

Towards Supporting Novice Programmers' Web Search Skills for Using Unfamiliar Libraries and APIs

Mehmet Arif Demirtaş, Jinyoung Hur, Alex Atcheson, Claire Zheng, Kathryn Cunningham

University of Illinois Urbana-Champaign

Urbana, Illinois, USA

{mad16,jhur10,samuella7,clairez7,katcun}@illinois.edu

Abstract

Programmers often rely on online help-seeking resources such as developer forums and language documentation when working with libraries and APIs. Finding relevant resources requires them to formulate search queries that effectively represent their information needs. However, factors influencing how novice programmers write search queries are not well documented.

To better understand how to support the development of search skills in novices, we investigate how novice programmers write search queries while learning about an unfamiliar data visualization library. We further examine how instructional design may influence how students formulate queries. We collected queries written by students in an introductory programming class after completing a self-directed tutorial on the Seaborn library in Python. Given a Seaborn-generated data visualization to recreate or modify, students were asked to formulate a search query they would use to find additional information on how to do so. In this work-in-progress paper, we present an exploratory study of the factors that shape the queries formulated by novice programmers. We further report insights from a subset of the data on patterns in the queries and search results as a step toward understanding how to support novice programmers in developing effective search query strategies for online help-seeking.

1. Introduction and Related Work

Modern programming projects rely on the integration of specialized libraries and APIs (application programming interfaces). Programmers commonly use online help-seeking resources to effectively use the latest libraries (Cai & Guo, 2019). Studies and surveys have reported that these resources include developer forums (e.g., Stackoverflow), online tutorials, and documentation pages (Alghamdi et al., 2025), and increasingly, AI assistants (JetBrains, 2024). Prior studies on API learning noted that performing web searches is a key part of using and understanding an API (Heinonen, 2024; Thayer et al., 2021).

Well-formulated search queries are key to retrieving relevant results. Prior literature has shown that even professional programmers can struggle to ask efficient questions for retrieving relevant information. For instance, Duala-Ekoko & Robillard (2012) found that developers use API elements as terms in their queries and that they encounter difficulties when searching for unfamiliar concepts that is difficult to relate to familiar API elements. Kelleher & Brachman (2023) observed that programmers were able to find information relevant to the fundamental elements of an API, but struggled to form effective debugging-related questions when errors arose during project work. Gao et al. (2020) examined web search as the initial step in obtaining and organizing knowledge about an API, and observed that ineffective searches can trigger longer search sessions and hinder learning by increasing cognitive load. Extended search sessions may also lead to users getting discouraged and losing confidence in the results (Jiang et al., 2014). In an analysis of query logs across search engines, White et al. (2009) found that users with domain expertise in computer science performing domain-relevant searches were more likely to include domain-specific terms in their queries, wrote longer queries on average, and had higher rates of success in retrieving relevant results compared to non-experts. Yet, it remains unclear how novice programmers develop the skill to write effective queries despite limited domain expertise. Notably, all of these studies focused on individuals with prior programming experience. The challenges identified in these works may be amplified for novice programmers who, in addition to learning to use APIs, are also learning to program, the challenges of which are many and well-documented (Robins, 2019).

In the current study, we investigate how novice programmers write search queries in the context of an

information-seeking task framed around an unfamiliar library and how instructional design decisions influence query formulation. To do so, we collected queries written by students in an undergraduate-level introductory programming course for non-CS majors. During the study, students were presented with three data visualizations generated using the Seaborn library in Python and, for each task, asked to write a query they would use to find relevant information on modifying or recreating elements of the visualizations. Prior to completing the search tasks, students completed one of two self-directed tutorials. One tutorial followed the structure of a popular online resource, covering individual Seaborn API functions in isolation, while the other was organized around subgoal labels—descriptions of the goals of short code segments—to cover authentic data visualization examples that drew on several functions at once. Subgoal labels have been argued to support students’ ability to explain code by providing relevant language (Cunningham et al., 2021), and have been shown to support learning outcomes for students with minimal programming experience by reducing focus on surface-level problem details (Margulieux et al., 2020). However, reducing focus on problem details may discourage familiarization with language syntax and hinder effective searching, as there is evidence to suggest that referencing API elements in queries leads to effective searches (Duala-Ekoko & Robillard, 2012).

