. <https://api.semanticscholar.org/CorpusID:12243641>
- [133] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock D. Mackinlay, Bill Howe, et al. 2016. Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations. *IEEE Transactions on Visualization and Computer Graphics* 22 (2016), 649–658. <https://api.semanticscholar.org/CorpusID:2366653>
- [134] Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, et al. 2017. Voyager 2: Augmenting Visual Analysis with Partial View Specifications. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (2017). <https://api.semanticscholar.org/CorpusID:14999239>
- [135] Yifan Wu, Joseph M. Hellerstein, and Arvind Satyanarayanan. 2020. B2: Bridging Code and Interactive Visualization in Computational Notebooks. *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (2020). <https://api.semanticscholar.org/CorpusID:221492874>
- [136] Kai Xiong, Siwei Fu, Guoming Ding, Zhongwu Luo, Rong Yu, et al. 2022. Visualizing the Scripts of Data Wrangling with SOMNUS. *IEEE Transactions on Visualization and Computer Graphics PP* (2022), 1–1. <https://api.semanticscholar.org/CorpusID:246287020>
- [137] Frank F. Xu, Bogdan Vasilescu, and Graham Neubig. 2021. In-IDE Code Generation from Natural Language: Promise and Challenges. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31 (2021), 1 – 47. <https://api.semanticscholar.org/CorpusID:231718679>
- [138] Pengcheng Yin, Wen-Ding Li, Kefan Xiao, A. Eashaan Rao, Yeming Wen, et al. 2022. Natural Language to Code Generation in Interactive Data Science Notebooks. *ArXiv abs/2212.09248* (2022). <https://api.semanticscholar.org/CorpusID:254854112>
- [139] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qiang Yang. 2023. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (2023). <https://api.semanticscholar.org/CorpusID:258217984>
- [140] Amy X. Zhang, Michael J. Muller, and Dakuo Wang. 2020. How do Data Science Workers Collaborate? Roles, Workflows, and Tools. *Proceedings of the ACM on Human-Computer Interaction* 4 (2020), 1 – 23. <https://api.semanticscholar.org/CorpusID:210839751>
- [141] 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. *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (2022). <https://api.semanticscholar.org/CorpusID:25794488>
- [142] Qiyu Zhi and Ronald A. Metoyer. 2020. GameBot: A Visualization-augmented Chatbot for Sports Game. *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (2020). <https://api.semanticscholar.org/CorpusID:216611752>
- [143] Albert Ziegler, Eirini Kalliamvakou, Shawn Simister, Ganesh Sittampalam, Alice Li, et al. 2022. Productivity assessment of neural code completion. *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming* (2022). <https://api.semanticscholar.org/CorpusID:248798468>

A TASK SELECTION PROCESS

To avoid datasets overlapping with the training data of LLMs [17, 97], we chose natural language queries associated with datasets uploaded to Kaggle after February 2022. This included 660 queries from 70 distinct datasets.

We selected a subset of analysis queries that were incorrectly answered by current state-of-the-art LLMs (GPT-4 and Code Interpreter). First, we tested the 660 queries against GPT-4 [97] using its API (temperature=0.7, n=3). For this, GPT-4 was provided a dataset summary similar to that of other LLM-based analysis tools [30], a scaffold to write code in, and the natural language query. Next, we sampled a set of 50 queries that resulted in incorrect outputs from GPT-4, specifically selecting more complex queries with a high number of Pandas API calls in the solution. These queries (and corresponding datasets) were then presented to Code Interpreter. Code Interpreter executed the code for the analysis, and detailed its steps in natural language.

After manually inspecting the 50 queries GPT-4 struggled with, we identified 22 that were also incorrectly answered by Code Interpreter. These errors included bad calculations, semantic misinterpretations of the prompt, and misunderstandings of the data. We then chose a subset of these 22 queries that cover a variety of error types (i.e., misunderstanding column semantics, bad calculations, or misunderstanding the query) and involve datasets and domains that were approachable to general audiences. Using this subset, we constructed the tutorial and primary tasks of our study. Sometimes, Code Interpreter encountered ambiguities and sought clarification from the user. For these cases, we included a follow-up prompt and integrated the multi-turn interaction into our study tasks.

