Challenges in Using Conversational AI for Data Science

Bhavya Chopra
University of California
Berkeley, CA, USA
bhavyachopra@berkeley.edu

Ananya Singha
Microsoft
Bangalore, India
ananyasingha@microsoft.com

Anna Fariha
University of Utah
Salt Lake City, UT, USA
afariha@cs.utah.edu

Sumit Gulwani
Microsoft
Redmond, WA, USA
sumitg@microsoft.com

Chris Parnin
Microsoft
Raleigh, NC, USA
chrisparnin@microsoft.com

Ashish Tiwari
Microsoft
Redmond, WA, USA
astiwar@microsoft.com

Austin Z. Henley
Carnegie Mellon University
Pittsburgh, PA, USA
azhenley@cmu.edu

Abstract

Large Language Models (LLMs) are transforming data science, offering assistance in coding, preprocessing, analysis, and decision-making. However, data scientists face significant challenges when interacting with LLM-powered agents and implementing their suggestions effectively. To explore these challenges, we conducted a mixed-methods study comprising contextual observations, semi-structured interviews (n=14), and a survey (n=114). Our findings reveal key obstacles, including difficulties in retrieving contextual data, crafting prompts for complex tasks, adapting generated code to local environments, and refining prompts iteratively. Based on these insights, we propose actionable design recommendations, such as data brushing for improved context selection and inquisitive feedback loops to enhance communication with conversational AI assistants in data science workflows.

CCS Concepts

• **Human-centered computing** → **HCI theory, concepts and models**; **Empirical studies in HCI**; • **Software and its engineering** → *Requirements analysis.*

Keywords

data science, computational notebooks, large language models

ACM Reference Format:
Bhavya Chopra, Ananya Singha, Anna Fariha, Sumit Gulwani, Chris Parnin, Ashish Tiwari, and Austin Z. Henley. 2025. Challenges in Using Conversational AI for Data Science. In *Workshop on Human-In-the-Loop Data Analytics (HILDA' 25), June 22–27, 2025, Berlin, Germany*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3736733.3736748

1 Introduction

Data scientists have traditionally relied on resources like documentation, tutorials, online courses, colleagues, Q&A forums, and online communities to develop skills and solve tasks [28, 29, 37, 38]. Such resources have been instrumental in helping them navigate the challenges of data acquisition, cleaning, wrangling, visualization, and presentation, especially for those with a non-programming background [10, 26]. Although useful, it can be time-consuming, tedious, and error-prone to rely on these resources for solving a data-science problem.

With the emergence of AI-powered chat assistants, data scientists now have access to potentially faster and more accessible resources through a chat interface. These AI-powered chatbots, like ChatGPT[1], enable users to ask questions in natural language, and get useful responses with little to no latency. For example, when prompted with "split my date column that is in MM/DD/YYYY format into three columns", ChatGPT provided usable Python code:

```
df[['Month', 'Day', 'Year']] = df['Date'].str.split('/', expand=True)
```

ChatGPT can also explain how the code works, and supports follow-up questions or changes. Evaluations of these AI tools have demonstrated over 80% accuracy on comprehensive benchmarks like MMLU—which assesses a model's knowledge across 57 diverse subjects including STEM, humanities, and law [21], and for programming benchmarks like HumanEval and MBPP [5, 11].

However, the effectiveness of these tools is dependent on data scientists successfully communicating their questions, context (overall problem, task at hand, datasets, etc.), assumptions, and domain knowledge to the AI assistant, through a back-and-forth conversation. Grice's maxims of conversation posit that a successful conversation involves content that has right amount of information, that is truthful and supported by evidence, that is relevant to the specific context, and that is presented clearly [17, 36]. But conversations go awry—either side of the conversation may be making false assumptions, there may be ambiguities, and conversations may require numerous clarifications.

The barriers to express intents are exacerbated in the context of data-science tasks. First, data scientists work with a variety of artifacts, including raw datasets, code, computational notebooks, visualizations, documentation, and machine-learning pipelines. Second, datasets are often large and normalized (spilt across multiple tables), which may not be feasible to share (e.g., due to token limits of AI tools or due to data being spread across heterogeneous sources) or to summarize the relevant portions succinctly (e.g., specifying regions of the data). Third, real-world data is messy, and suffers from quality issues; they may not strictly adhere to a schema or homogeneous format. Fourth, data-science tasks often require domain expertise and numerous assumptions (e.g., negative values in a column represent an error, 0 means missing value) and it may be laborious to ensure that the AI assistant is aware of such domain knowledge and assumptions.

In this work, we aim to understand the fundamental challenges encountered by a [human] data scientist in communicating to a conversational AI agent. In particular, we address the following research questions:

[1] https://chat.openai.com/

- *RQ1:* How do data scientists interact with ChatGPT to complete data-science tasks?
- *RQ2:* What challenges and unmet needs do data scientists face when interacting with ChatGPT?
- *RQ3:* How well do these challenges and needs generalize to the broader community of data scientists?