To better understand how tutorial organization may influence search behavior, we conducted an exploratory analysis comparing queries formulated by novice programmers in the context of an information-seeking task. Specifically, we posed the following two research questions:

- RQ1: What patterns can be observed in search queries written by novice programmers using an unfamiliar library?
- RQ2: How do teaching approaches influence the way novice programmers write search queries?

2. Methodology

We collected search queries in a randomized controlled trial in an introductory programming course for non-CS majors in the Spring of 2026.

2.1. Experimental Design

The study took place during hands-on lab sessions for the course. Students did not receive prior instruction on the Seaborn library¹. Instead, they first completed a self-directed tutorial introducing several concepts that were curated with instructor approval: creating two types of visualizations (bar plot and box plot), displaying multiple visualizations from the same dataset in a grid, and customizing the colors and labels of a plot. Students were randomly assigned to one of the two tutorial conditions. The tutorials covered the same content, but used different instructional approaches.

2.1.1. Tutorial Content

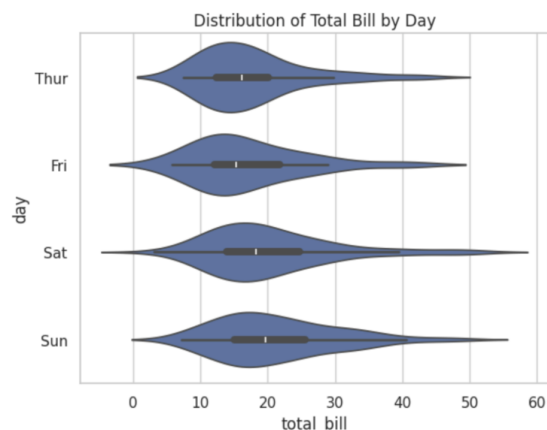
One group of students completed a tutorial designed with a *bottom-up* approach reflecting the structure of popular online tutorials. This tutorial was adapted from an online resource on visualizing data with Seaborn (Ali, 2023). The tutorial began with an overview of the activity and highlighted individual library functions in subsequent pages, covering the functions required for achieving the learning objectives. This organization emphasized API elements, such as function names, on the tutorial pages.

Students who were not assigned to the bottom-up condition completed the *top-down* condition. In the top-down condition, the tutorial utilized *subgoal labels*. This tutorial was modeled after *purpose-first tutorials* (Demirtas et al., 2026), which uses subgoal labels to emphasize the purpose of authentic code examples for teaching non-CS majors about computing applications. To this end, the top-down tutorial began with two examples of data visualizations showcasing the use of several API functions. These examples were annotated with subgoal labels, breaking down the larger programs into smaller “chunks”. Students studied these subgoal chunks in subsequent tutorial pages.

Both tutorials included an identical set of interactive exercises for students to practice. After completing the tutorials, students were presented with a post-assessment to measure their understanding of Seaborn.

¹Earlier in the course, students received instruction on the basics of `matplotlib`, another Python visualization library.

Q3.2. You saw the following plot on a blog post, and you want to find out how to create a similar plot for your presentation. You want to find the example code that will generate this type of plot.



Q3.2a. Describe this problem in your own words (e.g. similar to how you would ask for help from a friend). Assume that you cannot share images, and describe the images if necessary.

Q3.2b. How confident are you in your ability to solve this problem without seeking help (e.g., using any search engines)?

None at all A little A moderate amount A lot A great deal

I feel confident that I would be able to solve this problem without using search engines (e.g. Google)

☐ ☐ ☐ ☐ ☐

Q3.2c. Imagine that the below box is the Google search bar. If you were to use Google to help you solve this problem, what is the first thing you would search? Enter your search query below.

Figure 1 – The search task presented to students. The query data is collected from answers to the last textbox (Q3.2c).