A.1 Example of Dataset Summary used for GPT-4 Prompt

The following is an example of the dataset summary used as part of the prompt for GPT-4 for the Amazon Orders dataset (T1),

```
{
  "name": "",
  "file_name": "",
  "dataset_description": "",
  "fields": [
    {
      "column": "order_no",
      "properties": {
        "dtype": "string",
        "samples": [
          "171-5463316-4433940",
          "404-5515061-6165137",
          "407-6814126-3628337"
        ],
        "num_unique_values": 171,
        "semantic_type": "",
        "description": ""
      }
    },
    {
      "column": "order_date",
      "properties": {
        "dtype": "date",
        "min": "2021-06-13T19:08:00",
        "max": "2022-02-25T20:44:00",
        "samples": [
          "2022-01-27T17:31:00",
          "2021-10-15T20:27:00",
          "2021-12-08T23:54:00"
        ],
        "num_unique_values": 171,
        "semantic_type": "",
        "description": ""
      }
    }
  ],
  "field_names": [
    "order_no",
    "order_date",
    "buyer",
    "ship_city",
    "ship_state",
    "sku",
    "quantity",
    "order_status"
  ]
}
```

```

    "num_unique_values": 171,
    "semantic_type": "",
    "description": ""
  }
},
...
{
  "column": "sku",
  "properties": {
    "dtype": "category",
    "samples": [
      "SKU: CR-6E69-UXFW",
      "SKU: ST-27BR-VEMO",
      "SKU: 8V-0Q14-I63T"
    ],
    "num_unique_values": 54,
    "semantic_type": "",
    "description": ""
  }
},
{
  "column": "quantity",
  "properties": {
    "dtype": "number",
    "std": 0.44513190725972585,
    "min": 1,
    "max": 4,
    "samples": [
      4,
      3,
      1
    ],
    "num_unique_values": 4,
    "semantic_type": "",
    "description": ""
  }
},
{
  "column": "order_status",
  "properties": {
    "dtype": "category",
    "samples": [
      "Returned to seller"
    ],
    "num_unique_values": 2,
    "semantic_type": "",
    "description": ""
  }
},
{
  "field_names": [
    "order_no",
    "order_date",
    "buyer",
    "ship_city",
    "ship_state",
    "sku",
    "quantity",
    "order_status"
  ]
}
```

B EXAMPLE INTERFACE IN THE USER STUDY

An example of the interface used during the study for task 7 is shown in Fig. 6.

C PARTICIPANTS' VERIFICATION PROCESSES

We include the verification processes of all participants in Table 5.

Analysis Task **Study Scenario**

Motivation

In the competitive hospitality industry, maintaining or improving hotel rankings is vital for attracting customers and sustaining business growth. Travelers often rely on these rankings to make informed decisions about where to stay. By identifying which hotels have seen a decrease in ranking since 2021, stakeholders can pinpoint areas of concern, strategize on improvements, and make informed decisions about marketing, promotions, and other business-related initiatives.

Analysis Task

You want to identify the hotels that have seen a decline in their ranking (numerical decrease) as compared to 2021. For each of these hotels, provide details such as the hotel name, its location, and the exact drop in ranking (i.e., the difference between the current year's rank and the rank in 2021). This will give a clear picture of the hotels that need attention and the severity of their rank drop.

Dataset Description

This dataset contains the results of the [World's Best 100 Hotels in 2022](#) published by Travel+Leisure. Included are details about the hotels that made 2022's list. Each row is a hotel that appeared on 2022's list.

Columns

The dataset contains the following columns, not all of which may be necessary for the analysis task.

- Hotel: accommodation name
- Country: country of the hotel
- Region: region of the hotel based on continent, sub-region and culture
- Location: location of the hotel
- Score: hotel rating given by Travel+Leisure
- Rank: the hotel's rank in 2022
- Year: year the hotel was built/opened
- 2021: if the hotel is on 2021's top 100 list (1 means yes)
- Past_rank: past rank (unclear for which year).

View Original Data

Which hotels had a worse ranking this year than in 2021? Show the hotel name, location and the difference in ranking from last year.

To determine which hotels had a worse ranking this year compared to 2021, I'll need to:

1. Load the 2022 dataset provided.
2. Examine the structure of the dataset to understand the available columns and their meanings.
3. Identify the relevant columns that contain the hotel names, locations, and rankings.
4. Calculate the difference in rankings from last year.
5. Filter out the hotels that had a worse ranking this year compared to 2021.

Let's start by loading the data and inspecting its structure.

show/hide code +

Original Data

The dataset has been successfully loaded. Based on the columns provided, here's a brief description:

- Hotel: Name of the hotel.
- Country: Country where the hotel is located.
- Region: Geographical region of the hotel.
- Location: Location of the hotel.
- Score: Score assigned to the hotel.
- Rank: Rank of the hotel in 2022.
- Year: Year of establishment or opening
- 2021: (Assuming) Rank of the hotel in 2021.
- Past_rank: Past rank (unclear for which year).