To answer these questions, we conducted two mixed-method need-finding studies. In the first study, we observed 14 data scientists as they completed four diverse tasks with ChatGPT's assistance (Sections 3, 4). The second study surveyed 114 data scientists to validate and generalize these findings (Section 5).

This paper makes the following contributions:

- An observational study of how data scientists engage with ChatGPT to solve common data science (DS) tasks.
- A survey study to validate and generalize the findings of the observational study with a broader sample.
- Design recommendations to improve data science tools, including providing data context preemptively, providing inquisitive feedback loops, providing support for code validation & structuring, and providing transparency about shared context & assumptions.

2 Background and Related Work

This section discusses the recent surge of AI-powered chat assistants and how they are changing data science.

Large Language Models and AI-powered chat assistants. AI-powered chat assistants are built on top of large language models (LLMs). LLMs are generative machine-learning models with billions of parameters, and are trained on a vast amount of data (text and images) to generate human-like responses and perform various language tasks. These models, based on the transformer architecture [39], interact with users through a *prompt*, a natural-language query to which the LLM can provide a natural language response. There are restrictions on the *token limit*, or the amount of information that can be sent or generated from the LLM. Tools like ChatGPT add a user interface to enable a conversation, such that the model has *context* of the conversation history.

Tools for AI-assisted data science. AI-assisted tools for data cleaning suggestions—such as Trifacta[2] [4, 20, 25], Data Wrangler[3], CoWrangler [12], AWS Glue DataBrew[4], Salesforce Einstein Discovery[5], AutoPandas [7], Auto-Suggest [40]—have existed, and LLMs bring forth new opportunities for further enhancing these tools. More recently, several commercial and open-source tools leveraging conversational LLMs have emerged—DataChat AI [3, 24], Anaconda Assistant [1], Databricks Assistant [2],

There has been a recent line of work that tries to make use of LLMs to automate data-science tasks and evaluate performance of LLMs on data-science tasks [23, 27, 33, 42]. This is a promising, but orthogonal, area of research to ours as we focus on human-in-the-loop data science where AI-assistants and human data scientists work as a team. Recent works from Gu et al. also focus on the specific aspects of planning assistance from LLMs [18] and response verification approaches data scientists engage in [19]. Our work is differentiated by (1) examining the fundamental interactions with a generic AI-assistant (ChatGPT), rather than a domain-specific AI tool embedded in a system, (2) studying professional data scientists rather than non-programmers, and (3) Liu et al. specifically investigated the effect of co-editing a prompt by splitting it into natural language steps.

[2]https://docs.trifacta.com/display/SS/Overview+of+Predictive+Transformation
[3]https://devblogs.microsoft.com/python/data-wrangler-release/
[4]https://docs.trifacta.com/display/SS/Overview+of+Predictive+Transformation
[5]https://help.salesforce.com/s/articleView?id=sf.bi_edd_prep_terminology.htm

3 Study 1: Methodology

First, we conducted task-based semi-structured interviews and observed 14 professional data scientists engaged in data-science tasks. We used ChatGPT[6], a browser-based conversational AI-assistant to supplement the use of computational notebooks and the rich ecosystem of data science tools. Powered by the GPT-3.5 series, ChatGPT belongs to the InstructGPT family and allows collaborative conversations, as opposed to providing completions. Our selection of ChatGPT as the conversational assistant for the study is motivated by its accessibility and large-scale use for Python programming [14], alongside its competence in generating Python code [31, 34].

Participants. We recruited participants from a large technology company through random sampling of employees with the job title "Data Scientist". We further screened participants based on having experience with data science, Python, computational notebooks, and ChatGPT. Participants reported a median of 5 years of experience in the domain of data science ($\mu = 5.4$, $sd = 2.8$).

Tasks. We selected four tasks performed by data scientists in the lifetime of a typical project [32]—(T1 and T2) data wrangling and pre-processing tasks, (T3) feature extraction and selection, and (T4) data visualization for insight-finding and reporting. These are based on 730 computational notebooks randomly sampled from the KGTorrent dataset [35], where 4.2%, 5.7%, and 10.8% of 2511 data transformations are tasks T1, T2, and T3 respectively. Further 40.1% notebooks contain at least one visualization (T4). We also note that T1 and T2 are transactional, one-shot tasks; whereas T3 and T4 are driven by exploration and human-intuition, making them more subjective and time-consuming. We use a sample of the *New York EMS emergency calls dataset* [13].