2.1.2. Search Tasks

Queries were collected for three search tasks, which were completed during the post-assessment. For each task, students were presented with a scenario and a target data visualization. The scenarios required students to modify existing code or write code from scratch to replicate the target data visualization. The scenarios were written without technical instructions or API terminology, requiring students to infer the necessary modifications from the visualization. For each scenario, students were asked to (i) describe the task in their own words, (ii) rate their confidence in their ability to complete the task without searching for resources, and (iii) write the query they would use in an initial attempt to find relevant resources for solving the task. Queries were collected on a survey; students did not execute their searches.

Search tasks varied in difficulty and the type of information provided. In this work-in-progress, we focus our analysis on one of the three tasks. This task included a target data visualization of a violin plot, a plot type that was not covered in the tutorial. Students were asked to search for information that would help them replicate the provided visualization. The task instructions are shown in Figure 1.

2.2. Exploratory Analysis

Our analysis focuses on the search task centered around the violin plot because it requires *far transfer* (Barnett & Ceci, 2002): in other words, students must apply their data visualization knowledge in an unfamiliar context. Challenges with far transfer have been enumerated in prior work about API learning, which found that programmers struggled more with finding information about new API elements than with exploring details of API elements with which they were already familiar (Duala-Ekoko & Robillard, 2012). Of the plot types covered in the tutorial, the type most similar to a violin plot is a box plot. The bottom-up tutorial introduced box plots in general terms as a way of comparing the distribution of one or more variables across different categories. The top-down tutorial with subgoal labels emphasized the *purpose* of box plots in specific scenarios, introducing it in an example about comparing restaurants with consistent ratings across time. Given this difference, we were interested in knowing whether students exposed to different tutorial organizations use contrasting vocabulary to describe a new visualization type and achieve varying success at retrieving relevant resources. We present this preliminary analysis as an exploration of student querying behavior for a concept that is distant from the examples with which they may already be familiar.

2.2.1. Data Pre-Processing and Search Results

The collected dataset included queries from 372 students. 178 students had completed the bottom-up tutorial, and 194 students had completed the top-down tutorial that uses subgoal labels.

Search engines rank the pages in search results to prioritize pages that are most likely to be useful to the users, and users are more likely to visit pages that are ranked higher on the results pages. Therefore, we only look at the first 10 pages from the search results of each query. We collected these pages by executing a Google search for each query in our dataset using SerpAPI,². For each result, we extracted the title, link, text snippet, and list position. To reduce the influence of misspellings, we also extracted the Google-suggested query when one was provided—that is, the query used by Google for which the results were actually retrieved—and used it in place of the originally authored query in our later analysis. For word-level analyses, we removed all punctuation from the queries and applied stemming to capture the most general form of each word.

2.2.2. Uniqueness of Retrieved Results

We first examine how many unique pages were retrieved by students in their search results. There were 1468 unique results retrieved by 372 queries. Of these results, 293 were retrieved by queries from both top-down and bottom-up conditions. On the other hand, 622 results only appeared for queries from the top-down condition, while 553 results only appeared for queries from the bottom-up condition, suggesting that collected queries could retrieve a diverse set of search results. Even though there are many pages unique to one condition among all search results, most students would see pages from a smaller set of search results. When we examine the pages that appear as the top search result, we see that 71% of 372 queries retrieved a page that could be retrieved by a query from either condition. Considering that users are more likely to check the top result compared to the rest of their search results (Abualsaud & Smucker, 2019), the tutorial conditions might be less impactful on search success than expected.

2.2.3. Relevance of Retrieved Results

We use the relevance of retrieved search results to the search task as a measure of success. We define a relevant search result for this task as a page with a reference to a violin plot. To this end, we analyzed the links that would be displayed on the first page of Google search results for each query to measure how successful these queries would be at getting relevant information.

In our preliminary analysis, we looked at the top 30 most common pages in search results for both conditions. In both conditions, a majority of these pages were not relevant to the task: 40% of pages retrieved by top-down queries (12 out of 30) and 30% of pages retrieved by bottom-up queries (9 out of 30) contained a reference to violin plots.