To determine which hotels had a worse ranking this year compared to 2021, we'll compare the "Rank" column to the "2021" column. Let's filter the data to find the hotels that fit this criteria and calculate the difference in ranking.

show/hide code +

(result)

show/hide intermediate data description +

(intermediate) Worse Ranking Hotels

(intermediate) Worse Ranking Hotels With Rank Difference

Here are the hotels that had a worse ranking in 2022 compared to 2021:

Hotel	Location	Ranking Difference
Rosewood Castiglion del Bosco Montalcino	Montalcino, Italy	1

Original Data

Dataset Summary

20... : Hotel : Country : Regi... : Location : Sc... : R...

Hotel	Country	Region	Location	Score	Rank
Rosewood Castiglion del Bosco Montalcino	Italy	Europe	Montalcino	99.25	1
Grace Hotel Athens	Greece	Europe	Santorini	99.22	2
Waldorf Astoria Maldives Ithaafushi	Maldives	Southeast Asia	Ithaafushi Island	99.11	3
Pickering's Hotel New York	United States	North America	Wolfboro	98.95	4
One&Only Ocean Club Maldives	Maldives	Southeast Asia	North Male Atoll	98.93	5
Royalton Blue Bay Beach Resort & Spa	Morocco	Africa	Marrakech	98.93	5
Capella Singapore	Indonesia	Southeast Asia	Bali	98.97	7
The London Hotel New York	United States	North America	New York	98.8	8
Hôtel Meurice Paris	France	Europe	Paris	98.73	9
Rosewood Madrid	Spain	Europe	Madrid	98.67	10
The Oberoi New Delhi	India	Southeast Asia	New Delhi	98.67	10
The Oberoi Udaipur	India	Southeast Asia	Udaipur	98.58	12
Mandarin Oriental Bali	Indonesia	Southeast Asia	Bali	98.31	13
Wilderns Rwanda	Rwanda	Africa	Musanze	98.29	14
Portrait Hotel Florence	Italy	Europe	Florence	98.25	15
Raffles Istanbul	Turkey	Middle East	Istanbul	97.72	16
The Oberoi Morocco	Morocco	Africa	Marrakech	98.24	17
Capella Hanoi	Vietnam	Southeast Asia	Hanoi	98.18	18
White Swan Hotel Palm Beach	United States	North America	Palm Beach	98.13	19

(result)

(intermediate) Worse Ranking Hotels

(intermediate) Worse Ranking Hotels With Rank Difference

Figure 6: An example study task. We include the description of the analysis goal and dataset in the left side-panel.

Verification Processes	Count	Occurrences
S-PD-C	11	T4 [P10], T6 [P10, P20, P8], T9 [P10, P11, P21], T3 [P2, P22], T8 [P7], T7 [P8]
S-PD-N-D-C	7	T1 [P1], T2 [P12, P5], T4 [P2], T8 [P22], T10 [P22], T9 [P6]
S-PD-N-C	5	T5 [P10, P14], T2 [P2], T4 [P6], T1 [P7]
S-PD-N-PD-C	5	T3 [P12, P5], T7 [P12], T10 [P20], T6 [P4]
S-DP-N-D-C	3	T4 [P15], T7 [P16], T9 [P9]
S-PD-N-DP-N-C	2	T2 [P1], T10 [P13]
S-P-N-PD-C	2	T8 [P11], T5 [P8]
S-DP-C	2	T5 [P17], T1 [P20]
S-P-N-P-C	2	T7 [P4, P6]
S-DP-N-PD-C	1	T3 [P1]
S-PD-D-C	1	T9 [P13]
S-PD-N-PD-N-D-C	1	T6 [P14]
S-D-DP-C	1	T1 [P15]
S-P-N-DP-C	1	T6 [P17]
S-PD-R	1	T9 [P18]
S-D-C	1	T8 [P19]
S-DP-PD-N-P-PD-C	1	T9 [P19]
S-PD-N-DP-C	1	T4 [P21]
S-D-N-DP-R	1	T7 [P3]
S-P-N-PD-N-D-C	1	T5 [P4]
S-PD-R	1	T8 [P5]
S-D-N-PD-C	1	T10 [P7]

Table 5: All Participants' Behaviors in Their Workflows. **R** is when the participant ran out of time on the task or was unable to confirm or reject the AI-generated output has an error. In 52 verification workflows, we observed 3 such cases.