Study protocol. We conducted a one-hour long, task-based study with each of the 14 data scientists. Participants were presented with 4 tasks, one at a time, to be solved using Python in any computational notebook of their choice. Alongside having access to ChatGPT, they were encouraged to use any preferred tools and resources (such as web search, notebooks, Excel, integrated AI-assistants like GitHub Copilot, and so on), and think aloud while solving the tasks. The completion of any task was not a necessity. Once the participants had explored a task sufficiently, they were asked semi-structured questions to assess if ChatGPT helped them solve the task satisfactorily, and if it could have helped them with the task any better.

Analysis. We transcribed session recordings, annotated prompts, ChatGPT responses, and participant activities, and conducted thematic analysis resulting in 18 open-codes and 6 themes with an

[6]https://chat.openai.com/ (Versions: March 14, 2023; March 23, 2023)

inter-rater agreement of 0.872. Saturation occurred after 9 participants, and a subsequent member-check survey confirmed participants' agreement with our findings.

4 Study 1: Observations on Using ChatGPT for Data Science Tasks

We provide a summary of our participants' interactions with ChatGPT and then report observed obstacles and strategies used by the participants. Table 2 summarizes the themes that emerged from our analysis of the data scientists' actions and responses. We also report participants who identified with each obstacle and the corresponding prompting strategies they employed in Table 2.

Prompting behaviors. In total, we logged 111 prompts made by 14 participants. Table 1 presents an overview of the prompt distribution for each study task. We observed that participants spent 64% time on preparing prompts, 27% time on adapting the code returned by ChatGPT, and the remaining on validating the code. In raw time spent on these activities, on average, 302 seconds were spent on prompt writing, 57 seconds on adaptation and 24 seconds on validation. We also find that 37% of the total time was spent on writing the initial prompt, whereas making refinements to prompts and asking follow-up questions takes 28% of the total time.

Participants also made iterative refinements to their prompts. They sent across many follow-up prompts for performing feature selection (T3)—re-emphasizing task goals, decomposing tasks, providing additional context and examples, and requesting for tweaks to code snippets. The need to send follow-up prompts was also exacerbated for plotting visualizations (T4), where participants would go back-and-forth with ChatGPT for minor tweaks—changes in plot size, color scheme, and font size.

Table 1: Overview of the number of prompts queried per task, and average time spent in prompt-writing.

Task	# prompts	avg prompts	avg time (s)
T1–Datetime typecast	22	1.57	55
T2–Split using delimiter	19	1.36	59
T3–Feature selection	47	3.35	159
T4–Heatmap plotting	23	2.3	58

4.1 Communication Obstacles with ChatGPT

Participants faced difficulties in sharing data context, uncovering and recovering from underlying assumptions made by ChatGPT, and from misaligned expectations about the response format.

Sharing context is difficult. Since LLMs *"cannot make sense of the raw data"* (P2, P3, P7, P8, P10, P12), data scientists begin the process of prompt-writing by brainstorming what pieces of data, (such as specific rows matching a criteria, sample values with data quality issues, column names based on feature importance, etc.) and which descriptive information (such as pattern of strings in a column, number of unique values, missing values, outlier information, minimum/maximum for numeric data, etc.) will lead to satisfactory responses. Participants reported that starting with curating a prompt is a *"daunting"* task (P9) because it requires decision-making on what data context needs to be included. P2 said, *"I doubt if this will be successful, I feel like we need a little bit more of numbers, but let's see if only using the column names we can get something."*

Next, to construct the prompt, participants had to frequently write code snippets in their notebooks to gather the required context. For instance, consider a `pandas` DataFrame named `df`. Participants must write, execute, and copy the output of commands such as (1) `df.head()` to obtain a sample of rows and the data header, (2) `df.columns` or `df.dtypes` to obtain column names and their data types, (3) `df['INCIDENT_DATETIME'][0]` to obtain the first value in the column to provide as few-shot examples, etc.

Some participants implemented custom logic to extract information while P5, P7 and P8 had to leave their notebook environment to open the CSV file in Excel. This was because the participants wanted to share values from a column, and the raw output from `df.head()` could not be selected freely to be copied to the clipboard. Opening the file in Excel enabled them to select any range of cells and copy them with ease without writing additional code.

ChatGPT opaquely makes assumptions. Based on the context shared, ChatGPT made its own "mental model" of the data. Participants were often enthusiastic about the semantic capabilities of ChatGPT, and how it could infer domain knowledge about the data solely using the column names. P4 leveraged these capabilities and prompted ChatGPT to provide a data dictionary. Several participants (P5, P7, P10, P12—P14) were amazed when ChatGPT could understand the data domain sufficiently to convert the extracted `'RESPONSE_TIME'` column from `timedelta` format to seconds—which was the most appropriate unit of measurement, since emergency medical services are likely to be dispatched quickly. However, despite the semantic abilities of the LLM, participants struggled and had to re-assess ChatGPT's understanding of the task and the data.