Moreover, most of the students did not have one of these relevant pages in their top search results. 24.5% of students had one of the relevant pages as their top result, and 56.7% of students had at least one relevant page within their top 10 search results. While it is possible that students would retrieve relevant pages not captured by our analysis, this finding provides a lower bound that shows how difficult searching for relevant information can be for students.

23.9% of queries from the top-down condition had one of these relevant pages as the top search result. In contrast, only 16.5% of queries from the bottom-up condition had one of these relevant resources as the top result. However, 47.5% of top-down and 47.8% of bottom-up condition queries had one or more of these relevant results on the first page of search results.

2.2.4. Patterns in Search Queries

To explore the patterns in novice search queries, we built three exploratory linear regression models. Model 1 predicts the query length as the dependent variable. Models 2 and 3 predict the frequencies of API elements as search terms and data visualization jargon as search terms, respectively. For our analysis, the list of API elements includes terms from the code examples (e.g., classes, functions, parameters) originating from the Seaborn documentation related to our topics, and the list for data visualization jargon includes terms used in the explanations on the same pages. Then, each search query is assigned a relative frequency per list, defined as the number of the terms from a list normalized by the length of

²<https://serpapi.com/>

the query. These dependent variables were selected to capture any differences due to the instructional design in the bottom-up tutorial, which highlights API elements, and the top-down tutorial, which describes code at a higher level with subgoal labels. Predictors for all three models are the tutorial type (bottom-up vs top-down with subgoals) and self-reported confidence in completing the given search task without web searching, in addition to the self-reported demographic information on whether the student had programming experience before the class, frequency of search engine use, and frequency of AI assistant use. Queries from two students were excluded due to missing demographic data.

In our analysis, the only significant predictor of query length was self-reported confidence in completing the given search task without web searching ($p = .018, R^2 = .023$). Students who reported a higher confidence in completing the task without support used fewer words. There were no significant predictors for the frequency of code terms ($R^2 = .016$) and jargon terms ($R^2 = .009$). Notably, the tutorial condition was not a significant predictor for any of the dependent variables, indicating that students' search behavior for a new library may not be directly influenced by instruction on the specific library. Moreover, our models had low R^2 values, suggesting that the factors we explored might not account for the variability in students' search term usage (see Appendix A).

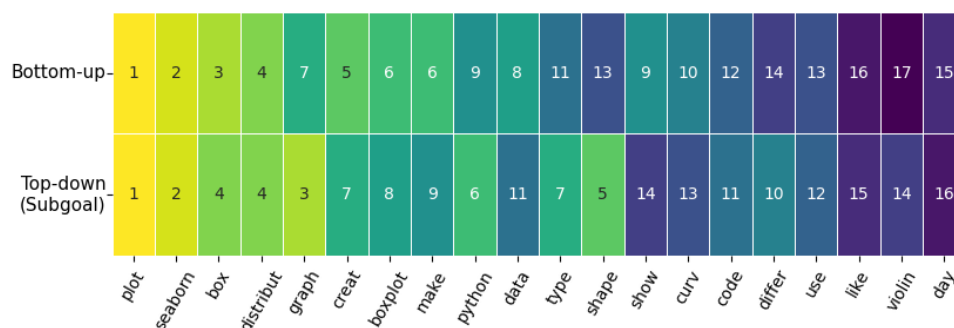


Figure 2 – Top 20 search terms by tutorial type. Terms are ordered from most common (left) to least common (right) for the full dataset. The numbers on the top row represent the rank of each term for queries submitted by students who completed the bottom-up tutorial, and the bottom row represents the ranks of terms in queries submitted by students who completed the top-down/subgoal tutorial.

Commonly Used Search Terms. In addition to the frequencies of types of terms and query lengths, we look at the 20 most common terms for queries submitted by students who completed the tutorials (Figure 2). Both conditions had the same set of words in their top 20, but some words (such as ‘graph’, ‘shape’, and ‘type’) were more common in the top-down/subgoal condition, and others (such as ‘create’ and ‘data’) were used more commonly in the bottom-up condition. However, we were not able to observe any consistent trends between the use of these terms and the relevance of the retrieved search results.