As part of T2 (split using delimiter) few rows in the parent column did not contain the zipcode, leading it to be a mix of two heterogeneous formats: (1) "`<borough>; <area_code>; <zipcode>`" ("Brooklyn; K7; 11211"), and (2) "`<borough>; <area_code>`" ("Manhattan; M3"). Participants typically did not discover this data-formatting inconsistency by themselves, since encountering such issues requires exploratory data analysis and on-ramping. As a result, ChatGPT often did not get visibility into the data quality issue through participant prompts (P1, P2, P5, P11), and generated non-executable code under the assumption of having clean and consistent data:

```
>>> df['Borough'] = df['DESC'].str.split(';').str[0]
>>> df['Precinct'] = df['DESC'].str.split(';').str[1]
>>> df['ZipCode'] = df['DESC'].str.split(';').str[2]
IndexError: list index out of range
```

In T3 (feature selection) and T4 (heatmap plotting), ChatGPT often made assumptions about the data characteristics and type of columns solely based on the column names. Participants found ChatGPT to be overly reliant on using column names to make a semi-informed guess about the data type. This led to run-time errors in T3 while trying to train a model (P5, P12), and visualizations that were not meaningful for any insight-finding in T4, such as scatter-plots for categorical and boolean variables (P2, P3, P10).

Missing specification leads to misaligned expectations. Participant prompts often lack several specifications, that lead LLM responses

Table 2: Themes—obstacles and strategies when using ChatGPT for data-science tasks.

Theme	Description	Representative Examples	Participants
Obstacles when Communicating with ChatGPT			
Sharing context is difficult (Section 4.1)	Gathering relevant information to prompt with (e.g., column names, datatypes, example datapoints) takes time.	"So this one is recommending `scikit-learn` which is not a package I typically use. I usually work in `pyspark`." "It is all about the context which we provide. If we could refine it, it would do better."	P1—P6, P8, P10, P12, P14 (10 *of* 14)
ChatGPT opaquely makes assumptions (Section 4.1)	Data scientists had to correct or adjust prompts in response to unanticipated assumptions made by ChatGPT about data.	"I will provide this example for the format" "Oh, it thinks the call type is int, let me say that it is categorical"	P1—P14 (14 *of* 14)
Missing specifications lead to misaligned expectations (Section 4.1)	Missing specifications in user prompts lead ChatGPT to return generic or incorrect responses.	"I was expecting it to give me code or make a plan" "it didn't bring up `to_datetime()` and suggested `strptime()` instead!"	P1—P10, P12—P14 (13 *of* 14)
Obstacles in Leveraging ChatGPT's Responses			
Repetitive code edits (Section 4.2)	Edits of similar nature are made to adapt & validate code	"We already have pandas and the data, I'll delete these lines [...] not required" "I need a temp df to see the new columns" "ChatGPT's affirmative language is deceiving [...]"	P1—P10, P12—P14 (13 *of* 14)
Strategies for Prompting and Alternate Resources			
Techniques for prompt construction (Section 4.3.1)	Data scientists use prompting techniques and attempt scaffolding ChatGPT with domain expertise	"I can be more specific and give it only the float columns" "I will paste column names later, so that I can write the prompt first"	P1—P10, P12, P14 (11 *of* 14)
Re-using code (Section 4.3.2)	Re-use of previously authored code for new data	"automated way to run the pipeline for incoming data" "copy paste old code [...] 80% of it would be repeating"	P3, P5—P7, P11 (5 *of* 14)

to be highly temperamental to the phrasing of prompts. The presence or absence of specific words in the prompt can lead to unanticipated responses from ChatGPT. For instance, for objective tasks (T1 and T2), not mentioning "pandas" or "dataframe" in the initial prompt to ChatGPT led to code generations with APIs belonging to standard python libraries, instead of `pandas` APIs (P1, P3, P12). P12 reflected on the issue and realized that they *"didn't talk about pandas, that's why it didn't bring up* `to_datetime()` *over here and suggested* `strptime()` *instead!"* This helps us understand how participants form a mental model of the nature of responses they would get from ChatGPT.

We observe that this problem manifests in different ways for T3 (a long-form task) and T4 (an exploratory task), where participants also have more conversation turns with ChatGPT (Table 1). Most participants struggled to obtain satisfactory responses for T3 (feature selection), owing to several reasons:

First, participants expected ChatGPT to formulate an actionable plan for feature selection. However, ChatGPT frequently responded with a *"generic block of text"* (P10) on why feature selection is important, and 3–4 ways to perform it. These responses were not contextualized to the dataset at hand, bringing the onus of adding specification to prompts through planning and performing task-decomposition onto data scientists again. Second, we observe a mismatch in expected and actual response formats. Participants often expected the LLM to generate code that analyzes relations between the columns, or assesses the importance of each feature (e.g., by training a model and using feature weights). But ChatGPT rarely moved past NL responses, unless provided with specifications for code generation. Third, participants often mentioned that ChatGPT cannot be completely trusted as it lacks domain expertise (P1–P5, P7, P8, P11). ChatGPT's responses are rarely aligned with participants' domain knowledge, as it misses necessary data pre-processing steps, or provides technically incorrect responses. In one case, while prompting ChatGPT to train a model for predicting `'RESPONSE_TIME'`, P5 noticed it skipped standardization. P5 highlighted its importance: *"One thing it has missed is standardizing data, which is important for checking outliers and understanding the distribution."* Lastly, T4 (heatmap plotting) saw frequent back-and-forth prompts as participants tweaked visual details like size, colors, and label fonts. They also expected ChatGPT to suggest plot options to quickly explore the most insightful visuals.

4.2 Obstacles in Leveraging ChatGPT Responses

We now discuss challenges faced in *using* LLM generated code—where participants had to edit the generated code in certain repetitive ways to successfully adapt it to their notebooks.

Generation of repeated code. A common adaptation the participants had to make to the generated code was removing lines that were already present in their notebook. This included (1) repeated library import statements—especially `import pandas as pd`, (2) repeated data ingestion using the `pandas.read_csv()` API—where accidentally executing this code sometimes led the participants'

dataframes to be over-written, causing them to re-run all notebook cells from the beginning, and (3) performing exploratory data analysis using APIs like `pandas.DataFrame.describe()`

Mismatch in data and notebook management preferences. Participants preferred standard notebook practices—imports at the top, modular cells for viewing intermediate results, and interleaved markdown comments. ChatGPT often generated large code blocks combining imports and multiple transformations. Participants (P2, P3, P5, P7, P8, P14) split these into smaller cells and moved imports to the top to better trace errors.

Data scientists have also been studied to commonly author exploratory code; or use the fluent programming pattern, composing multiple transformations into a chain [9, 15]. Several code generations contained *functions* for data transformation tasks, with excessive number of parameters and intermediate steps, contrary to the typical exploratory and fluent patterns. P3 and P7 refactored such code generations to obtain chains of transformations.

Lacking support for code validation. Participants emphasized on the importance of thoroughly validating every operation performed using LLM-generated code. P7 mentioned that validation is even more essential as *"ChatGPT's confirmatory and affirmative language in responses—like 'Definitely! Here's the code you need'—is extremely deceiving because it doesn't really know about my data."* Though code validation took the least amount of time when compared to other activities (prompt construction and code adaptation), participants verified the functioning of generated code snippets in different ways. In some cases, ChatGPT self-generated the code snippet(s) required to validate the data transformation. In other cases, the participants wrote additional code snippets, or manually inspected the data to validate changes.

4.3 Strategies to Overcome Obstacles

We now discuss strategies employed by data scientists to overcome communication and code-adaptation obstacles.

4.3.1 Techniques for prompt construction. We observed frequent use of one-shot and few-shot prompting for T1 (datetime typecast) and T2 (split using delimiter) (P1–P4, P6, P8, P10, P12, P14); and chain of thought prompting for T3 (feature selection).

Participants often noted that LLMs lack *"any actual understanding of the data"* (see Section 4.1). To address this, some data scientists, like P10 and P14, used domain expertise to iteratively refine prompts. For example, P14 excluded columns with `'TIME'` or `'ID'` in their names to prevent ChatGPT from suggesting timestamp-based features (T3).

We also observe an instance with P5, where ChatGPT asks a ***"clarifying question"*** in response to the user provided prompt. This interaction made P5 feel assured that ChatGPT *"is able to understand the prompt"*, and *"make sense of the data context"*:

> *P5: Consider this dataset for the prediction of response time: [First 10 rows with header]*
> *ChatGPT: How is response time calculated?*
> *P5: Response time is first_onscene_datetime – incident_datetime*

4.3.2 Re-using previously authored code. Data scientists mentioned that they periodically receive new batches of data, for which they must re-run analyses and visualization reports, and re-train models (P3, P5, P6, P7, P11). In such cases, data scientists are already familiar with the structure of the data, and they reuse previously written code for data pre-processing and cleaning. P11 mentioned, *"We pretty much just copy and paste canned code for every project. A lot of variations can be created on the charts, but 80% of it would be repeating."* The need to author new code for pre-processing or analytics in such scenarios is rare. However, P7 and P11 suggest that LLMs can help automate retrieval of such snippets, and report any anomalies in newer batches of data for human-inspection.

5 Study 2: Confirmatory Survey

To validate findings from Study 1, we conducted a survey with a broader population of 114 data scientists. The confirmatory survey is aimed to enhance the generalizability of our findings from study 1, and explore whether data scientists using different tools for a wider assortment of data science tasks encountered similar benefits or challenges when utilizing LLMs for data science tasks.