3. Discussion

This work presents our ongoing work on how novice programmers write search queries for unfamiliar libraries. Our initial findings suggest that instructional design choices may influence students' efficiency at retrieving relevant search results to some extent. However, our preliminary analysis of patterns in search queries did not reveal significant differences in query lengths, frequencies of API elements and domain terminology as keywords, or commonly used search terms. These results might imply that the success of search results is sensitive to minor changes in search queries, motivating further investigation. For instance, exploring the effects of word ordering or comparing frequencies of n-grams as opposed to individual words in search queries may reveal patterns that affect the success of students at search tasks. One aspect of our work is that data visualization tasks may require three types of knowledge: programming knowledge, knowledge about data visualization types, and knowledge about the particular data visualization library (Seaborn). Future work can consider how the learning process differs between libraries that require different types of prior knowledge.

What makes a search query ‘successful’ is still an open question, as previous work mainly focuses on programmers’ search behaviors in think-aloud sessions as opposed to assessing individual search queries. These studies (e.g., Gao et al. (2020)) evaluate the effectiveness of a search query based on whether the programmer conducts a new search immediately after the current one, or whether they stay on the pages they access as a result of the search query. However, this might be overly strict, as studies on web searches in other contexts, such as seeking medical information (White & Horvitz, 2009), indicate that an effective search session may include several successive searches that are informed by the results of the previous query searches. Moreover, novices may struggle to recognize relevant information in complex tasks even when it is present in a resource (Kiani et al., 2019), adding another layer of complexity to whether a query is successful. Thus, we believe that evaluating programmers’ web search skills on a query-level and exploring the patterns in these queries is a promising avenue for the development of instructional interventions.

Increasing prevalence of AI assistants for information seeking raises two questions about the importance of web search skills in next-generation programming environments. First, students may defer to AI assistants for search tasks, as opposed to traditional search engines. How users decide between web search and generative AI is being actively investigated (Zhou & Li, 2026), and whether either option provides benefits for learning or productivity when working with unfamiliar libraries remains to be explored. Second, students may use AI-assisted coding environments to obtain code as opposed to seeking information that will help them write code. Even in this case, there are similarities between using AI assistants for coding and seeking information online, as discussed by Sarkar et al. (2022). In both settings, the help request starts with a query written in a natural language, and the user needs to evaluate the output and identify what parts of the output are relevant to the problem at hand. Thus, certain traditional web search skills may transfer to AI-assisted information seeking and coding environments. However, Sarkar et al. (2022) also highlights the *mixed media experience* in traditional search engines, where code snippets shared via text are one type of media out of many, such as forum discussions, videos, and images. Therefore, users might develop mechanisms that leverage several forms of information simultaneously for processing traditional search results and identifying relevant information. In contrast, Sarkar et al. (2022) describes AI assistance in coding as a *fixed media experience* where the written code and descriptions are the main form of output presented to the user. In this case, some skills developed in traditional web search might not be applicable in AI-assisted search tasks. Research on search skills should consider the skills that will be required to explain the new paradigm of AI-assisted search and coding environments.

References

- Abualsaud, M., & Smucker, M. D. (2019). Patterns of search result examination: Query to first action. In *Proceedings of the 28th acm international conference on information and knowledge management* (p. 1833–1842). New York, NY, USA: Association for Computing Machinery. Retrieved from <https://doi-org.proxy2.library.illinois.edu/10.1145/3357384.3358041> doi: 10.1145/3357384.3358041
- Alghamdi, S. S., Bull, C., & Kharrufa, A. (2025). *Designing for self-regulation in informal programming learning: Insights from a storytelling-centric approach*. Retrieved from <https://arxiv.org/abs/2507.22671>
- Ali, M. (2023). *Python seaborn tutorial for beginners: Start visualizing data*. Retrieved 2026-05-10, from <https://www.datacamp.com/tutorial/seaborn-python-tutorial>
- Barnett, S. M., & Ceci, S. J. (2002). When and where do we apply what we learn?: A taxonomy for far transfer. *Psychological Bulletin*, 128(4), 612–637. doi: 10.1037/0033-2909.128.4.612
- Cai, C. J., & Guo, P. J. (2019). Software developers learning machine learning: Motivations, hurdles, and desires. In *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VLHCC)* (p. 25-34). doi: 10.1109/VLHCC.2019.8818751