5.1 Methodology

Survey protocol. We used findings from Study 1 responses to frame at least one question for each identified obstacle. The survey consisted of 8 'Agree'/'Disagree' statements, and all questions were optional to answer. Finally, to evaluate if our observations from Study 1 reached theoretical saturation, we asked respondents to indicate if they have faced any other difficulties in using LLMs for data science tasks that were not mentioned in the survey questions.

Respondents. 114 data scientists responded to the survey. After inspecting whether the respondents have meaningful prior experiences (D2), we removed 16 respondents (14%) that had no experience with using LLMs for data science tasks. We report all findings using the screened responses. 76% of the respondents self-reported having more than 5 years of experience in data science related professional roles. We discuss the survey results in Section 5.2.

5.2 Survey Results

The survey results help us understand how well do the identified obstacles (Section 4) generalize to a diverse set of data science tasks, tools, and to a broader community of data scientists. The survey respondents report experience with using LLMs for varied data science tasks spanning across project timelines, including synthetic data generation, wrangling, pre-processing, labelling, exploratory analysis, insight finding and summarization, hypothesis generation, training models, outlier detection, and generating plots.

Figure 1 shows the distribution of responses for each question. Results indicate that data scientists find sharing of relevant data context to be important (86%), as well as tedious (59%), echoing findings from Section 4.1. 62% data scientists agreed that prompting ChatGPT for columns with mixed formats is challenging, reflecting the issues discussed in Section 4.1. These challenges lead to communication breakdowns between data scientists and ChatGPT—causing them to make repeated prompt refinements (92%) (Section 4). Respondents also indicate that LLMs do not have sufficient domain expertise in data science (60%) (Section 4.1), and that generated code requires several modifications (87%) (Section 4.2).

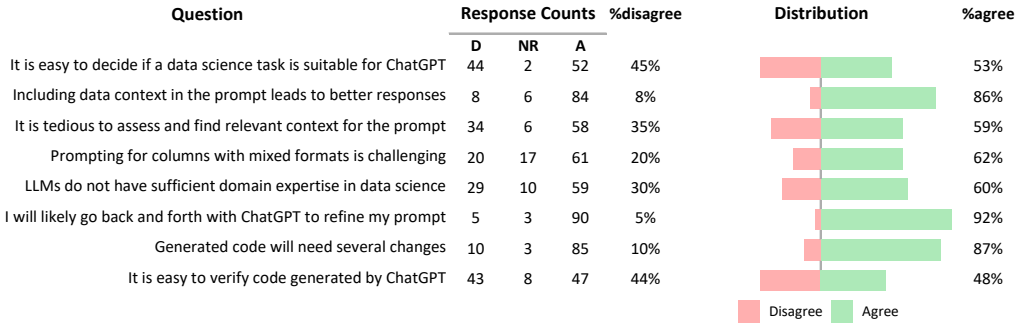

Figure 1: Distribution of survey responses. Response Counts: D (Disagree); NR (No Response); A (Agree).

6 Design Recommendations

In this section, we discuss the implications of our findings and identify the ways in which they could be adapted to various data-science environments.

Recommendation I—Provide preemptive and fluid context when interacting with AI assistants. Participants spent a significant amount of time constructing prompts (Section 4) and gathering and expressing their context (Section 4.1) to ChatGPT. Nearly 45% of the prompts included portions of data that they either manually entered, or wrote additional code snippets to obtain; however, participants did not enjoy being *"slowed-down"* in fetching this data context for every interaction. Thus, data scientists need help in providing context about their environments, and we should support mechanisms to either preemptively provide implicit context [24], or provide fluid interactions that allow data scientists to easily refer to fragments of their work in conversations with AI assistants. In data visualization, classic techniques such as *brushing* [8], enable a user to select and highlight specific data points, allowing the user to use those data points as a filter, or as a point of interest for further analysis. Recent advancements in Retrieval-Augmented Generation (RAG) can assist in dynamic retrieval of relevant data slices, code, outputs, or notes to ground AI responses [30].

Recommendation II—Provide inquisitive feedback loops and planning assistance. For transactional tasks, such as T1 (datetime typecast), ChatGPT often provided satisfactory responses. Unfortunately, long-form tasks, such as T3 (feature selection), required a more involved conversation with ChatGPT. Initial responses from ChatGPT were often "*excessively long*" and were not particularly helpful for the task. As a result, participants were uncertain how much value would be gained by continuing the conversation or what further information would help (Section 4.1). One mechanism to help data scientists have more productive conversations with AI assistants is through *inquisitive feedback loops*, where the system guides the user in expressing what they need by proactively asking the user a series of questions [22]. Recent work in agentic AI may support this need, where AI assistants take a goal-oriented role and plan multi-step solutions, check with users for missing context, and adjust their strategy dynamically [41].

Recommendation III—Provide support for code adaptation and validation. In several cases, participants performed task-decomposition, which often lead to errors, e.g., as ChatGPT always included code to ingest the dataframe—regardless of the stage of conversation—some participants accidentally executed it, leading to their data being over-written (Section 4.2). Tools for supporting *refactoring*, behavior-preserving transformations, can be one approach for assisting data scientists who must make repeated edits of similar nature, or adapt generated code snippets to their local notebook contexts (Section 4.2). For instance, GhostFactor [16] is a lightweight program analysis tool with refactoring detection algorithms that identify incomplete changes to code.

Recommendation IV—Provide transparency about shared context and domain expertise slots. Barke et al. [6] posit that lack of transparency about the shared context, and lack of control in refining the context selections can leave users in a confused state. Our participants echo a similar consideration, wherein they expressed the urge to *"get to the first response as fast as possible, but have the ability to look at what context was provided and edit it if I need to refine my prompt"* (P3, P4, P7, P12). Even with automated context management or prompt augmentation, it is important to consider how that context is ultimately shared. For example, having sufficient transparency into the shared context can enable data scientists to ensure that sensitive data does not leak into the prompt. Data-science environments interacting with AI should consider mechanisms for ensuring a two-way knowledge transfer between users and AI. For example, an AI assistant chat interface could integrate a *context panel*, a dedicated, editable grid that mirrors the AI assistant's and user's assumptions about data and tasks.

7 Conclusions

To understand fundamental obstacles, needs, and design opportunities in using AI assistants for data science, we conducted a task-based study and confirmatory survey. We found participants faced two key sets of barriers: in communicating with LLMs about data, and in adapting LLM responses to their specific situation. Participants spent considerable time collecting information to provide to ChatGPT, such as relevant dataset slices or descriptive statistics. Furthermore, ChatGPT often made assumptions unbeknownst to users, leading to bugs and additional things to remedy. Even when ChatGPT gave useful responses, adapting the code to fit existing workflows and preferences took effort. We offer key design recommendations for data science tools, such as flexible context selection, preemptive data summaries, and inquisitive feedback loops.

References

[1] 2023. Anaconda Assistant Launches to Bring Instant Data Analysis, Code Generation, and Insights to Users. https://www.anaconda.com/blog/anaconda-assistant-launches-to-bring-instant-data-analysis-code-generation-and-insights-to-users.

[2] 2023. Introducing Databricks Assistant, a context-aware AI assistant. https://www.databricks.com/blog/introducing-databricks-assistant.

[3] 2024. DataChat. https://datachat.ai/.

[4] 2024. Overview of Predictive Transformation. https://help.alteryx.com/aac/de/trifacta-classic/concepts/feature-overviews/overview-of-predictive-transformation.html.

[5] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732* (2021).

[6] Shraddha Barke, Michael B. James, and Nadia Polikarpova. 2023. Grounded Copilot: How Programmers Interact with Code-Generating Models. 7, OOPSLA1, Article 78 (April 2023), 27 pages.

[7] Rohan Bavishi, Caroline Lemieux, Roy Fox, Koushik Sen, and Ion Stoica. 2019. AutoPandas: Neural-Backed Generators for Program Synthesis. *Proc. ACM Program. Lang.* 3, OOPSLA, Article 168 (oct 2019), 27 pages.

[8] Richard A. Becker and William S. Cleveland. 1987. Brushing Scatterplots. *Technometrics* 29, 2 (1987), 127–142.

[9] Mary Beth Kery and Brad A. Myers. 2017. Exploring exploratory programming. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 25–29.

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

[11] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374 [cs.LG]

[12] Bhavya Chopra, Anna Fariha, Sumit Gulwani, Austin Z. Henley, Daniel Perelman, Mohammad Raza, Sherry Shi, Danny Simmons, and Ashish Tiwari. 2023. CoWrangler: Recommender System for Data-Wrangling Scripts. In *Companion of the 2023 International Conference on Management of Data (SIGMOD '23)*. Association for Computing Machinery, New York, NY, USA, 147–150.

[13] Noah Daniels. 2021. NY EMS Incident Dispatch Data. https://www.kaggle.com/datasets/new-york-city/ny-ems-incident-dispatch-data.

[14] Yunhe Feng, Sreecharan Vanam, Manasa Cherukupally, Weijian Zheng, Meikang Qiu, and Haihua Chen. 2023. Investigating Code Generation Performance of ChatGPT with Crowdsourcing Social Data. In *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*. 876–885.

[15] Martin Fowler. 2005. Bliki: Fluentinterface. https://www.martinfowler.com/bliki/FluentInterface.html

[16] Xi Ge and Emerson Murphy-Hill. 2014. Manual Refactoring Changes with Automated Refactoring Validation. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. Association for Computing Machinery, New York, NY, USA, 1095–1105.

[17] Paul Grice. 1991. *Studies in the Way of Words*. Harvard University Press.