- Cunningham, K., Ericson, B. J., Agrawal Bejarano, R., & Guzdial, M. (2021, May). Avoiding the Turing Tarpit: Learning Conversational Programming by Starting from Code's Purpose. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (pp. 1–15). Yokohama Japan: ACM. doi: 10.1145/3411764.3445571
- Demirtas, M. A., Zheng, C., & Cunningham, K. (2026, February). Teaching Authentic Programming Applications to Novices: Purpose-first Tutorials in a General Education Computing Course. In *Proceedings of the 57th ACM Technical Symposium on Computer Science Education V.1* (Vol. 1, pp. 274–280). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3770762.3772574
- Duala-Ekoko, E., & Robillard, M. P. (2012, June). Asking and answering questions about unfamiliar APIs: An exploratory study. In *2012 34th International Conference on Software Engineering (ICSE)* (pp. 266–276). Zurich: IEEE. doi: 10.1109/ICSE.2012.6227187
- Gao, G., Voichick, F., Ichinco, M., & Kelleher, C. (2020). Exploring Programmers' API Learning Processes: Collecting Web Resources as External Memory. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 1–10). doi: 10.1109/VL/HCC50065.2020.9127274
- Heinonen, A. (2024). Understanding apis and the software that provides them - analysis of programmers' api mental models used in programming tasks. In *Proceedings of the 35th Annual Workshop of the Psychology of Programming Interest Group (PPIG 2024)*.
- JetBrains. (2024). *Python Developers Survey 2024 Results*. Python Software Foundation and JetBrains PyCharm. Retrieved 2026-05-19, from <https://lp.jetbrains.com/python-developers-survey-2024/>
- Jiang, J., He, D., & Allan, J. (2014). Searching, browsing, and clicking in a search session: changes in user behavior by task and over time. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval* (pp. 607–616). doi: 10.1145/2600428.2609633
- Kelleher, C., & Brachman, M. (2023, January). A sensemaking analysis of API learning using React. *Journal of Computer Languages*, 74, 101189. doi: 10.1016/j.cola.2022.101189
- Kiani, K., Cui, G., Bunt, A., McGrenere, J., & Chilana, P. K. (2019). Beyond "one-size-fits-all": Understanding the diversity in how software newcomers discover and make use of help resources. In *Proceedings of the 2019 chi conference on human factors in computing systems* (p. 1–14). New York, NY, USA: Association for Computing Machinery. Retrieved from <https://doi-org.proxy2.library.illinois.edu/10.1145/3290605.3300570> doi: 10.1145/3290605.3300570
- Margulieux, L. E., Morrison, B. B., & Decker, A. (2020, May). Reducing withdrawal and failure rates in introductory programming with subgoal labeled worked examples. *International Journal of STEM Education*, 7(1), 19. doi: 10.1186/s40594-020-00222-7
- Robins, A. V. (2019). Novice Programmers and Introductory Programming. In S. A. Fincher & A. V. Robins (Eds.), *The Cambridge Handbook of Computing Education Research* (pp. 327–376). Cambridge University Press. doi: 10.1017/9781108654555.013
- Sarkar, A., Gordon, A. D., Negreanu, C., Poelitz, C., Srinivasa Ragavan, S., & Zorn, B. (2022, September). What is it like to program with artificial intelligence? In *Proceedings of the 33rd Annual Conference of the Psychology of Programming Interest Group (PPIG 2022)*.
- Thayer, K., Chasins, S. E., & Ko, A. J. (2021, January). A Theory of Robust API Knowledge. *ACM Trans. Comput. Educ.*, 21(1), 8:1–8:32. doi: 10.1145/3444945