[18] Ken Gu, Madeleine Grunde-McLaughlin, Andrew M. McNutt, Jeffrey Heer, and Tim Althoff. 2024. How Do Data Analysts Respond to AI Assistance? A Wizard-of-Oz Study. arXiv:2309.10108 [cs.HC]

[19] Ken Gu, Ruoxi Shang, Tim Althoff, Chenglong Wang, and Steven M. Drucker. 2024. How Do Analysts Understand and Verify AI-Assisted Data Analyses? arXiv:2309.10947 [cs.HC]

[20] Philip J. Guo, Sean Kandel, Joseph M. Hellerstein, and Jeffrey Heer. 2011. Proactive Wrangling: Mixed-Initiative End-User Programming of Data Transformation Scripts. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. Association for Computing Machinery, New York, NY, USA, 65–74. https://doi.org/10.1145/2047196.2047205

[21] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300* (2020).

[22] Austin Z. Henley, Julian Ball, Benjamin Klein, Aiden Rutter, and Dylan Lee. 2021. An Inquisitive Code Editor for Addressing Novice Programmers' Misconceptions of Program Behavior. In *Proceedings of the 43rd International Conference on Software Engineering: Joint Track on Software Engineering Education and Training (ICSE-JSEET '21)*. IEEE Press, 165–170.

[23] Noah Hollmann, Samuel Müller, and Frank Hutter. 2023. LLMs for Semi-Automated Data Science: Introducing CAAFE for Context-Aware Automated Feature Engineering. arXiv:2305.03403 [cs.AI]

[24] Rogers Jeffrey Leo John, Dylan Bacon, Junda Chen, Ushmal Ramesh, Jiatong Li, Deepan Das, Robert Claus, Amos Kendall, and Jignesh M. Patel. 2023. DataChat: An Intuitive and Collaborative Data Analytics Platform. In *Companion of the 2023 International Conference on Management of Data (SIGMOD '23)*. Association for Computing Machinery, New York, NY, USA, 203–215.

[25] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive Visual Specification of Data Transformation Scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. Association for Computing Machinery, New York, NY, USA, 3363–3372. https://doi.org/10.1145/1978942.1979444

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

[27] Majeed Kazemitabaar, Jack Williams, Ian Drosos, Tovi Grossman, Austin Henley, Carina Negreanu, and Advait Sarkar. 2024. Improving Steering and Verification in AI-Assisted Data Analysis with Interactive Task Decomposition. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology (UIST '24)*. Association for Computing Machinery, New York, NY, USA, 19 pages.

[28] 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, 11 (2018), 1024–1038.

[29] Sean Kross and Philip J. Guo. 2019. Practitioners Teaching Data Science in Industry and Academia: Expectations, Workflows, and Challenges. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–14.

[30] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.

[31] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. arXiv:2305.01210 [cs.SE]

[32] Michael Muller, Ingrid Lange, Dakuo Wang, David Piorkowski, Jason Tsay, Q. Vera Liao, Casey Dugan, and Thomas Erickson. 2019. How Data Science Workers Work with Data: Discovery, Capture, Curation, Design, Creation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–15.

[33] David Noever and Forrest McKee. 2023. Numeracy from Literacy: Data Science as an Emergent Skill from Large Language Models. arXiv:2301.13382 [cs.CL]

[34] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[35] Luigi Quaranta, Fabio Calefato, and Filippo Lanubile. 2021. KGTorrent: A Dataset of Python Jupyter Notebooks from Kaggle. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. 550–554.

[36] Itamar Shatz. Accessed 2025. Grice's Maxims of Conversation: The Principles of Effective Communication. https://effectiviology.com/principles-of-effective-communication/.

[37] Nischal Shrestha, Titus Barik, and Chris Parnin. 2021. Remote, but Connected: How #TidyTuesday Provides an Online Community of Practice for Data Scientists. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW1, Article 52 (apr 2021), 31 pages.

[38] Bogdan Vasilescu, Alexander Serebrenik, Prem Devanbu, and Vladimir Filkov. 2014. How Social Q&A Sites Are Changing Knowledge Sharing in Open Source Software Communities. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '14)*. Association for Computing Machinery, New York, NY, USA, 342–354.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[40] Cong Yan and Yeye He. 2020. Auto-Suggest: Learning-to-Recommend Data Preparation Steps Using Data Science Notebooks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 1539–1554.

[41] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

[42] Pengcheng Yin, Wen-Ding Li, Kefan Xiao, Abhishek Rao, Yeming Wen, Kensen Shi, Joshua Howland, Paige Bailey, Michele Catasta, Henryk Michalewski, Alex Polozov, and Charles Sutton. 2022. Natural Language to Code Generation in Interactive Data Science Notebooks. arXiv:2212.09248 [cs.CL]