White, R. W., Dumais, S. T., & Teevan, J. (2009). Characterizing the influence of domain expertise on web search behavior. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining* (pp. 132–141). doi: 10.1145/1498759.1498819

White, R. W., & Horvitz, E. (2009, November). Cyberchondria: Studies of the escalation of medical concerns in web search. *ACM Trans. Inf. Syst.*, 27(4). doi: 10.1145/1629096.1629101

Zhou, T., & Li, S. (2026). Understanding user switch of information seeking: From search engines to generative ai. *Journal of Librarianship and Information Science*, 58(1), 696-708. Retrieved from <https://doi.org/10.1177/09610006241244800> doi: 10.1177/09610006241244800

A. Linear Regression Results

Query length:

OLS Regression Results						
=====						
Dep. Variable:	query_length	R-squared:	0.023			
Model:	OLS	Adj. R-squared:	0.009			
Method:	Least Squares	F-statistic:	1.683			
Date:	Tue, 19 May 2026	Prob (F-statistic):	0.138			
Time:	11:11:48	Log-Likelihood:	-1204.4			
No. Observations:	370	AIC:	2421.			
Df Residuals:	364	BIC:	2444.			
Df Model:	5					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]

Intercept	14.3838	1.303	11.042	0.000	11.822	16.945
C(condition)[T.pfp]	-0.5091	0.661	-0.771	0.441	-1.808	0.790
experience_binary	-0.7308	0.680	-1.075	0.283	-2.068	0.606
task_confidence	-0.8858	0.373	-2.372	0.018	-1.620	-0.151
traditional_search_frequency	-0.1599	0.340	-0.470	0.639	-0.829	0.509
llm_assistant_frequency	-0.1095	0.270	-0.405	0.685	-0.641	0.422

Frequencies of code terms:

OLS Regression Results						
=====						
Dep. Variable:	code_match_norm	R-squared:	0.016			
Model:	OLS	Adj. R-squared:	0.002			
Method:	Least Squares	F-statistic:	1.159			
Date:	Tue, 19 May 2026	Prob (F-statistic):	0.329			
Time:	11:11:48	Log-Likelihood:	31.168			
No. Observations:	370	AIC:	-50.34			
Df Residuals:	364	BIC:	-26.85			
Df Model:	5					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]
Intercept	0.2245	0.046	4.861	0.000	0.134	0.315
C(condition)[T.pfp]	0.0107	0.023	0.458	0.647	-0.035	0.057
experience_binary	0.0242	0.024	1.003	0.317	-0.023	0.072
task_confidence	0.0248	0.013	1.874	0.062	-0.001	0.051
traditional_search_frequency	0.0041	0.012	0.338	0.735	-0.020	0.028
llm_assistant_frequency	0.0062	0.010	0.647	0.518	-0.013	0.025

Frequencies of data visualization jargon terms:

OLS Regression Results						
=====						
Dep. Variable:	jargon_match_norm	R-squared:	0.009			
Model:	OLS	Adj. R-squared:	-0.005			
Method:	Least Squares	F-statistic:	0.6631			
Date:	Tue, 19 May 2026	Prob (F-statistic):	0.652			
Time:	11:11:48	Log-Likelihood:	48.390			
No. Observations:	370	AIC:	-84.78			
Df Residuals:	364	BIC:	-61.30			
Df Model:	5					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]
Intercept	0.2535	0.044	5.749	0.000	0.167	0.340
C(condition)[T.pfp]	0.0178	0.022	0.796	0.427	-0.026	0.062
experience_binary	0.0258	0.023	1.121	0.263	-0.019	0.071
task_confidence	0.0140	0.013	1.107	0.269	-0.011	0.039
traditional_search_frequency	0.0006	0.012	0.053	0.957	-0.022	0.023
llm_assistant_frequency	-2.536e-05	0.009	-0.003	0.998	-0.018	0.018